DAIANE GONÇALVES FERREIRA

MÉTODOS DE OTIMIZAÇÃO DE TERCEIRA ORDEM

CAMPINAS
2013

UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA

DAIANE GONÇALVES FERREIRA

MÉTODOS DE OTIMIZAÇÃO DE TERCEIRA ORDEM

Orientadora: Profa. Dra. Margarida Pinheiro Mello
Coorientadora: Profa. Dra. Maria Aparecida Diniz Ehrhardt

Dissertação de mestrado apresentada ao Instituto de Matemática, Estatística e Computação Científica da Unicamp para obtenção do título de Mestra em Matemática Aplicada.

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELA ALUNA DAIANE GONÇALVES FERREIRA, E ORIENTADA PELA PROF. DRA. MARGARIDA PINHEIRO MELLO.

Assinatura do Orientador

Margarida P. Mello

Assinatura do Coorientador

M.A. Diniz Ehrhardt

CAMPINAS
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

F413m

Ferreira, Daiane Gonçalves, 1988-
Métodos de otimização de terceira ordem / Daiane Gonçalves Ferreira. – Campinas, SP : [s.n.], 2013.

Orientador: Margarida Pinheiro Mello.
Coorientador: Maria Aparecida Diniz Ehrhardt.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Derivadas de ordem superior. 2. Halley, Método de. 3. Hipérboles tangentes. 4. Otimização matemática. I. Mello, Margarida Pinheiro,1957-. II. Ehrhardt, Maria Aparecida Diniz,1956-. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. IV. Título.

Informações para Biblioteca Digital

Título em inglês: Third order optimization methods
Palavras-chave em inglês:
High order derivative
Halley's method
Tangent hyperbolas
Mathematical optimization
Área de concentração: Matemática Aplicada
Titulação: Mestra em Matemática Aplicada
Banca examinadora:
Margarida Pinheiro Mello [Orientador]
Márcia Aparecida Gomes Ruggiero
Ernesto Julián Goldberg Birgin
Data de defesa: 27-03-2013
Programa de Pós-Graduação: Matemática Aplicada

Dissertação de Mestrado defendida em 27 de março de 2013 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.

Margarida P. Mello

Prof.(a). Dr(a). MARGARIDA PINHEIRO MELLO

Prof.(a). Dr(a). MÁRCIA APARECIDA GOMES RUGGIERO

Prof.(a). Dr(a). ERNESTO JULIÁN GOLDBERG BIRGIN

À minha família, pois família é tudo!

"Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas graças a Deus, não sou o que era antes."

Marthin Luther King

Agradecimentos

Mais uma etapa de minha vida vencida. Só quem acompanhou de perto minha trajetória sabe o quanto eu lutei para chegar até aqui. Não foi nada fácil e, como em toda batalha, eu não teria obtido nenhum sucesso se estivesse sozinha nesta empreitada. Eu não posso deixar de agradecer à todas as pessoas que estiveram ao meu lado e me ajudaram durante os últimos seis anos, uns diretamente em meu trabalho, outros pelo simples fato de estarem ao meu lado me dando força para seguir.

Agradeço primeiramente a Deus e à Nossa Senhora que sempre estiveram ao meu lado, guiando meus passos e iluminando meu caminho. Foram muitos os momentos que, quando nada parecia dar certo, uma simples oração resolveu tudo. Se minha fé não fosse tão grande certamente eu não teria chegado até aqui e é essa mesma fé que me faz seguir e acreditar que vou alcançar todos os meus objetivos, realizar todos os meus sonhos.

Posso dizer que possuo muita sorte pois eu tive mais do que alguém para me ajudar, eu tive uma irmã para lutar junto comigo, desde o ventre da nossa mãe, e esse é um privilégio de poucos. Ter uma irmã gêmea é uma benção, está muito além de apenas ter uma irmã, ainda mais quando ela tem a mesma paixão pela matemática que eu. Sem medo de estar sendo injusta com todos que já passaram pela minha vida, posso dizer que a Deise é a pessoa a quem mais devo prestar meus agradecimentos, essa vitória não foi minha, foi nossa.

Além de ter mais do que uma irmã, posso dizer que também tive mais do que uma simples orientadora, ela é também uma amiga e confidente como nunca tive. Trabalhar com a Margarida foi um prazer e pude aprender e crescer muito durante os dois anos do mestrado. Em cada cantinho deste texto é possível encontrar um toque dela, sem os quais o texto certamente estaria mais pobre. Tenho certeza que vou levar seus ensinamentos para toda minha vida e dizer apenas obrigada é pouco diante de tanta dedicação e carinho que recebi.

Eu não poderia deixar de agradecer ao Sr. Joaquim Miranda, diretor da escola em que cursei minha educação básica, que acreditou no nosso potencial e não mediu esforços para nos ajudar a chegar onde chegamos. Sem sua ajuda tudo teria sido muito mais difícil e talvez eu nem estivesse aqui hoje, foi um anjo que Deus colocou na nossa vida.

Agradeço à banca de qualificação e à banca de defesa, Márcia Aparecida Gomes Ruggiero, Antonio Carlos Moretti e Ernesto Julián Goldberg Birgin, pelas correções e dicas, foram de grande importância para a melhoria deste trabalho.

Agradeço à minha família, ao meu noivo Agnaldo e a todos meus amigos, que me apoiaram em todas as minhas decisões e estiveram sempre ao meu lado, me dando muita força para seguir.

Agradeço a todos os professores, da educação básica e superior, que contribuíram diretamente na minha formação fornecendo toda a base necessária para a realização desse trabalho. Eu poderia citar os nomes daqueles pelos quais guardo um carinho mais especial, mas prefiro não fazê-lo pois posso ser injusta e esquecer de alguém.

Agradeço à UNICAMP e ao IMECC por toda estrutura e apoio oferecidos, principalmente pelos funcionários da secretaria de graduação e de pós-graduação. Foi de fundamental importância para o bom desenvolvimento deste trabalho.

E por último, mas não menos importante, agradeço à FAPESP pelo apoio financeiro recebido através do processo 2011/03911-9.

Resumo

Métodos de Otimização de terceira ordem, embora de longa tradição, eram considerados, até passado recente, impraticáveis, devido à taxa com que o esforço computacional cresce em função da dimensão do problema. Avanços no desenvolvimento de estruturas de dados, rotinas que trabalham com estas estruturas e a exploração da esparsidade de grande parte dos problemas encontrados na prática já permitem implementações destes métodos que podem torná-los competitivos com métodos de segunda ordem. O objeto desta dissertação é a apresentação do método de Halley, um método de terceira ordem, sua implementação em MATLAB e a realização de testes computacionais, visando uma comparação empírica de sua eficiência frente ao método de Newton, o método de segunda ordem mais empregado na atualidade.

Abstract

Higher order optimization methods, though of long-standing tradition, until recently have been deemed impractical, due to the rate of increase of the computational effort as a function of the size of the problem. Advances in the development of data structures, routines that work with these structures and the use of the sparsity of a vast range of practical problems have led to implementations of these methods that are competitive with second order methods. The object of this dissertation is the study of Halley's method, a third-order method, the development of a MATLAB implementation thereof and its testing, aiming at an empirical comparison of its efficiency against that of Newton's method, the second-order method most widely used today.

Sumário

Introdução

Quando estamos trabalhando com problemas de otimização não-linear há sempre uma grande preocupação com relação à escolha do melhor método a ser empregado na resolução dos problemas. É importante que o método forneça uma boa aproximação da solução e, ao mesmo tempo, é fundamental que o custo computacional não seja muito alto para que mesmo problemas de grande porte, que são muito comuns na prática, possam ser resolvidos.

Na atualidade, o método mais empregado para a solução do problema de otimização não-linear sem restrições

$$\begin{array}{ll} \min & f(x) \\ x \in \mathbb{R}^n & \end{array} \tag{1}$$

é o método de Newton, que sempre funcionou muito bem na prática. Este método é aplicado para a classe C^2 de funções, utiliza as derivadas de primeira e segunda ordem e possui convergência quadrática. Sua fórmula iterativa é dada por

$$x_{x+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k), \qquad k = 0, 1, \ldots.$$

Existem na literatura diversos métodos baseados no uso de derivadas de ordem superior que, apesar de possuírem uma ordem de convergência maior e terem surgido à mesma época que o método de Newton, não costumam ser utilizados para a resolução de problemas aplicados na atualidade.

Um dos métodos mais populares baseado no uso de derivadas de ordem superior é o método de Halley, proposto por Edmond Halley, um brilhante astrônomo e matemático, famoso por ter calculado a órbita do cometa que recebe o seu nome, ver [38]. Sua fórmula iterativa é dada por

$$x_{k+1} = x_k - [I - \frac{1}{2}\nabla^2 f(x_k)^{-1} \nabla^3 f(x_k) \nabla^2 f(x_k)^{-1} \nabla f(x_k)]^{-1} \nabla^2 f(x_k)^{-1} \nabla f(x_k), \;\; k = 0, 1, \ldots.$$

O método de Halley é aplicável à classe C^3 de funções e além de utilizar as derivadas de primeira e segunda ordem, utiliza também derivadas de terceira ordem em seu processo iterativo, o que aumenta muito o custo por iteração à medida que a dimensão cresce, uma vez que envolve o cálculo de n^3 derivadas parciais de terceira ordem para uma função com n variáveis.

Em geral, o método de Halley necessita de menos iterações do que os métodos de segunda ordem, como o método de Newton e suas variantes, para atingir a mesma precisão.

Por outro lado, o custo por iteração e requerimento de memória são maiores, pois além do cálculo de n elementos do gradiente e $\frac{n(n+1)}{2}$ elementos da matriz Hessiana, realizado no método de Newton, também são calculados, a princípio, $\frac{2n+3n^2+n^3}{6}$ elementos do tensor a cada iteração, para uma função de n variáveis. O alto custo computacional do método Halley, para problemas de grande porte, justificaria o fato do método não ser muito empregado na atualidade.

No Capítulo 1, apresentamos alguns aspectos teóricos mais relevantes sobre os métodos de Newton e Halley para zero de funções reais de uma variável real, com abordagem algébrica e geométrica de ambos os métodos, bem como condições para a convergência local e as respectivas taxas de convergência. No final do capítulo, apresentamos alguns testes numéricos realizados no MATLAB nos quais observamos uma melhor eficiência e robustez do algoritmo de Halley, neste conjunto de testes, comparado ao algoritmo de Newton.

No Capítulo 2, abordamos o caso geral do problema de localização de zeros de função, onde $F\colon \mathbb{R}^n \to \mathbb{R}^n$, é uma função vetorial de várias variáveis reais. Neste caso, as fórmulas iterativas dos métodos de Newton e Halley para uma variável possuem uma extensão natural para o caso em que F é uma função vetorial real.

No Capítulo 3 introduzimos o problema de otimização não-linear sem restrição e fazemos a ponte entre o problema de localização de zero de função e o problema de otimização não-linear sem restrições. A resolução do problema de localização dos pontos estacionários da função, ou seja, os zeros do vetor gradiente, vai fornecer candidatos ao ponto ótimo.

No Capítulo 4 criamos uma classe de funções para a realização de uma bateria de testes preliminares, com o objetivo de aprimorar as implementações dos algoritmos. Versões mais eficientes destas últimas são obtidas fazendo uso da simetria e esparsidade das matrizes (resp., matrizes e tensores) envolvidos na fórmula iterativa de Newton (resp., Halley). Problemas, onde as matrizes envolvidas no processo iterativo são esparsas, são muito comuns na prática para problemas de grande porte, e fazer uso desta estrutura para eliminar operações com os elementos nulos diminui consideravelmente o esforço computacional. Estudos recentes, ver [16, 17, 18], mostram que o uso da estrutura de esparsidade e simetria dos tensores das funções pode tornar os métodos de terceira ordem competitivos com o método de Newton. Posteriormente, em nossos testes principais, com problemas retirados da literatura, utilizamos as implementações que se mostraram mais eficientes.

Finalmente, no Capítulo 5 apresentamos o conjunto de problemas para a realização da fase final de testes do nosso trabalho. Foram selecionadas dez classes de funções com dimensão variável, sendo seis da coleção CUTE [7] e quatro da coleção apresentada em [5] para otimização irrestrita. Para cada problema resolvido, apresentamos a média do número de iterações e tempo de execução dos métodos de Newton e Halley, para vinte testes com ponto inicial aleatório. Além disso, resolvemos também os problemas com o ponto inicial fornecido na respectiva coleção. Os resultados obtidos são organizados em gráficos e tabelas para uma melhor visualização e comparação dos dados.

Nossas conclusões estão resumidas ao final. A implementação do algoritmo de Halley

teve performance próxima à de Newton, sendo melhor em algumas dimensões, em alguns problemas, e pior em outras. Lembrando que os testes foram realizados no ambiente de computação MATLAB, sem grandes sofisticações no que tange à programação, os resultados levam a supor que implementações mais eficientes em uma linguagem como C.

Capítulo 1

Métodos para Zeros de Funções de Uma Variável

Resolver uma equação não-linear é um problema que tem ocupado matemáticos por séculos. Como veremos no Capítulo 4, o problema de localização de zeros de funções de uma variável

$$\text{encontrar } x^* \in \mathbb{R} \text{ tal que } f(x^*) = 0,$$

está intimamente relacionado com o problema de otimização (1).

Assumimos $f \in C^2$ para que os métodos de Newton e Halley, estudados neste capítulo, estejam bem definidos, embora para o método de Newton basta supor $f \in C^1$.

Consequentemente, o estudo de métodos para a localização de zeros de função é extremamente relevante em Matemática Aplicada. Neste trabalho focamos dois métodos específicos para tal, o método de Newton e o método de Halley. Nosso objetivo final é realizar uma comparação empírica de sua eficiência relativa.

Sendo assim, apresentamos neste capítulo alguns aspectos teóricos de maior relevância para o estudo destes métodos, na versão para uma variável, apresentando uma abordagem algébrica e outra geométrica, assim como resultados de convergência para o caso de função real de uma variável real.

1.1 O método de Newton para zeros de função

O método de Newton, ou Newton-Raphson, é o mais clássico método para a resolução de uma equação algébrica não-linear. Foi proposto por Newton em 1669 e depois por Raphson em 1690 [37] e, mesmo depois de mais de três séculos, o método ainda atrai a atenção de diversos pesquisadores.

O método de Newton para localização de zeros de funções de uma variável é um método iterativo que, dado um ponto que é uma aproximação para o zero, resolve o problema de achar o zero da aproximação linear da função no ponto dado, obtendo assim uma nova aproximação. A vantagem da escolha do tipo de aproximação reside no fato que o problema de localizar o zero de uma função linear é trivial.

Algebricamente, se $f\colon \mathbb{R} \to \mathbb{R}$ é a função cujo zero desejamos obter, sua aproximação linear no ponto $\bar{x}$ é justamente o polinômio de Taylor de primeiro grau desta função em torno deste ponto:

$$\ell(x) = f(\bar{x}) + f'(\bar{x})(x - \bar{x}). \tag{1.1}$$

A simplicidade da função permite obter uma expressão analítica para o zero de $\ell(x)$:

$$x = \bar{x} - \frac{f(\bar{x})}{f'(\bar{x})}. \tag{1.2}$$

Note que a fórmula acima pressupõe que a derivada em $\bar{x}$ seja não nula.

Assim, o método de Newton parte de uma estimativa x_0 para o zero de $f(x)$ e constrói uma sequência de aproximações sucessivas pela fórmula

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \qquad k = 0, 1, \ldots \tag{1.3}$$

Sob o ponto de vista de custo computacional, o método necessita, a cada iteração, uma avaliação da função e de sua derivada de primeira ordem. Mostraremos na próxima seção que, se a sequência gerada pelo método de Newton convergir para o zero da função f e a derivada de f for não nula neste ponto, a convergência será quadrática.

Sob o ponto de vista geométrico, o gráfico da aproximação linear de f no ponto $\bar{x}$ é uma reta tangente ao gráfico de f no ponto $(\bar{x}, f(\bar{x}))$. Ou seja, dado que a equação da reta tangente ao gráfico no ponto atual é

$$y = f(\bar{x}) + f'(\bar{x})(x - \bar{x}),$$

a próxima aproximação, a abscissa do ponto onde a reta tangente intersecta o eixo horizontal, é dada pelo valor de x que satisfaz $y = 0$:

$$0 = f(\bar{x}) + f'(\bar{x})(x - \bar{x}).$$

A solução da equação acima é precisamente a fórmula (1.2), obtida via a interpretação algébrica do método.

Na Figura 1.1 apresentamos duas iterações do método de Newton para quatro funções. Para cada função tomamos uma aproximação inicial x_0 e a reta tangente ao gráfico no ponto $(x_0, f(x_0))$, $r_0(x)$. A próxima aproximação da solução é x_1, abscissa do ponto onde r_0 intercecta o eixo x. No próximo passo, tomamos a reta tangente ao gráfico em $(x_1, f(x_1))$, $r_1(x)$. Podemos perceber que, nos quatro exemplos, a sequência de pontos gerada pelo processo iterativo do método de Newton parece estar convergindo para um zero da função.

Um pseudo-código do método de Newton é apresentado no Algoritmo 1. Os dados de entrada são a função, uma estimativa inicial para seu zero, uma tolerância positiva ϵ utilizada para critério de parada, e um valor utilizado como salvaguarda: *maxit*, o número máximo de iterações. A variável lógica *Sucesso* indica o resultado da execução do algoritmo. Se seu valor é *Verdade*, o valor de x^* é a aproximação encontrada para o zero da função de entrada, dentro da tolerância especificada. Caso contrário, o algoritmo parou sem ter atingido uma aproximação satisfatória.

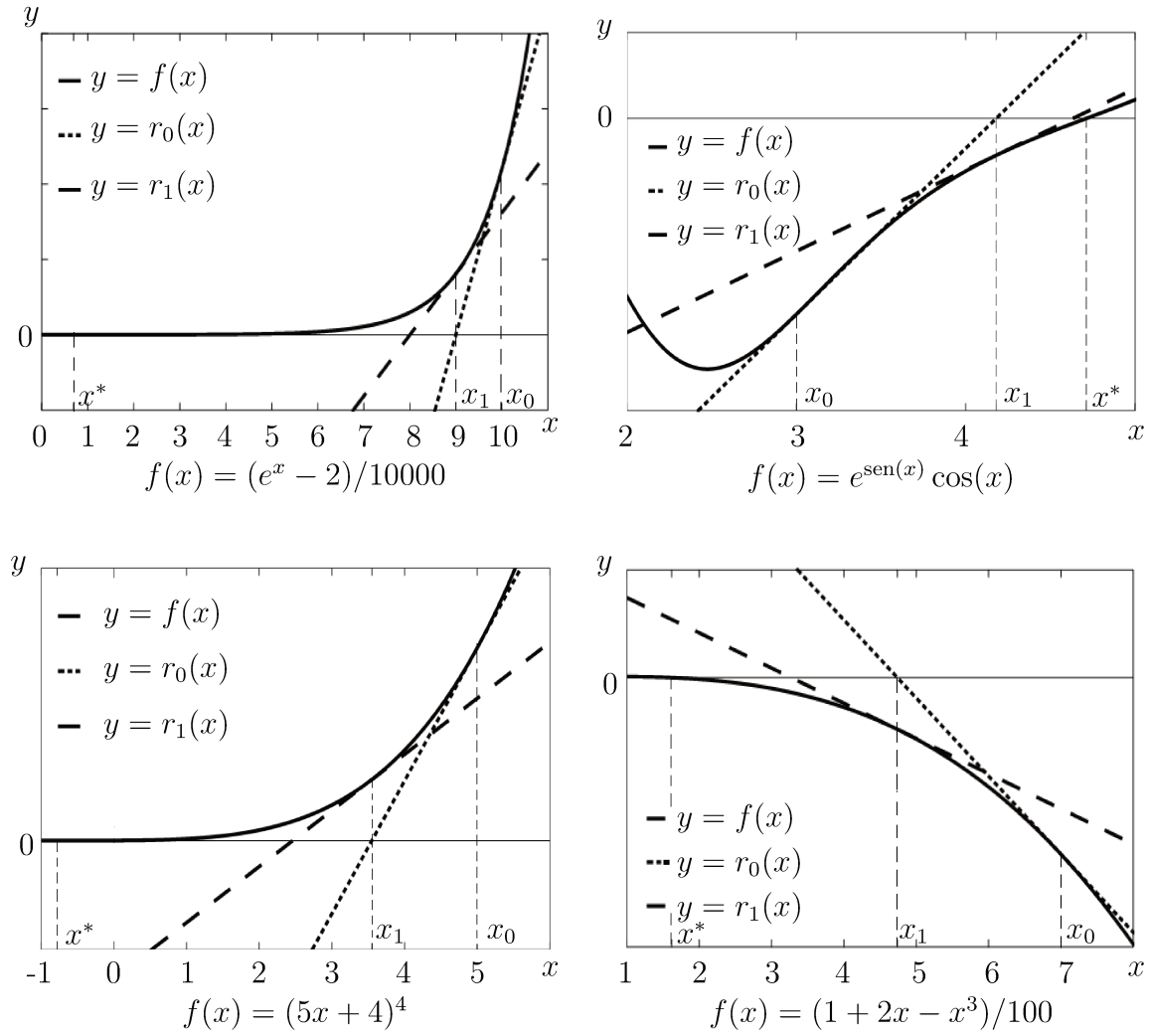


Figura 1.1: Processo iterativo do método de Newton, onde $r_0(x)$ e $r_1(x)$ são a primeira e a segunda aproximação do método de Newton respectivamente.

Algoritmo 1: Método de Newton para zero de função

Entrada:

f (função real de uma variável real), $x_0 \in \mathbb{R}$ (ponto inicial), $\epsilon > 0$ (tolerância), *maxit* (número máximo de iterações).

Inicialização:

$x^* \leftarrow 0$,

Sucesso$\leftarrow$*Verdadeiro*, $k \leftarrow 0$.

Enquanto $|f(x_k)| > \epsilon$, $k \leq$*maxit* e *Sucesso*=*Verdadeiro*

Se $f'(x_k) \neq 0$ **Então**

$$x_{k+1} \leftarrow x_k - \frac{f(x_k)}{f'(x_k)}$$

$k \leftarrow k + 1$

Caso contrário

Sucesso$\leftarrow$*Falso*

Se $k <$*maxit* e *Sucesso*=*Verdadeiro* **Então**

$x^* \leftarrow x_k$

Caso contrário

Sucesso$\leftarrow$*Falso*

Saída:

Sucesso, x^*.

1.1.1 Convergência do método de Newton para zeros de função

Ao trabalharmos com métodos iterativos, é importante conhecermos as condições de convergência do método e sua ordem de convergência, assumindo que o método converge.

Em geral, não se pode garantir convergência global do método de Newton e por isso focamos o estudo da taxa de convergência do método para os casos onde ele converge. Mesmo assim, é interessante conhecermos resultados que abordem este aspecto.

O teorema abaixo, por exemplo, apresenta condições suficientes para a convergência do método de Newton. Trata-se de um resultado de importância mormente teórica, pois na prática normalmente desconhecemos a vizinhança do zero procurado e o método eventualmente converge mesmo para funções que não satisfazem todas as hipóteses.

Teorema 1.1 (Convergência local do método de Newton) *Seja $f \in C^2[a,b]$. Se $x^* \in [a,b]$ é tal que $f(x^*) = 0$ e $f'(x^*) \neq 0$, então existe um $\delta > 0$ tal que o método de Newton gera uma sequência $\{x_k\}$ convergente para x^* para qualquer aproximação inicial $x_0 \in [x^* - \delta, x^* + \delta]$.*

A demonstração do Teorema 1.1 pode ser encontrada em [9] e é baseada na análise do método de Newton como sendo um caso particular do método de iteração do ponto fixo, cuja fórmula iterativa é $x_k = g(x_{k-1})$, $k \geq 1$, para

$$g(x) = x - \frac{f(x)}{f'(x)}. \tag{1.4}$$

Apresentamos abaixo a definição de ordem de convergência e, em seguida, o Teorema que garante a convergência quadrática do método de Newton, que é o que mais nos interessa nesta dissertação.

Definição 1.1 *Suponhamos que $\{p_k\}$ seja uma sequência que converge para p, com $p_k \neq p$ para todo k. Se existem as constantes positivas λ e α com*

$$\lim_{k\to\infty} \frac{|p_{k+1} - p|}{|p_k - p|^\alpha} = \lambda,$$

então $\{p_k\}$ comverge para p com ordem α e erro assintótico λ.

Teorema 1.2 (Ordem de convergência do método de Newton) *Assuma que o método de Newton aplicado à função $f \in C^2$ produz uma sequência $\{x_k\}$ que converge para ξ tal que $f(\xi) = 0$ e $f'(\xi) \neq 0$. Então esta convergência é quadrática.*

Prova [11]: Vamos considerar a função de iteração de ponto fixo do método de Newton

$$x_{k+1} = g(x_k) \tag{1.5}$$

onde g é dada em (1.4).

Por hipótese, o método de Newton é convergente e, se ξ é tal que $f(\xi) = 0$, ξ é um ponto fixo de g, ou seja,

$$\xi = g(\xi). \tag{1.6}$$

Subtraindo (1.6) de (1.5), temos

$$x_{k+1} - \xi = g(x_k) - g(\xi),$$

e utilizando a expansão de Taylor de g em torno de ξ para expressar $g(x_k)$, obtemos

$$x_{k+1} - \xi = g(\xi) + g'(\xi)(x_k - \xi) + \frac{g''(\psi_k)}{2!}(x_k - \xi)^2 - g(\xi), \tag{1.7}$$

onde ψ_k é um valor entre x_k e ξ.

Como

$$g'(x) = \frac{f(x)f''(x)}{[f'(x)]^2},$$

temos que $g'(\xi) = 0$. Utilizando este fato em (1.7), temos

$$x_{k+1} - \xi = \frac{g''(\psi_k)}{2!}(x_k - \xi)^2,$$

e, portanto,

$$\frac{|x_{k+1} - \xi|}{|x_k - \xi|^2} = \left|\frac{g''(\psi_k)}{2!}\right|.$$

Tomando o limite na igualdade acima,

$$\lim_{k\to\infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|^2} = \lim_{k\to\infty} \left|\frac{g''(\psi_k)}{2!}\right|,$$

Por hipótese, $x_k \to \xi$ e ψ_k está entre x_k e ξ. Portanto, pelo Teorema do Confronto,

$$\lim_{k\to\infty} \psi_k = \xi.$$

Como g'' é uma função contínua, temos então que

$$\lim_{k\to\infty} \left| \frac{g''(\psi_k)}{2!} \right| = \left| \frac{g''(\xi)}{2!} \right|$$

e podemos concluir que o método de Newton converge quadraticamente. □

1.2 O método de Halley para zero de função

O método de Halley para zero de função foi proposto por Edmond Halley em 1694, à mesma época que o método de Newton. Praticamente não é utilizado na atualidade mesmo sendo um método com convergência cúbica, provavelmente pelo fato de necessitar do cálculo das derivadas de segunda ordem, o que representava um alto custo para a época em que ele foi proposto. Para funções de uma variável isso já não se justifica mais e mesmo para funções de várias variáveis já há estudos indicando sua competitividade em casos especiais. Edmond Halley (1656-1742) foi um brilhante astrônomo e matemático, famoso por ter calculado a órbita do cometa que recebe o seu nome [38].

O método de Halley, juntamente com o método da secante, é considerado o método mais frequentemente redescoberto na literatura [31]. Por este motivo, encontramos diversas deduções para o método, algumas delas podem ser vistas em [2, 8, 31, 14, 1, 37]. Assim como o método de Newton, Halley constrói uma sequência de aproximações sucessivas para o zero da função original f resolvendo, a cada passo, um problema que modela o original. Não se trata, entretanto, do que poderia ser considerada a generalização natural do método de Newton, na qual o problema aproximado tomaria o polinômio de Taylor de segunda ordem da função original no ponto atual. Este método é chamado de método de Euler. No método de Halley, ver [31], o problema aproximado consiste em achar o zero de uma função h cujo gráfico é uma hipérbole osculante ao gráfico de f no ponto atual, motivando a alcunha atribuída ao método, de *método das hipérboles tangentes*. Ou seja, se x_k é o ponto atual, a função cujo zero seria encontrado na $(k+1)$-ésima iteração é

$$h_k(x) = \frac{(x - x_k) + c}{a(x - x_k) + b}, \tag{1.8}$$

onde as constantes[1] a, b e c devem ser tais que h_k tenha ordem de contato igual a 2 com a função f no ponto x_k. Ou seja, satisfazem as equações

$$h_k^{(i)}(x_k) = f^{(i)}(x_k), \quad \text{para } i = 0, 1, 2. \tag{1.9}$$

[1]Estas constantes também dependem da iteração, a rigor deveriam ser a_k, b_k e c_k. Eliminamos o subescrito para simplificar a notação.

Substituindo a expressão de $h_k(x)$ em (1.9), obtemos o seguinte sistema linear:

$$\begin{cases} f(x_k) &= \dfrac{c}{b}, \\ f'(x_k) &= \dfrac{b-ac}{b^2}, \\ f''(x_k) &= \dfrac{2a(ac-b)}{b^3}, \end{cases}$$

cuja solução é dada por:

$$\begin{cases} a &= \dfrac{-f''(x_k)}{2f'(x_k)^2 - f(x_k)f''(x_k)}, \\ b &= \dfrac{2f'(x_k)}{2f'(x_k)^2 - f(x_k)f''(x_k)}, \\ c &= \dfrac{2f(x_k)f'(x_k)}{2f'(x_k)^2 - f(x_k)f''(x_k)}. \end{cases}$$

Note que, para que a iteração do método de Halley esteja bem definida, o denominador que aparece nas expressões das constantes a, b e c deve ser não nulo.

A escolha de aproximações para a função original costuma ter em vista a obtenção de um problema de fácil solução. Neste caso, o zero de $h_k(x)$ pode ser obtido analiticamente: $x = x_k - c$. Substituindo o valor de c determinado no sistema linear acima, temos a fórmula iterativa do método de Halley

$$x_{k+1} = x_k - \frac{2f(x_k)f'(x_k)}{2f'(x_k)^2 - f(x_k)f''(x_k)}, \qquad k = 0, 1, \ldots \tag{1.10}$$

Note que, se $f''(x_k) = 0$, a fórmula iterativa de Halley coincide com a fórmula de Newton.

Na Figura 1.2 apresentamos duas iterações do método de Halley para quatro funções. Para cada função tomamos uma aproximação inicial x_0 e a função cujo gráfico é a hipérbole tangente ao gráfico nesse ponto, $h_0(x)$. O próximo ponto do processo iterativo é o ponto x_1, onde h_0 intercepta o eixo das abscissas, e tomamos então a função cujo gráfico é a hipérbole tangente ao gráfico em x_1, $h_1(x)$. Podemos perceber que, nos quatro exemplos, o sequência de pontos gerada pelo processo iterativo do método de Halley está convergindo para um zero da função.

Outra maneira de pensar o método de Halley [31, 14, 1] é trabalhar inicialmente com o polinômio de Taylor de segundo grau e aplicar uma correção de Newton para obter a aproximação. Seja p_k o polinômio de Taylor de segundo grau que aproxima f em torno de x_k:

$$p_k(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{f''(x_k)}{2}(x - x_k)^2. \tag{1.11}$$

Analogamente ao método de Newton, suponha que desejamos obter o zero de $p_k(x)$. Com algumas manipulações algébricas chegamos em

$$(x - x_k)\left(f'(x_k) + \frac{f''(x_k)}{2}(x - x_k)\right) = -f(x_k),$$

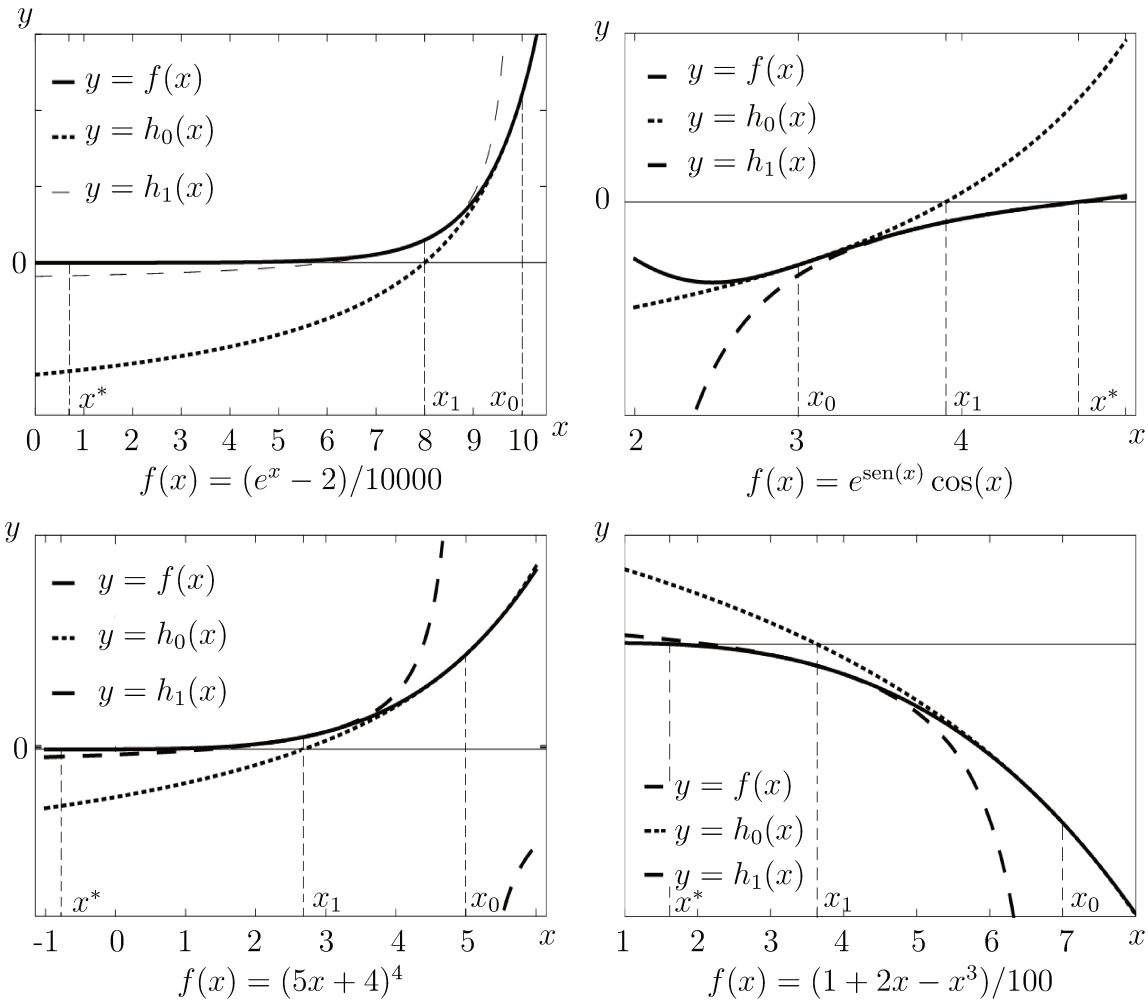


Figura 1.2: Processo iterativo do método de Halley, onde $h_0(x)$ e $h_1(x)$ são a primeira e a segunda aproximação do método de Halley respectivamente.

que pode ser reescrito como

$$x - x_k = -\frac{f(x_k)}{f'(x_k) + \frac{1}{2}f''(x_k)(x - x_k)}. \tag{1.12}$$

Aplicando uma correção de Newton

$$(x - x_k) = -\frac{f(x_k)}{f'(x_k)}$$

no lado direito da equação (1.12), obtemos:

$$\begin{aligned} x - x_k &= -\frac{f(x_k)}{f'(x_k) - \frac{f''(x_k)f(x_k)}{2f'(x_k)}} \\ &= \frac{-2f(x_k)f'(x_k)}{2f'(x_k)^2 - f''(x_k)f(x_k)}, \end{aligned}$$

e portanto, temos a fórmula iterativa do método de Halley para localização de zeros de função.

No Algoritmo 2 apresentamos o pseudo-código para o método de Halley. As entradas são a função f, a estimativa inicial x_0 para o zero de f, a tolerância positiva ϵ para o critério de parada (valor de função suficientemente pequeno). Finalmente *maxit* determina o número máximo de iterações, fornecendo uma salvaguarda para o caso do método não convergir. A saída é constituída pela variável lógica *Sucesso* e pelo valor x^*. Se *Sucesso* é *Verdadeiro*, então x^* contém a aproximação encontrada para o zero de f pelo método. Caso contrário, sabemos que o método fracassou.

Algoritmo 2: Método de Halley para zero de função

Entrada:

f (função real de uma variável real), $x_0 \in \mathbb{R}$ (ponto inicial), $\epsilon > 0$ (tolerância), *maxit* (número máximo de iterações).

Inicialização:

$x^* \leftarrow 0$,

Sucesso$\leftarrow$*Verdadeiro*, $k \leftarrow 0$.

Enquanto $|f(x_k)| > \epsilon$, $k \leq$*maxit* e *Sucesso*=*Verdadeiro*

Se $(2f'(x_k)^2 - f''(x_k)f(x_k)) \neq 0$

$x_{k+1} \leftarrow x_k - \dfrac{2f(x_k)f'(x_k)}{2f'(x_k)^2 - f''(x_k)f(x_k)}$

$k \leftarrow k + 1$

Caso contrário

$Sucesso \leftarrow Falso$

Se $k <$*maxit* e *Sucesso*=*Verdadeiro* **Então**

$x^* \leftarrow x_k$

Caso contrário

Sucesso$\leftarrow$*Falso*

Saída:

Sucesso, x^*.

1.2.1 Convergência do método de Halley

Sob determinadas condições, pode-se mostrar a convergência monótona do método de Halley. O resultado a seguir é extraído de [27].

Teorema 1.3 (Convergência do método de Halley) *Seja f''' contínua, $f' \neq 0$ e $((\operatorname{sgn}(f')f')^{-\frac{1}{2}})'' \geq 0$ num intervalo I contendo o zero x^* de f. Então o método de Halley converge monotonicamente para x^* partindo de qualquer ponto em I.*

Prova: Se a função contínua f' é não-nula em I, então seu sinal é constante neste intervalo. Sem perda de generalidade vamos supor $f' > 0$, o caso restante tem demonstração análoga.

Para obter a convergência monótona, basta mostrar que, se começamos em $\bar{x} \in I$, então o próximo ponto de Halley situa-se entre $\bar{x}$ e x^*. Dividimos a demonstração em dois casos.

Caso 1: $f''(\bar{x}) \neq 0$.

Neste caso a função racional h em (1.8) que aproxima f em torno de $\bar{x}$ em uma iteração genérica do método de Halley pode ser reescrita como

$$h(x) = A + \frac{B}{x + C}, \tag{1.13}$$

onde as constantes A, B e C devem satisfazer as condições

$$A + \frac{B}{\bar{x} + C} = f(\bar{x}), \tag{1.14}$$

$$-\frac{B}{(\bar{x} + C)^2} = f'(\bar{x}), \tag{1.15}$$

$$\frac{2B}{(\bar{x} + C)^3} = f''(\bar{x}). \tag{1.16}$$

para que h tenha ordem de contato 2 com f em $x = \bar{x}$.

De (1.15) segue que

$$B < 0. \tag{1.17}$$

Por outro lado, de (1.15) e (1.16) temos

$$\operatorname{sgn}(\bar{x} + C) = -\operatorname{sgn}(f''(\bar{x})). \tag{1.18}$$

O domínio de h é $(-\infty, -C) \cup (-C, \infty)$. Da condição (1.18) acima, temos que o intervalo deste domínio que contém $\bar{x}$ é determinado pelo sinal de $f''(\bar{x})$. Se $f''(\bar{x}) > 0$, $\bar{x}$ encontra-se à esquerda de $-C$, e se $f''(\bar{x}) < 0$, então $\bar{x} > -C$. Veremos que o gráfico de h encontra-se abaixo do gráfico de f para pontos no domínio à esquerda de $\bar{x}$ e acima para pontos à direita.

Para tanto, considere

$$g(x) = \frac{-\operatorname{sgn}(f''(\bar{x}))}{\sqrt{-B}}(x + C). \tag{1.19}$$

De (1.17) segue que g está bem definida. Esta função linear é o polinômio de Taylor que aproxima $f'^{-1/2}(x)$ até a segunda ordem, em torno do ponto $x = \bar{x}$.

A hipótese que $(f'^{-1/2})'' \geq 0$ implica que $f'^{-1/2}$ é convexa em I. Portanto $g(x) \leq f'^{-1/2}(x)$. A função linear g se anula em $x = -C$, tem sinal $\operatorname{sgn}(f''(\bar{x}))$ no intervalo $(-\infty, -C)$ e sinal $-\operatorname{sgn}(f''(\bar{x}))$ em $(-C, \infty)$. Portanto, destes dois, o intervalo que contém $\bar{x}$ é aquele no qual g é positiva. Então, neste intervalo,

$$0 < \frac{-\operatorname{sgn}(f''(\bar{x}))}{\sqrt{-B}}(x + C) \leq f'^{-\frac{1}{2}},$$

o que implica

$$-\frac{B}{(x+C)^2} \geq f'(x). \tag{1.20}$$

Assim, para $x \geq \bar{x}$,

$$\int_{\bar{x}}^{x} \frac{-B}{(t+C)^2} dt \geq \int_{\bar{x}}^{x} f'(t)dt,$$

e, portanto,

$$A + \frac{B}{x+C} - A - \frac{B}{\bar{x}+C} \geq f(x) - f(\bar{x}),$$

o que equivale a dizer, usando (1.14)

$$h(x) \geq f(x), \quad \text{para } x \geq \bar{x}. \tag{1.21}$$

Para $x \leq \bar{x}$,

$$\int_{x}^{\bar{x}} \frac{-B}{(t+C)^2} dt \geq \int_{x}^{\bar{x}} f'(t)dt,$$

e, portanto,

$$A + \frac{B}{\bar{x}+C} - A - \frac{B}{x+C} \geq f(\bar{x}) - f(x).$$

Temos então, analogamente,

$$h(x) \leq f(x), \quad \text{para } x \leq \bar{x}. \tag{1.22}$$

Concluímos então o resultado desejado, isto é, se $x^* > \bar{x}$, então $f(\bar{x}) < 0$ e o gráfico de h corta o eixo (no próximo iterando $\tilde{x}$ do método de Halley) antes do gráfico de f, se percorremos o gráfico da esquerda para a direita. Ou seja, $x^* \leq \tilde{x} \leq \bar{x}$. Analogamente, se $\bar{x} > x^*$, como $f(\bar{x}) < 0$, então $\bar{x} \leq \tilde{x} \leq x^*$.

Caso 2: $f''(\bar{x}) = 0$.

Neste caso a função racional h de (1.8) toma a forma

$$h(x) = Ax + B, \tag{1.23}$$

ou seja, o gráfico da aproximação h é uma reta tangente ao gráfico de f em $x = \bar{x}$. Note que neste caso a aproximação h coincide com a aproximação do método de Newton.

Como h deve satisfazer

$$h'(\bar{x}) = f'(\bar{x}),$$

temos que $A = f'(\bar{x})$.

Assim, se $f' > 0$, temos que $A > 0$ e $g(x) = A'^{-1/2}$ aproxima $f''^{-1/2}$ até segunda ordem em torno de $x = \bar{x}$. Portanto, da convexidade de $f''^{-1/2}$, seguem as mesmas conclusões. □

Mais relevante para o nosso estudo é o resultado que garante a taxa de convergência cúbica do método de Halley, que apresentamos a seguir, cuja demonstração segue os moldes da demonstração da ordem de convergência do método de Newton.

Teorema 1.4 (Ordem de convergência do método de Halley) *Assuma que o método de Halley produz uma sequência $\{x_k\}$ que converge para ξ tal que $f(\xi) = 0$ e $f'(\xi) \neq 0$, para $f \in C^3$. Então a convergência da sequência é cúbica.*

Prova: Vamos considerar a função de iteração de ponto fixo do método de Halley

$$x_{k+1} = g(x_k) \tag{1.24}$$

onde g é dada em por

$$g(x) = x - \frac{2f(x)f'(x)}{2f'(x) - f(x)f''(x)}. \tag{1.25}$$

Por hipótese, o método de Halley é convergente e, se ξ é tal que $f(\xi) = 0$, ξ é um ponto fixo de g, ou seja

$$\xi = g(\xi). \tag{1.26}$$

Subtraindo (1.26) de (1.24)

$$x_{k+1} - \xi = g(x_k) - g(\xi)$$

e utilizando o polinômio de Taylor de terceiro grau de g em torno de ξ, obtemos

$$x_{k+1} - \xi = g(\xi) + g'(\xi)(x_k - \xi) + \frac{g''(\xi)}{2!}(x_k - \xi)^2 + \frac{g'''(\psi_k)}{3!}(x_k - \xi)^3 - g(\xi), \tag{1.27}$$

onde ψ_k é um número entre x_k e ξ.

Note que

$$g'(\xi) = 0 \qquad \text{e} \qquad g''(\xi) = 0.$$

Utilizando os fatos acima em (1.27), temos

$$x_{k+1} - \xi = \frac{g'''(\psi_k)}{3!}(x_k - \xi)^3$$

e, portanto,

$$\frac{|x_{k+1} - \xi|}{|x_k - \xi|^3} = \left|\frac{g'''(\psi_k)}{3!}\right|.$$

O fato que x_k converge para ξ implica, pelo Teorema do Confronto, que ψ_k converge para ξ. Usando a continuidade de g''' temos que

$$\lim_{k\to\infty} \frac{|x_{k+1} - \xi|}{|x_k - \xi|^3} = \left|\frac{g'''(\xi)}{3!}\right|$$

e, portanto, podemos concluir que o método de Halley converge cubicamente. □

1.3 Testes Numéricos - Localização de Zeros de Funções de Uma Variável

Apesar do método de Halley possuir convergência cúbica e ter surgido à mesma época que o método de Newton ele praticamente não é utilizado na atualidade. Isso se deve ao fato do método ser aplicado para uma classe menor de funções, C^2, e necessitar do cálculo das derivadas de segunda ordem, o que leva a um custo computacional maior.

Na época em que o método foi criado, o maior custo computacional provavelmente justificava o fato dele não ser empregado para funções de uma variável, mas na atualidade isto não se aplica mais. O método de Halley a princípio necessitaria de um número menor de iterações para convergir, o que pode compensar o maior custo por iteração.

Implementamos os métodos de Newton e Halley no MATLAB, versão 7.10 (R2010a), para podermos investigar sua eficiência relativa. As funções foram retiradas de livros de cálculo [33, 15], uma vez que não encontramos bibliotecas de função teste para funções de uma variável.

Foram selecionadas 30 funções de maneira aleatória, dentre as funções da classe C^2. Tanto o método de Newton quanto o método de Halley possuem convegência global assegurada apenas em alguns casos particulares. Não buscamos selecionar funções para as quais os métodos possuíssem convergência global, mas sim funções com as quais obtivessemos convergência para um número razoável de testes, para pelo menos um dos métodos.

Para cada função selecionada, foram realizados 20 testes com ponto inicial aleatório, um número real entre -50 e 50. Os resultados que apresentamos são as médias aritméticas do número de iterações e tempo de execução dos testes em que obtivemos convergência. Nos testes em que a aproximação inicial levou à não convergência de algum dos métodos, os resultados apresentados por ambos os métodos, para este ponto inicial, foram desconsiderados na média.

Na Tabela 1.1 apresentamos a média aritmética do número de iterações e tempo de execução em segundos, obtidos para as 30 funções selecionadas. Também estão indicadas as funções que apresentaram algum tipo de falha, em qual método falhou e para quantos pontos iniciais. Consideramos como falha quando o método atingiu um número máximo de 200 iterações, apresentou o denominador da fórmula iterativa nulo ou um NaN (Not a Number).

Das 30 funções selecionadas, 8 apresentaram algum tipo de falha. O método de Newton não convergiu em 86 testes, distribuídos nas 8 funções, dos 600 realizados. Já o método de Halley apresentou falha em apenas 2 das 8 funções, num total de 8 testes, sendo 2 falhas para a função f_{13}, na qual Newton apresentou 11 falhas, e 6 falhas para a função f_{26}, contra 8 do método de Newton para mesma função. É importante ressaltar que, apesar do ponto inicial ser aleatório, foi utilizado o mesmo conjunto de pontos para ambos os métodos.

Na Figura 1.3 apresentamos uma representação gráfica das 8 funções que apresentaram

Função	Método de Halley			Método de Newton		
	iterações	tempo(s)	falhas	iterações	tempo(s)	falhas
$f_1(x) = e^x \cos^2(x)$	16	0.486	0	25	0.482	1
$f_2(x) = (5x+4)^4$	19	0.464	0	34	0.559	0
$f_3(x) = e{-}x^2 - x^2 + x$	5	0.144	0	7	0.150	0
$f_4(x) = x^2 + \operatorname{sen}(x)$	6	0.141	0	9	0.158	0
$f_5(x) = 2\cos(x) + \sec(x)^2$	6	0.187	0	10	0.213	0
$f_6(x) = e^x + 20\,\mathrm{tg}^{-1}(x)$	13	0.4041	0	27	0.544	8
$f_7(x) = e^{\operatorname{sen}(x)}\cos(x)$	4	0.112	0	4	0.091	0
$f_8(x) = x^3 - 4x + 1$	6	0.159	0	10	0.198	0
$f_9(x) = \sqrt[3]{x^4}$	9	0.261	0	13	0.219	0
$f_{10}(x) = 1 - x^4$	8	0.184	0	13	0.216	0
$f_{11}(x) = (4x+1)^5(3 - x^2 + x)^8$	42	2.597	0	81	2.314	0
$f_{12}(x) = x^2\sqrt{2 + x + x^2} - 1$	7	0.335	0	11	0.296	0
$f_{13}(x) = (48x-1)(1+x)^{60} + 1$	92	6.205	2	141	6.048	11
$f_{14}(x) = x^2\operatorname{sen}(1/x)$	4	0.119	0	4	0.088	0
$f_{15}(x) = 5x^3 - 8$	9	0.220	0	15	0.264	0
$f_{16}(x) = ((3 - x^3)^4 - 16)/(x^3 - 1)$	23	1.277	0	34	0.933	0
$f_{17}(x) = \sqrt[3]{x^2 - 9}$	53	1.684	0	–	–	20
$f_{18}(x) = 3x^2/(\mathrm{tg}(x)\operatorname{sen}(x))$	4	0.226	0	5	0.144	0
$f_{19}(x) = 1 - 2^x$	9	0.211	0	29	0.482	9
$f_{20}(x) = (3x^3 + 1)e^x$	13	0.463	0	22	0.513	0
$f_{21}(x) = 6xe^{1/x} + (3x^2 + 1)e^x$	12	0.495	0	18	0.495	0
$f_{22}(x) = (x+1)\cos(x) - \operatorname{sen}(x)/(x+1)^2$	4	0.149	0	4	0.126	0
$f_{23}(x) = x^3 + \ln(\lvert x\rvert)$	10	0.320	0	16	0.325	0
$f_{24}(x) = 6x^3 + \sqrt[6]{x^2}$	8	0.265	0	13	0.285	9
$f_{25}(x) = (x^4 + \sqrt[8]{x^2})/(x^2 + 3)$	145	9.133	0	–	–	20
$f_{26}(x) = (x^3 + x^5)/\operatorname{sen}(x)$	41	1.930	6	47	1.159	8
$f_{27}(x) = \cos(x)^4/(x^2 + 1)$	6	0.304	0	7	0.211	0
$f_{28}(x) = 4 + 5x\ln(\lvert x\rvert)$	5	0.157	0	7	0.154	0
$f_{29}(x) = (x^2 + 3x)^2$	15	0.860	0	24	0.933	0
$f_{30}(x) = (5x^3 + 6x - 1)^8$	49	4.406	0	94	4.618	0

Tabela 1.1: Média de número de iterações e tempo (s) para 20 testes com ponto inicial aleatório.

falhas, com o número de falhas apresentado por cada método. Note que o método de Halley é o mais robusto neste conjunto de funções, pois, além de apresentar falha para um número reduzido de funções, o número de falhas para cada função é menor do que o do método de Newton.

É importante destacar que, para as funções f_{17} e f_{25}, o método de Newton não converge, qualquer que seja a aproximação inicial, enquanto que o método de Halley sempre converge. No caso de f_{17} o valor de sua derivada não está definida no zero da função. Já

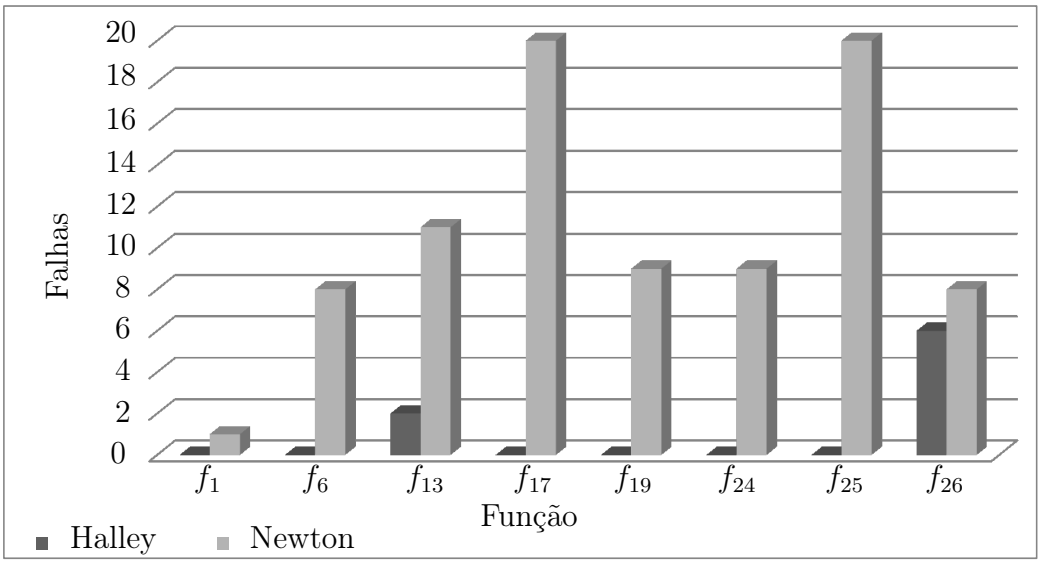


Figura 1.3: Funções que apresentaram falha na execução do algoritmo do método de Newton e do método de Halley, em 20 testes com ponto inicial aleatório.

a derivada de f_{25} se anula no zero, não satisfazendo, assim, as condições de convergência tanto do método de Newton quanto do método de Halley. Este exemplo sugere uma maior robustez do método de Halley.

Na Figura 1.4 apresentamos geometricamente três iterações do método de Newton para a função f_{17}. Tomamos $x_0 = 3.1$ como ponto inicial e tomamos a aproximação linear da função nesse ponto, $\ell_0(x)$, cujo zero é o próximo ponto do processo iterativo. ℓ_1 e ℓ_2 representam a segunda e a terceira aproximação linear da função, respectivamente. Observando a figura, fica claro que a cada iteração a sequência se afasta mais da solução e o método nunca vai convergir.

Para um estudo mais abrangente dos resultados apresentados na Tabela 1.1 separamos nosso conjunto de funções teste em dois grupos:

- testes com falha - aqueles que apresentaram algum tipo de falha no método de Halley e/ou no método de Newton;

- testes com êxito - os que convergiram, nos dois métodos, para todos os pontos iniciais.

Na Figura 1.5 apresentamos a média aritmética do número de iterações necessárias para a convergência do método de Newton e Halley. Note que, as f_{17} e f_{25} não aparecem no gráfico, uma vez que não temos nenhuma informação do número de iterações para o método de Newton, que não converge para estas funções.

Em todas as funções que apresentaram algum tipo de falha o método de Halley é o que apresenta a menor média do número de iterações. A diferença entre os números de iterações apresentados pelos métodos de Newton e Halley é bem significativa para a

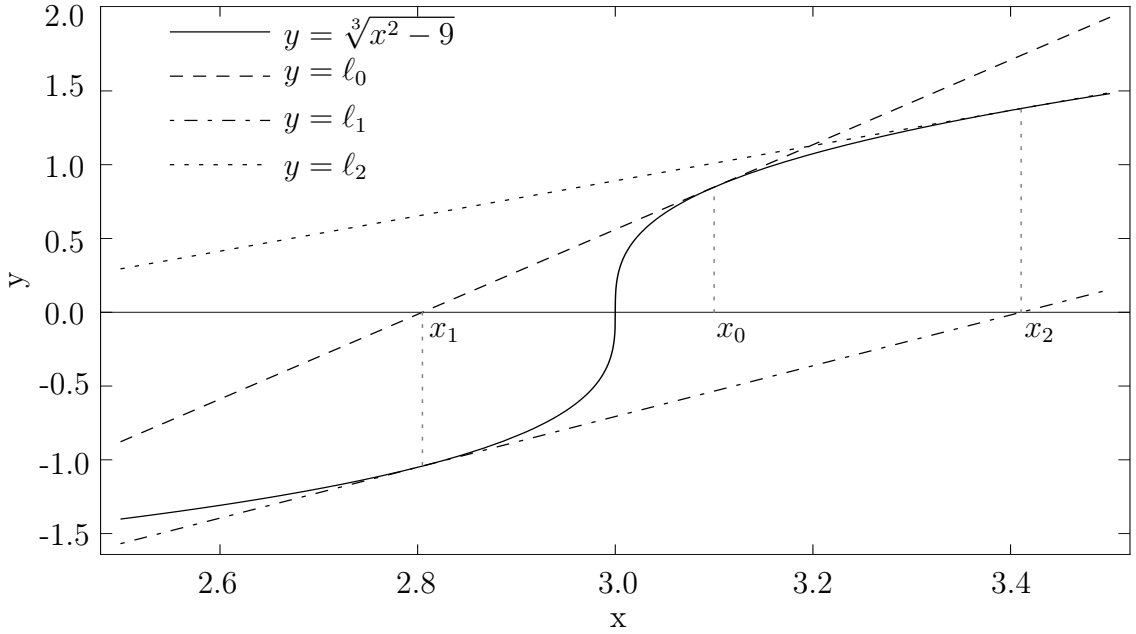


Figura 1.4: Três iterações do método de Newton para a função f_{17} com $x_0 = 3.1$ que ilustra a não convergência do método.

maioria das funções e com isso esperamos que o método de Halley também obtenha o melhor tempo de execução.

Na Figura 1.6 podemos comparar as médias dos tempos de execução obtidos pelos métodos de Newton e Halley para as 6 funções que apresentaram falhas. Foram realizados 20 testes para cada função com ponto inicial aleatório. Para média consideramos apenas os testes com ponto inicial que levou a convergência de ambos os métodos.

Analisando a Figura 1.6 vemos que o método de Halley foi mais eficiente em 50% dos testes, sendo que para a função f_1 a diferença no tempo de execução está em milésimos de segundo. Apesar do método de Newton convergir mais rápido para 3 funções ele não seria o método mais recomendado para localizar os zeros destas funções uma vez que ele apresenta muitas falhas para estas funções.

Na Figura 1.7 apresentamos uma comparação da média aritmética dos números de iterações necessárias para a convergência dos métodos de Newton e Halley para as 22 funções que obtiveram convergência para todos os pontos iniciais testados.

Vemos que, das funções que apresentaram êxito em todos os testes, o método de Halley apresentou a menor média do número de iterações para 19 funções, cerca de 86% do total, sendo que nas outras três funções ambos os métodos convergiram com 4 iterações, o que é aceitável, uma vez que este número é bem pequeno.

Este resultado já era esperado, uma vez que a ordem de convergência do método de Halley é maior do que a do método de Newton. Assim, pudemos comprovar na prática este resultado teórico, no qual baseamos nossos estudos.

Na Figura 1.8 apresentamos a média aritmética do tempo de execução dos métodos de Newton e Halley para as 22 funções com as quais obtivemos êxito em todos os testes.

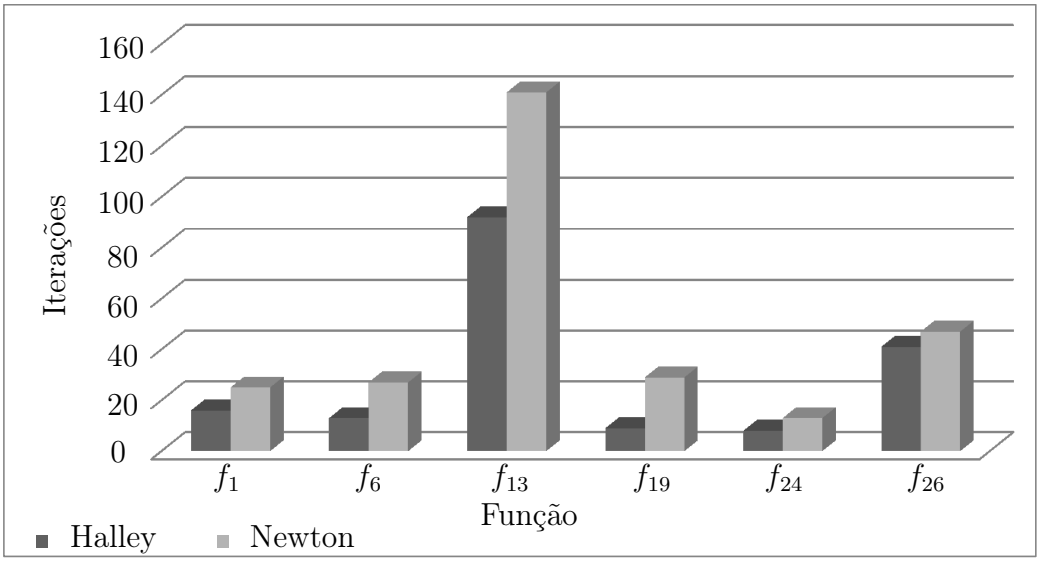


Figura 1.5: Média do número de iterações obtidos para os método de Newton e Halley para as funções que apresentaram falha. 20 testes para cada função com ponto inicial aleatório. A média considera apenas os resultados dos pontos iniciais que levaram à convergência de ambos os métodos.

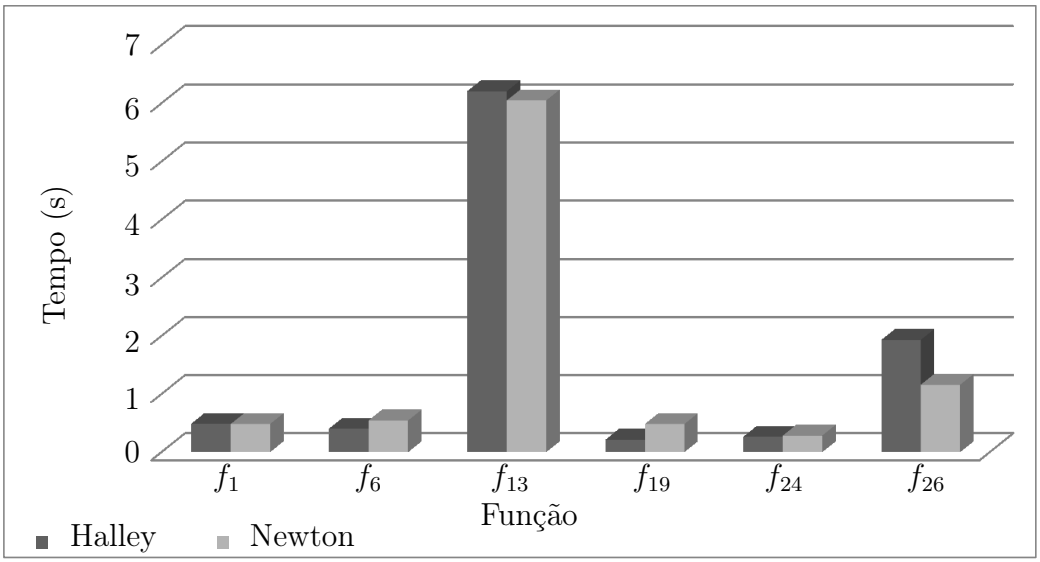


Figura 1.6: Média do tempo de execução do método de Newton e Halley para as funções que apresentaram falha. 20 testes para cada função com ponto inicial aleatório. A média considera apenas os resultados dos pontos iniciais que levaram à convergência de ambos os métodos.

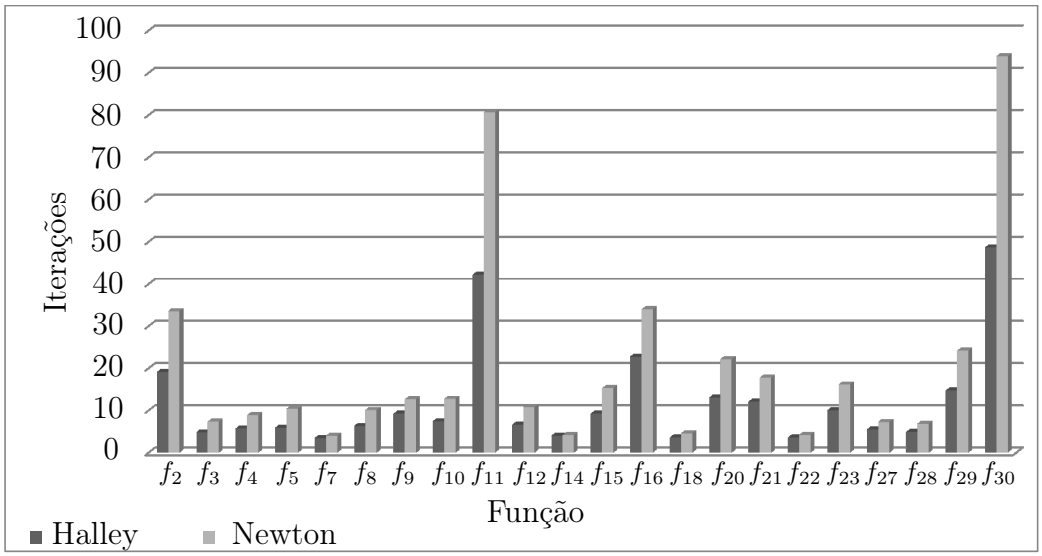


Figura 1.7: Média aritmética do número de iterações obtidos dos métodos de Newton e Halley para as funções que apresentaram êxito em todos os testes. 20 testes para cada função com ponto inicial aleatório.

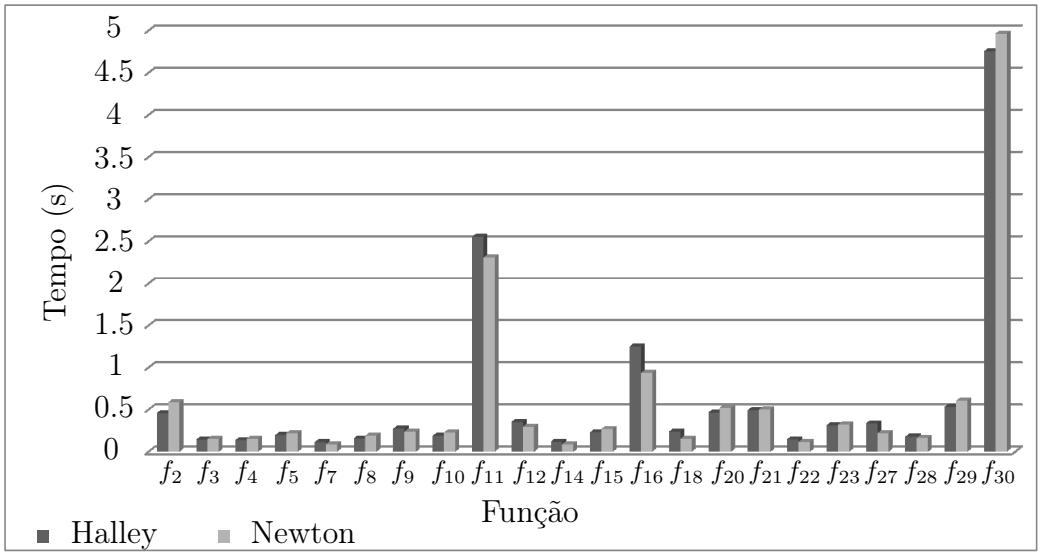


Figura 1.8: Média aritmética do tempo de execução do método de Newton e Halley para as funções que apresentaram êxito em todos os testes. 20 testes para cada função com ponto inicial aleatório.

Com base no gráfico apresentado na Figura 1.8 e na Tabela 1.1, vemos que, das 22 funções consideradas, o método de Halley apresentou o melhor tempo para 12 funções, cerca de 55% do total, contra 10 funções onde o método de Newton obteve o menor tempo. Dessas 10 funções, em 3 delas já era esperado um desempenho superior do método de Newton, uma vez que os dois métodos apresentaram o mesmo número de iterações.

Nosso principal objetivo com a realização destes testes era analisar até que ponto o maior custo computacional do método de Halley interfere no seu desempenho. Acreditamos que, a necessidade de um número menor de iterações para obter a convergência acabe compensando o maior custo por iteração. De modo geral, isto se verifica para mais da metade dos testes e esse resultado fica mais evidente quando consideramos apenas os testes em que o método de Halley apresenta uma média menor do número de iterações. Nesse caso, a superioridade do método de Halley se verifica em 63% dos testes.

Vale ressaltar que, na grande maioria dos testes, o resultado obtido pelo método de Halley é mais preciso do que o resultado apresentado pelo método de Newton, uma vez que ao avaliar a função no ponto encontrado pelo algoritmo de Halley obtemos um valor menor do que o obtido com a aproximação de Newton, e nem sempre os dois métodos convergem para o mesmo ponto.

De modo geral, pudemos constatar nos exemplos práticos que o método mais indicado para resolver o problema de zero de funções de uma variável é o método de Halley, pois é o método mais robusto e mais eficiente, se comparado ao método de Newton. Como veremos nos próximos capítulos, para funções vetoriais isso nem sempre é verdade, uma vez que o custo computacional cresce muito com o aumento da dimensão do problema. Por isso buscamos formas de tornar o algoritmo do método de Halley mais eficiente.

Capítulo 2

Métodos para Zeros de Funções Vetoriais

Neste capítulo utilizamos os resultados apresentados no Capítulo 1 para funções de uma variável e estendemos para o caso geral de localização de zeros de funções de várias variáveis.

2.1 Método de Newton

Seja $F : \mathbb{R}^n \to \mathbb{R}^n$ uma função vetorial, estamos interessados em resolver o problema de localizar os zeros de F, ou seja

$$\text{encontrar } x^* \in \mathbb{R}^n \text{ tal que } F(x^*) = 0.$$

Assumimos $F \in C^2$ para que os métodos de Newton e Halley estejam bem definidos, embora para o método de Newton basta supor $F \in C^1$.

Assim como para funções de uma variável, o método mais utilizado na atualidade é o método de Newton, cuja fórmula iterativa é dada por

$$x_{k+1} = x_k - [F'(x_k)]^{-1} F(x_k), \qquad k = 0, 1, \ldots \tag{2.1}$$

onde $F'(x_k) \in \mathbb{R}^{n \times n}$ é a matriz jacobiana da função F.

Em dimensão n genérico, não é possível fornecer uma visualização gráfica. No entanto, a abordagem algébrica é facilmente extensível [37].

Analogamente ao método de Newton para zeros de função de uma variável, a cada passo resolvemos um problema que aproxima o original no qual a função original é substituída pelo seu polinômio de Taylor de primeiro grau em torno do ponto atual.

Seja $F : \mathbb{R}^n \to \mathbb{R}^n$ e considere $L_k(x)$ sua aproximação linear em torno do ponto atual x_k:

$$L_k(x) = F(x_k) + F'(x_k)(x - x_k). \tag{2.2}$$

Caso a matriz $F'(x_k)$ seja não singular, o problema aproximado $L_k(x) = 0$ tem a solução

analítica

$$x = x_k - [F'(x_k)]^{-1}F(x_k),$$

que vai constituir a próxima aproximação da solução.

Assim temos a fórmula iterativa do método de Newton para localização de zeros de funções vetoriais (2.1). Na próxima seção apresentamos o resultado de convergência que garante a convergência quadrática da sequência gerada pelo método de Newton, assumindo que o método é convergente.

O Algoritmo 3 apresenta um pseudo-código do método de Newton. Os dados de entrada são a função cujo zero deseja-se encontrar, a estimativa inicial deste zero, a tolerância positiva ϵ utilizada no critério de parada (F suficientemente próxima de zero) e o número máximo de iterações, uma salvaguarda para o caso de não convergência. Se a variável lógica *Sucesso* vale *Verdadeiro* na saída do algoritmo, então o ponto x^* fornecido na saída é estimativa do zero de F produzida pelo método. Caso contrário, concluímos que o método não conseguiu convergir.

Algoritmo 3: Método de Newton para zeros de funções vetoriais

---

Entrada:

F (função real vetorial), $x_0 \in \mathbb{R}^n$ (ponto inicial), $\epsilon > 0$ (tolerância), *maxit* (número máximo de iterações),

Inicialização:

$x^* \leftarrow 0$,

Sucesso $\leftarrow$ *Verdadeiro*, $k \leftarrow 0$.

Enquanto $\|F(x_k)\| > \epsilon$, $k \leq$ *maxit* e *Sucesso* = *Verdadeiro*

  Se $F'(x_k)$ é não singular **Então**

    $x_{k+1} \leftarrow x_k - (F'(x_k))^{-1}F(x_k)$

    $k \leftarrow k + 1$

  Caso contrário

    Sucesso $\leftarrow$ *Falso*

Se $k <$*maxit* e *Sucesso* = *Verdadeiro* **Então**

  $x^* \leftarrow x_k$

Caso contrário

  Sucesso $\leftarrow$ *Falso*

Saída:

Sucesso, x^*.

2.1.1 Convergência do método de Newton

Como o método de Newton é um método muito utilizado na atualidade, existem diversos artigos na literatura que abordam a convergência do método. Em [37] Yamamoto apresenta um breve histórico de todo desenvolvimento da análise de convergência do método de Newton em espaços de Banach.

Em geral, não se pode garantir convergência global do método de Newton, e embora existam resultados que estipulem condições suficientes para a convergência do método, eles não costumam ser verificados em aplicações práticas.

Pensando nisso, apresentamos apenas o resultado de convergência local, que inclui a garantia da convergência quadrática do método de Newton, que é o resultado que mais nos interessa.

Teorema 2.1 *Seja $F : \mathbb{R}^n \to \mathbb{R}^n$. Suponha que $F(\xi) = 0$, que em alguma vizinhança aberta $N(\xi)$ de ξ onde F está definida e é contínua, todas as derivadas parciais de segunda ordem de f estão definidas e são contínuas, e que a matriz jacobiana $J_F(\xi)$ de F no ponto ξ é não singular. Então a sequência $\{x_k\}$ definida pelo método de Newton (2.1) converge para a solução ξ, desde que x_0 esteja suficientemente próximo de ξ. A convergência da sequência $\{x_k\}$ para ξ é pelo menos quadrática.*

Assim como no caso de uma variável, a prova desse Teorema formula o método de Newton como caso particular do método de ponto fixo e mostra que, com essas hipóteses, o método satisfaz as condições de convergência do método de ponto fixo. Já a prova da convergência quadrática utiliza a mesma abordagem que apresentamos no capítulo anterior, utilizando o polinômio de Taylor. A prova pode ser encontrada em [35].

2.2 Método de Halley

Na literatura encontramos diversas denominações que caracterizam o método de Halley, entre elas "método como Newton", definido em [13, 37], ou mesmo um método de dois passos de Newton, abordado em [16].

Para várias variáveis não temos mais uma interpretação do gênero "na iteração k resolvemos um problema aproximado, onde a função original é substituída pela função ...". No entanto, como notam Ortega e Rheinboldt [29], isto não impediu vários pesquisadores de generalizarem o método para $\mathbb{R}^n$ ou para um espaço de dimensão infinita. Em particular, o desenvolvimento utilizado para chegar na fórmula iterativa em uma variável, pode ser estendido para várias variáveis, como segue.

Seja $F : \mathbb{R}^n \to \mathbb{R}^n$, e suponha que queremos encontrar $x^* \in \mathbb{R}^n$ tal que $F(x^*) = 0$. Primeiramente, aproximamos a função pelo seu polinômio de Taylor de segunda ordem em x_k

$$P_k(x) = F(x_k) + F'(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^t F''(x_k)(x - x_k). \tag{2.3}$$

Fazendo $P_k(x) = 0$ obtemos

$$-F(x_k) = [F'(x_k) + \frac{1}{2}(x - x_k)^t F''(x_k)](x - x_k)$$

e

$$(x - x_k) = -[F'(x_k) + \frac{1}{2}(x - x_k)^t F''(x_k)]^{-1} F(x_k). \tag{2.4}$$

A inconveniência de (2.4) é que $(x - x_k)$ aparece em ambos os lados da igualdade.

Considere agora a aproximação de Taylor de primeira ordem para F em x_k

$$L_k(x) = F(x_k) + F'(x_k)(x - x_k). \tag{2.5}$$

Tomando $L_k(x) = 0$ obtemos

$$(x - x_k) = -F'(x_k)^{-1}F(x_k). \tag{2.6}$$

Fazemos agora uma correção de Newton, como foi feito para funções de uma variável. Substituímos (2.6) apenas no lado direito da equação (2.4) obtendo

$$(x - x_k) = -[F'(x_k) - \frac{1}{2}(F'(x_k)^{-1}F(x_k))^t F''(x_k)]^{-1}F(x_k). \tag{2.7}$$

Colocando $F'(x_k)$ em evidência, temos

$$\begin{aligned}(x - x_k) &= -[F'(x_k)(I - \frac{1}{2}F'(x_k)^{-1}(F'(x_k)^{-1}F(x_k))^t F''(x_k))]^{-1}F(x_k) \\ &= -[I - \frac{1}{2}F'(x_k)^{-1}\underbrace{(F'(x_k)^{-1}F(x_k))^t F''(x_k)}_{=F''(x_k)F'(x_k)^{-1}F(x_k)}]^{-1}F'(x_k)^{-1}F(x_k),\end{aligned}$$

e temos, finalmente, a fórmula iterativa do método de Halley

$$x_{k+1} = x_k - [I - \frac{1}{2}F'(x_k)^{-1}F''(x_k)F'(x_k)^{-1}F(x_k)]^{-1}F'(x_k)^{-1}F(x_k), \quad k = 0, 1, \ldots \tag{2.8}$$

onde $F'(x_k) \in \mathbb{R}^{n\times n}$ é a matriz jacobiana da função F e $F''(x_k) \in \mathbb{R}^{n\times n}$ é o tensor com as derivadas de terceira ordem da função F.

Com relação ao produto tensor $\cdot$ vetor, consideramos o produto como uma soma de matrizes multiplicadas por escalares:

$$\text{Se } T \in \mathbb{R}^{n\times n\times n} \text{ e } s \in \mathbb{R}^{n\times 1} \qquad \text{então} \qquad T \cdot s \in \mathbb{R}^{n\times n\times 1},$$

Portanto,

$$(T \cdot s)_{ij1} = \sum_{k=1}^{n} T_{ijk}s_{k1},$$

o que implica

$$T \cdot s = \sum_{k=1}^{n} T_{\bullet\bullet k}s_k.$$

Com isso, as entradas do tensor são acessadas à medida que vai sendo necessário.

O Algoritmo 4 apresenta o pseudo-código do método de Halley. Além da função F e da estimativa inicial, devemos fornecer a tolerância positiva ϵ (para critério de parada) e o número máximo de iterações. A interpretação da saída é análoga à dos algoritmos anteriores.

Algoritmo 4: Método de Halley para zeros de funções vetoriais

Entrada:

F (função real vetorial), $x_0 \in \mathbb{R}^n$ (ponto inicial), $\epsilon > 0$ (tolerância), $maxit$ (número máximo de iterações)

Inicialização: $Sucesso \leftarrow Verdadeiro$

$x^* \leftarrow 0$, $k \leftarrow 0$

Enquanto $\|F(x_k)\| > \epsilon$, $k \leq maxit$ e $Sucesso = Verdadeiro$

Se $F'(x_k)$ é não singular **Então**

$s \leftarrow (F'(x_k))^{-1}F(x_k)$

$P \leftarrow F''(x_k)s$

$L \leftarrow (F'(x_k))^{-1}P$

$H \leftarrow I - \frac{1}{2}L$

Se H é não singular **Então**

$d \leftarrow H^{-1}s$

$x_{k+1} \leftarrow x_k - d$

$k \leftarrow k + 1$

Caso contrário

$Sucesso \leftarrow Falso$

Caso contrário

$Sucesso \leftarrow Falso$

Se $k < maxit$ e $Sucesso = Verdadeiro$ **Então**

$x^* \leftarrow x_k$

Caso contrário

$Sucesso \leftarrow Falso$

Saída:

$Sucesso$, x^*.

2.2.1 Convergência do método de Halley

Encontramos na literatura diversos artigos relacionados à convergência do método de Halley, ver, por exemplo, [3, 4, 26, 32, 36, 37]. A grande maioria das publicações sobre o método de Halley e afins restringe-se a uma abordagem teórica, onde procura-se dar a caracterização mais abrangente possível aos objetos matemáticos relevantes. Embora de inquestionável valor, a formulação dos resultados supõe, em geral, um grande número de hipóteses, cuja verificação em casos concretos parece difícil.

Como exemplo, apresentamos abaixo os resultados de Safiev [30], o terceiro dos quais é supostamente mais "conveniente para aplicações". Assim como outros artigos nesta área, Safiev [30] apóia-se no trabalho de Kantorovich [21]. Outros trabalhos deste último autor citados na literatura pesquisada foram [20, 22, 23, 24, 25]. Em particular, Miel [28] atribui a [23] a primeira demonstração, baseada no princípio majorante, do seu teorema para a convergência do método de Newton. Seguindo esta linha, Safiev [30] introduz o operador não linear $P(x)$ que leva a bola $D(x_0, R) = \{x \mid \|x_0 - x\| < R\}$ no espaço de Banach X

ao espaço de Banach Y, três vezes continuamente diferenciável no sentido de Gateaux no fecho de $D(x_0, R)$. Diz que a equação

$$P(x) = 0, \tag{2.9}$$

é majorada pela equação

$$\phi(t) = 0, \tag{2.10}$$

onde $\phi(t)$ é uma função real em C^3, definida no intervalo $[t_0, t']$, tal que $t' = t_0 + r < t_0 + R$ se as seguintes condições são satisfeitas:

(1) $\|P(x_0)\| \leq \phi(t_0)$.

(2) O operador $\Gamma_0 = [P'(x_0)]^{-1}$ existe, $\phi'(t_0) \neq 0$ e $\|\Gamma_0\| \leq -1/\phi'(t_0)$.

(3) $\|P''(x_0)\| \leq \phi''(t_0)$, e $\|P'''(x)\| \leq \phi'''(t)$, se $\|x - x_0\| \leq t - t_0 \leq t' - t_0$.

Além das sequências $(x_k)_{k\geq 0}$ e $(t_k)_{k\geq 0}$ produzidas pela aplicação do método de Halley aos problemas (2.9) e (2.10), o resultado concerne também a convergência das sequências $(x'_0 = x_0, x'_1, x'_2 \ldots)$ e $(t'_0 = t_0, t'_1, t'_2, \ldots)$ produzidas pelas fórmulas iterativas

$$x'_{k+1} = x'_k - [I - \frac{1}{2}P'(x_0)^{-1}P''(x_0)P'(x_0)^{-1}P(x'_k)]^{-1}P'(x_0)^{-1}P(x_k), \quad k = 0, 1, \ldots \tag{2.11}$$

e

$$t_{k+1} = t_k - \frac{2f'(t_0)f(t'_k)}{2[\phi'(t_0)]^2 - \phi''(t_0)\phi(t'_k)}, \quad k = 0, 1, \ldots \tag{2.12}$$

O princípio majorante permite tirar conclusões sobre a sequência $(x_k)_{k\geq 0}$ a partir de resultados sobre a sequência $(t_k)_{k\geq 0}$. Em particular, note que a taxa de convergência para a sequência de vetores segue do resultado de convergência para o método de Halley para uma variável.

Teorema 2.2 (Teorema 1 de [30]) *Suponha que:*

(a) a equação (2.10) tem um zero $t^* \in (t_0, t']$;

(b) a equação (2.10) majoriza a equação (2.9);

(c) $\phi''(t)\phi(t)[\phi'(t)]^{-2} \leq \sigma < 2$, para $t \in [t_0, t^*]$.

Então as seguintes afirmações são válidas:

(i) existe uma solução x^* de (2.9), pertencente à bola $D(x_0, r)$, onde $r = t^* - t_0$, e a sequência $(x_k)_{k\geq 0}$ converge para esta solução;

(ii) a sequência $(t_k)_{k\geq 0}$ converge para t^*;

(iii) a taxa de convergência de $(x_k)_{k\geq 0}$ pode ser determinada pela desigualdade

$$\|x^* - x_k\| \leq t^* - t_k, \quad \forall k. \tag{2.13}$$

O Teorema 2 de Safiev [30] é praticamente o análogo de seu Teorema 1 para o par de sequências definidas por (2.11) e (2.12).

Teorema 2.3 (Teorema 2 de [30]) *Suponha que:*

(a) a equação (2.10) tem um zero $t^* \in (t_0, t']$;

(b) a equação (2.10) majoriza a equação (2.9);

(c) $\phi''(t_0)\phi(t_0)[\phi'(t_0)]^{-2} \leq \sigma < 2$.

Então as seguintes afirmações são válidas:

(i) existe uma solução x^* de (2.9), pertencente à bola $D(x_0, r)$, onde $r = t^* - t_0$, e a sequência $(x'_k)_{k\geq 0}$ converge para esta solução;

(ii) a sequência $(t'_k)_{k\geq 0}$ converge para t^*;

(iii) a taxa de convergência de $(x'_k)_{k\geq 0}$ pode ser determinada pela desigualdade

$$\|x^* - x'_k\| \leq t^* - t'_k, \quad \forall k. \tag{2.14}$$

Os Teoremas 1 e 2 de Safiev [30] envolvem a (desconhecida) função ϕ. O mérito do Teorema 3, apresentado a seguir, é propor uma forma específica para esta função, forma esta que depende de constantes que precisariam ser estimadas, no caso da aplicação do teorema a um problema concreto.

Teorema 2.4 (Teorema 3 de [30]) *Suponha que as seguintes condições estão satisfeitas para o ponto $x_0 \in X$:*

(a) $\|P(x_0\| \leq \delta$;

(b) o operador $\Gamma = [P'(x_0)]^{-1}$ existe, e $\|\Gamma\| \leq B$;

(c) no domínio $G = \{x \mid \|x - x_0\| \leq t^*\}$, onde t^* é a menor solução positiva da equação

$$\phi(t) = \frac{1}{6}Nt^3 + \frac{1}{2}Mt^2 - B^{-1}t + \delta = 0, \tag{2.15}$$

é satisfeita a desigualdade

$$M \geq \|P''(x_0)\|, \quad N \geq \sup_{x\in G} \|P'''(x)\|; \tag{2.16}$$

(d) $h = MB^2\delta \leq \dfrac{1}{2+\gamma}$, onde $\gamma = NB^{-1}M^{-2}$.

Então existe uma solução x^* da equação (2.9), e a sequência $(x_k)_{k\geq 0}$ produzida por (2.11) converge para esta solução, com taxa de convergência determinada pela desigualdade $\|x^* - x_k\| \leq t^* - t_k$, para todo k, onde a sequência $(t_k)_{k\geq 0}$ produzida por (2.12) com $t_0 = 0$ converge para t^*.

Para ainda outros resultados sobre convergência ver [10, 12, 19, 34].

Capítulo 3

O Problema de Otimização Não Linear Irrestrito

Agora estamos interessados em resolver o problema de otimização não linear dado por

$$\begin{aligned} \min \quad & f(x) \\ x \in \mathbb{R}^n & \end{aligned} \tag{3.1}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$ é uma função não linear de várias variáveis reais.

Assumimos $f \in C^3$ para que os métodos de Newton e Halley para o problema de otimização estejam bem definidos, embora para o método de Newton basta supor $f \in C^2$.

Na grande maioria dos problemas, estamos procurando tanto minimizadores locais quanto minimizadores globais da função f, como definidos abaixo.

Definição 3.1 *Considere uma função $f : \mathbb{R}^n \to \mathbb{R}$ e $x^* \in \Omega \subset \mathbb{R}^n$. Dizemos que x^* é um minimizador local de f em Ω quando existe $\delta > 0$, tal que $f(x^*) \leq f(x)$, para todo $x \in B(x^*, \delta) \cap \Omega$. Caso $f(x^*) \leq f(x)$, para todo $x \in \Omega$, x^* é dito um minimizador global de f em Ω.*

Quando as desigualdades da definição acima são estritas, dizemos que x^* é um minimizador estrito. No caso particular que estamos trabalhando, o problema de otimização é irrestrito, portanto temos que $\Omega = \mathbb{R}^n$. Questões que surgem naturalmente neste contexto são: dada uma função $f : \mathbb{R}^n \to \mathbb{R}$ é sempre possível determinar um minimizador de f? Sob quais condições podemos garantir a existência de um minimizador? Os resultados abaixo apresentam condições suficientes para a existência de minimizadores.

Teorema 3.1 (Weierstrass) *Sejam $f : \mathbb{R}^n \to \mathbb{R}$ contínua e $\Omega \subset \mathbb{R}^n$ compacto não vazio. Então existe minimizador global de f em Ω.*

Corolário 3.1 *Seja $f : \mathbb{R}^n \to \mathbb{R}$ contínua e suponha que existe $c \in \mathbb{R}$ tal que o conjunto $L = \{x \in \mathbb{R}^n \mid f(x) \leq c\}$ é compacto não vazio. Então f tem um minimizador global.*

Na próxima seção, apresentamos as condições de otimalidade do problema (3.1), base para podermos aplicar os métodos de Newton e Halley na solução do problema. Os resultados apresentados neste capítulo podem ser encontrados em [6].

3.1 Condições de otimalidade

Teorema 3.2 (Condição necessária de 1^a ordem) *Seja $f : \mathbb{R}^n \to \mathbb{R}$ diferenciável no ponto $x^* \in \mathbb{R}^n$. Se x^* é um minimizador local de f, então*

$$\nabla f(x^*) = 0 \tag{3.2}$$

Definição 3.2 *Um ponto $x^* \in \mathbb{R}^n$ que cumpre a condição (3.2) é dito ponto estacionário da função f.*

Teorema 3.3 (Condição necessária de 2^a ordem) *Seja $f : \mathbb{R}^n \to \mathbb{R}$ duas vezes diferenciável no ponto $x^* \in \mathbb{R}^n$. Se x^* é um minimizador local de f, então a matriz Hessiana de f no ponto x^* é semidefinida positiva, isto é*

$$d^T \nabla^2 f(x^*) d \geq 0, \tag{3.3}$$

para todo $d \in \mathbb{R}^n$.

Teorema 3.4 (Condição suficiente de 2^a ordem) *Seja $f : \mathbb{R}^n \to \mathbb{R}$ duas vezes diferenciável no ponto $x^* \in \mathbb{R}^n$. Se x^* é um ponto estacionário da função f e $\nabla^2 f(x^*)$ é definida positiva, então x^* é um minimizador local estrito de f.*

Definição 3.3 *Considere uma função diferenciável $f : \mathbb{R}^n \to \mathbb{R}$ e $\bar{x} \in \mathbb{R}^n$ um ponto estacionário de f. Dizemos que $\bar{x}$ é um ponto de sela da função f quando para todo $\epsilon > 0$, existem $x,\ y \in B(\bar{x}, \epsilon)$ tais que*

$$f(x) < f(\bar{x}) < f(y).$$

Teorema 3.5 *Seja $f : \mathbb{R}^n \to \mathbb{R}$ duas vezes diferenciável no ponto estacionário $\bar{x} \in \mathbb{R}^n$. Se $\nabla^2 f(\bar{x})$ é indefinida, então $\bar{x}$ é ponto de sela de f.*

Com base nos teoremas e definições apresentados nesta seção, buscamos métodos para resolver o problema de otimização (3.1), que tratamos com mais detalhes na seção seguinte.

3.2 Métodos para solução do problema de otimização irrestrito

Pelo Teorema 3.2 e pela definição 3.2 temos que se uma função continuamente diferenciável $f : \mathbb{R}^n \to \mathbb{R}$ possui um minimizador local, então esse ponto de mínimo é um ponto estacionário da função f. Sendo assim, se estamos interessados em localizar os pontos que minimizam a função f, devemos começar localizando seus pontos estacionários, ou seja

$$\text{encontrar } \bar{x} \in \mathbb{R}^n \text{ tal que } \nabla f(\bar{x}) = 0.$$

Mas sabemos que $\nabla f(x) : \mathbb{R}^n \to \mathbb{R}^n$ é uma função vetorial, então resolver o problema de otimização (3.1) é equivalente a resolver o problema de localizar os zeros de uma função vetorial. É importante observar que quando estamos localizando os zeros de $\nabla f(x)$ estamos localizando apenas os pontos estacionários de f, que são os candidatos a minimizadores.

Sendo assim, podemos empregar os métodos estudados no capítulo 3 para resolver o problema de otimização não linear irrestrito. Nas próximas seções adaptamos os métodos de Newton e de Halley para resolver o problema de otimização e apresentamos as suas fórmulas iterativas, utilizadas em nossos testes numéricos.

3.2.1 O método de Newton para o problema de otimização não linear irrestrito

Queremos então utilizar o método de Newton

$$x_{k+1} = x_k - [F'(x_k)]^{-1}F(x_k), \qquad k = 0, 1, \ldots$$

para encontrar os pontos estacionários de $f : \mathbb{R}^n \to \mathbb{R}$, ou seja, resolver $\nabla f(x) = 0$. Como estamos querendo localizar os zeros de $\nabla f(x)$, que é uma função vetorial, temos que

$$F(x) = \nabla f(x) \qquad \text{e} \qquad F'(x) = \nabla^2 f(x).$$

Assim, podemos escrever o método de Newton para o problema de otimização

$$x_{x+1} = x_k - [\nabla^2 f(x_k)]^{-1}\nabla f(x_k), \qquad k = 0, 1, \ldots$$

Se o método de Newton convergir, temos a garantia que a convergência será quadrática. Aplicado o método, devemos utilizar as condições de otimalidade de segunda ordem para analisar a natureza do ponto estacionário determinado.

3.2.2 O método de Halley para o problema de otimização não linear irrestrito

Analogamente ao que foi feito na seção anterior, suponha que queremos utilizar o método de Halley

$$x_{k+1} = x_k - [I - \frac{1}{2}F'(x_k)^{-1}F''(x_k)F'(x_k)^{-1}F(x_k)]^{-1}F'(x_k)^{-1}F(x_k), \quad k = 0, 1, \ldots$$

para encontrar os pontos estacionários de $f : \mathbb{R}^n \to \mathbb{R}$. Portanto temos que

$$F(x) = \nabla f(x), \quad F'(x) = \nabla^2 f(x) \quad \text{e} \quad F''(x) = \nabla^3 f(x).$$

Assim, com as substituições acima apresentadas, temos o método de Halley para o problema de otimização irrestrito

$$x_{k+1} = x_k - [I - \frac{1}{2}\nabla^2 f(x_k)^{-1}\nabla^3 f(x_k)\nabla^2 f(x_k)^{-1}\nabla f(x_k)]^{-1}\nabla^2 f(x_k)^{-1}\nabla f(x_k), \quad k = 0, 1, \ldots$$

Capítulo 4

Testes Preliminares - Problema de Otimização Irrestrito

4.1 Aspectos computacionais

Neste capítulo exploramos as fórmulas iterativas dos métodos de Newton e Halley buscando implementações mais eficientes. Para tanto será relevante levar em consideração a estrutura das matrizes envolvidas nestas fórmulas iterativas.

Por exemplo, embora estas fórmulas envolvam matrizes inversas, que permitem expressá-las de forma sintética, a implementação deve evitar o cálculo destas inversas, caro computacionalmente. Em particular, lembramos que, se x é dado pela fórmula

$$x = A^{-1}b,$$

então x é a solução do sistema linear

$$Ax = b.$$

Quase todos os sistemas lineares envolvidos na implementação dos métodos de Newton e Halley tem a hessiana, uma matriz simétrica, como matriz de coeficientes. Apenas o último sistema linear necessário na implementação do método de Halley possui matriz de coeficientes não simétrica.

Nossos testes foram realizados em um computador com processador *Core i5 - 2.27 GHz*, sistema operacional *Windows 7 Home Premium, 32 bits* e memória RAM de *4GB*. Nossos algoritmos foram todos implementados no MATLAB versão 7.10.0 (R2010a), que oferece rotinas eficientes para a resolução de sistemas lineares. Estas rotinas são baseadas na fatoração da matriz de coeficientes do sistema. No caso em que esta matriz é simétrica escolhemos a rotina `ldl(·)`, que realiza a fatoração LDL^T da matriz A acessando apenas os elementos armazenados na parte triangular inferior da matriz, tirando proveito de sua simetria, e pode ser aplicada para matrizes armazenadas na forma densa ou esparsa. Quando a matriz não é simétrica, utilizamos o comando 'barra invertida' do MATLAB,

que identifica o método adequado a ser utilizado na resolução do sistema. Para a resolução do sistema linear $Ax = b$ o comando seria $x = A\backslash b$.

De modo análogo, o cálculo da matriz $X = A^{-1}B$, onde $B \in \mathbb{R}^{n\times n}$, é feito via a resolução de n sistemas lineares

$$\begin{aligned} AX_{\bullet 1} &= B_{\bullet 1} \\ AX_{\bullet 2} &= B_{\bullet 2} \\ &\vdots \\ AX_{\bullet n} &= B_{\bullet n}, \end{aligned}$$

onde $X_{\bullet i}$ e $B_{\bullet i}$ são as i-ésimas colunas das matrizes X e B, respectivamente. Quando utilizamos a notação ' $\bullet$ ' estamos considerando que o subescrito naquela posição assume todos os valores possíveis.

Este artifício permite economizar na fatoração da matriz A, pois apenas uma fatoração é necessária para resolver todos os n sistemas lineares.

Na fórmula iterativa do método de Newton para o problema de otimização

$$x_{k+1} = x_k - \underbrace{\nabla^2 f(x_k)^{-1}\nabla f(x_k)}_{d} \tag{4.1}$$

a cada passo calculamos a direção d resolvendo apenas um sistema linear com a hessiana e gradiente como vetor do lado direito, $\nabla^2 f(x_k)d = \nabla f(x_k)$, e temos o próximo ponto do processo iterativo, $x_{k+1} = x_k - d$.

Já na fórmula iterativa do método de Halley temos:

$$x_{k+1} = x_k - \underbrace{\left[I - \frac{1}{2}\overbrace{(\nabla^2 f(x_k))^{-1}\underbrace{\nabla^3 f(x_k)(\nabla^2 f(x_k))^{-1}\nabla f(x_k)}_{P}}^{M}\right]^{-1}\underbrace{(\nabla^2 f(x_k))^{-1}\nabla f(x_k)}_{s}}_{d}, \tag{4.2}$$

e, como indicado em (4.2) acima, o cálculo da direção d é decomposto nos seguintes passos:

1. Resolução do sistema linear $\nabla^2 f(x_k)s = \nabla f(x_k)$, para determinar o vetor s;
2. Cálculo da matriz P, produto do tensor pelo vetor s obtido no item acima, $P = \nabla^3 f(x_k)s$;
3. Cálculo de M, resolução de n sistemas lineares, $\nabla^2 f(x_k)M_{\bullet i} = P_{\bullet i}$, $i = 1, 2, \ldots, n$;
4. Encontrar a direção d resolvendo o sistema linear $(I - \frac{1}{2}M)d = s$;
5. Obter o próximo ponto do processo iterativo, $x_{k+1} = x_k - d$.

A cada iteração do método de Halley resolvemos $n + 2$ sistemas lineares de ordem n, que envolve $O(n^3)$ operações, além de um produto de tensor $\cdot$ vetor, $O(n^3)$ operações.

Vale destacar que a matriz de coeficientes do sistema apresentado no item 4, em geral, não é simétrica uma vez que a matriz M só será simétrica se o produto $\nabla^2 f(x_k)P$ for comutativo.

Uma comparação superficial dos cálculos necessários a cada iteração levaria à conclusão da desvantagem do método de Halley perante o método de Newton, apesar da convergência mais rápida do primeiro.

4.2 Testes Preliminares

Para uma análise preliminar, vamos utilizar funções da classe

$$f_n(x) = \sum_{i=1}^{n} e^{x_i} \operatorname{sen}^2(x_{n-i+1}). \tag{4.3}$$

Implementamos os métodos de Newton e Halley no MATLAB. Variamos a dimensão n entre 10 e 50. Conseguimos variar a dimensão apenas até $n = 50$, uma vez que para a função com 100 variáveis já não é mais possível armazenar, no MATLAB, as matrizes e o tensor envolvidos no método de Halley. Utilizamos como critério de parada a norma do gradiente, com precisão de 10^{-8}. Trabalhamos com o cálculo simbólico das derivadas utilizando as rotinas do MATLAB. Consideramos o problema como sendo denso e realizamos todos os cálculos, sem explorar esparsidade e simetria. Em todos os testes tomamos o vetor de 'uns' como ponto inicial. Note que, para essa classe, conhecemos seu ponto de mínimo, $x^* = (0, \ldots, 0)$.

Na Tabela 4.1 apresentamos o número de iterações e o tempo de execução obtidos para os métodos de Newton e Halley quando implementados da forma descrita acima.

n	**Halley**		**Newton**	
	iterações	*tempo (s)*	*iterações*	*tempo (s)*
10	4	2.321	6	1.579
20	4	7.039	6	2.523
30	4	17.051	6	3.881
40	4	34.686	6	5.368
50	4	66.756	6	7.110

Tabela 4.1: Resultados dos testes para funções de várias variáveis

Na Figura 4.1 apresentamos a representação gráfica dos tempos de execução da Tabela 4.1, onde podemos perceber um grande aumento do tempo de execução do método de Halley com o aumento da dimensão.

Como era de se esperar, o custo por iteração fez com que o método de Halley ficasse mais lento do que o de Newton para valores maiores do número de variáveis, apesar de continuar apresentando um número menor de iterações.

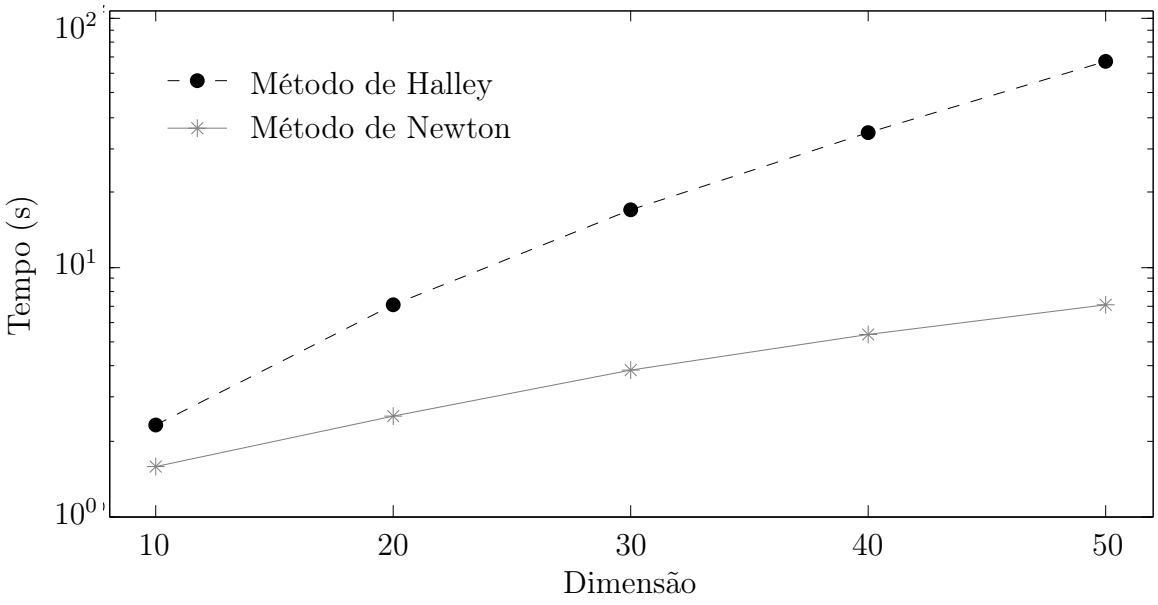


Figura 4.1: Testes numéricos preliminares dos métodos de Newton e Halley com vetor de 'uns' como ponto inicial, utilizando as rotinas do MATLAB para o cálculo das derivadas de mais alta ordem.

Embora Halley apresente um tempo próximo ao de Newton para funções de dez variáveis a diferença começa a crescer à medida que a dimensão aumenta, necessitando de um tempo quase 10 vezes maior do que o método de Newton para achar o mínimo da função com 50 variáveis, sendo que para a função com 40 variáveis o tempo era cerca de 6 vezes maior.

Um fenômeno frequente para o método de Newton e que notamos nos testes apresentados na Tabela 4.1 é que, apesar de estarmos variando a dimensão das funções, o número de iterações necessárias para a convergência do método de Newton permanece praticamente inalterado e baseados nisso, pacotes do Mathematica e do MATLAB utilizam um número pequeno como salvaguarda. Curiosamente, este fenômeno também foi observado para o método de Halley e em 40% dos nossos testes posteriores.

A matriz hessiana de f_{10} tem a estrutura de esparsidade indicada abaixo,

$$\begin{bmatrix} * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * \\ 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 \\ 0 & 0 & * & 0 & 0 & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & * & 0 & 0 & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & * & 0 & 0 & * & 0 & 0 & 0 \\ 0 & 0 & * & 0 & 0 & 0 & 0 & * & 0 & 0 \\ 0 & * & 0 & 0 & 0 & 0 & 0 & 0 & * & 0 \\ * & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & * \end{bmatrix},$$

onde os $*$'s localizados ao longo das diagonais indicam as posições das entradas não-nulas.

Este padrão se repete nas hessianas de todas as funções desta classe.

Assim como as hessianas, que possuem $O(n)$ elementos não nulos ao invés de $O(n^2)$, os tensores das funções da classe (4.3) também possuem $O(n)$ elementos não nulos e nos testes apresentados nesta seção não tiramos proveito dessa estrutura da matriz.

Essa observação levou ao desenvolvimento das implementações apresentadas na próxima seção, que exploram a possível esparsidade das matrizes de derivadas de ordem superior envolvidas nos métodos de Newton e Halley. Este fenômeno ocorre com bastante frequência em funções de grande porte, o que motiva o estudo de implementações que tirem proveito de esparsidade.

4.3 Aproveitando a esparsidade dos problemas

Funções com alto grau de esparsidade como a apresentada em (4.3) são muito comuns na prática para problemas de grande porte e o ideal é que possamos tirar proveito dessa estrutura para agilizar os cálculos e tornar os algoritmos mais eficientes. Estudos recentes, ver [16, 17, 18], mostram que tirar proveito da estrutura de esparsidade das funções podem tornar os métodos de terceira ordem competitivos com os métodos de segunda ordem.

Infelizmente, as funções do MATLAB, que calculam a hessiana e o tensor, não oferecem a possibilidade de aproveitarmos nos cálculos a estrutura de esparsidade das funções e a simetria envolvida no problema, o que diminuiria o custo computacional. Sendo asim, programamos nosso próprio algoritmo para o cálculo da hessiana e do tensor, ainda utilizando diferenciação simbólica. Com isso podemos analisar o efeito do uso da esparsidade e da simetria no desempenho do método de Halley.

Como não vamos armazenar todas as entradas da matriz, o primeiro ponto a ser pensado é uma forma eficiente de armazenar os dados para que eles possam ser facilmente acessados para a realização dos cálculos envolvidos nos métodos, incluindo a resolução dos sistemas lineares. O MATLAB oferece um bom suporte para o trabalho com dados esparsos e grande parte de suas rotinas podem ser utilizadas para matrizes armazenadas na forma esparsa, desde que elas estejam armazenadas na forma correta.

Tendo como base as rotinas disponíveis no MATLAB, armazenamos a matriz hessiana utilizando três vetores: dois vetores de inteiros, onde guardamos os índices das colunas e os índices das linhas, e um vetor de variáveis simbólicas, onde armazenamos as entradas da matriz, que são as derivadas simbólicas de segunda ordem da função.

O tensor é armazenado de forma análoga à matriz hessiana, a única diferença é que utilizamos três vetores de inteiros para armazenar os índices do tensor.

Como sabemos, dada uma função $f : \mathbb{R}^n \to \mathbb{R}$ três vezes continuamente diferenciável, o gradiente, a hessiana e o tensor das derivadas de terceira ordem, para $x \in \mathbb{R}^n$ são dados por

$$g_i(x) = \frac{\partial f(x)}{\partial x_i}, \quad H_{ij}(x) = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \quad T_{ijl}(x) = \frac{\partial^3 f(x)}{\partial x_i \partial x_j \partial x_l}, \quad 1 \leq i, j, l \leq n.$$

Então temos que H é uma matriz simétrica e T é um tensor super-simétrico, ou seja,

$$\begin{aligned} T_{ijl} = T_{ilj} = T_{jil} = T_{jli} = T_{lij} = T_{lji}, \qquad & i \neq j, j \neq l, i \neq l, \\ T_{iil} = T_{ili} = T_{lii}, \qquad & i \neq l. \end{aligned}$$

Portanto, além de não armazenarmos os elementos nulos, aproveitamos a simetria e calculamos apenas

$$\begin{aligned} H_{ij}, \qquad & 1 \leq j \leq i \leq n \\ T_{ijl}, \qquad & 1 \leq l \leq j \leq i \leq n. \end{aligned}$$

Considerando a função f_4 pertencente a classe de funções (4.3), sua matriz hessiana é

$$\begin{bmatrix} I & 0 & 0 & V \\ 0 & II & III & 0 \\ 0 & III & IV & 0 \\ V & 0 & 0 & VI \end{bmatrix},$$

calculamos apenas os elementos não nulos da parte triangular inferior desta matriz, e armazenamos em três vetores, como na Tabela 4.2.

Hi	**Hj**	**Hval**	**Ref**.
1	1	$2e^{x_4}\cos^2(x_1) - 2e^{x_4}\operatorname{sen}^2(x_1) + e^{x_1}\operatorname{sen}^2(x_4)$	I
2	2	$2e^{x_3}\cos^2(x_2) - 2e^{x_3}\operatorname{sen}^2(x_2) + e^{x_2}\operatorname{sen}^2(x_3)$	II
3	2	$2e^{x_3}\cos(x_2)\operatorname{sen}(x_2) + 2e^{x_2}\cos(x_3)\operatorname{sen}(x_3)$	III
3	3	$2e^{x_2}\cos^2(x_3) + e^{x_3}\operatorname{sen}^2(x_2) - 2e^{x_2}\operatorname{sen}^2(x_3)$	IV
4	1	$2e^{x_4}\cos(x_1)\operatorname{sen}(x_1) + 2e^{x_1}\cos(x_4)\operatorname{sen}(x_4)$	V
4	4	$2e^{x_1}\cos^2(x_4) + e^{x_4}\operatorname{sen}^2(x_1) - 2e^{x_1}\operatorname{sen}^2(x_4)$	VI

Tabela 4.2: Estrutura de armazenamento da matriz hessiana de f_4 na nossa implementação.

Analogamente à matriz hessiana, armazenamos o tensor como na Tabela 4.3.

Ti	Tj	Tk	Tval
1	1	1	$e^{x_1}\operatorname{sen}^2(x_4) - 8ex_4\cos(x_1)\operatorname{sen}(x_1)$
2	2	2	$e^{x_2}\operatorname{sen}^2(x_3) - 8ex_3\cos(x_2)\operatorname{sen}(x_2)$
3	2	2	$2e^{x_3}\cos^2(x_2) - 2ex_3\operatorname{sen}^2(x_2) + 2e^{x_2}\cos(x_3)\operatorname{sen}(x_3)$
3	3	2	$2e^{x_2}\cos^2(x_3) - 2e^{x_2}\operatorname{sen}^2(x_3) + 2e^{x_3}\cos(x_2)\operatorname{sen}(x_2)$
3	3	3	$e^{x_3}\operatorname{sen}^2(x_2) - 8e^{x_2}\cos(x_3)\operatorname{sen}(x_3)$
4	1	1	$2e^{x_4}\cos^2(x_1) - 2ex_4\operatorname{sen}^2(x_1) + 2e^{x_1}\cos(x_4)\operatorname{sen}(x_4)$
4	4	1	$2e^{x_1}\cos^2(x_4) - 2e^{x_1}\operatorname{sen}^2(x_4) + 2e^{x_4}\cos(x_1)\operatorname{sen}(x_1)$
4	4	4	$e^{x_4}\operatorname{sen}^2(x_1) - 8e^{x_1}\cos(x_4)\operatorname{sen}(x_4)$

Tabela 4.3: Estrutura de armazenamento do tensor de f_4 na nossa implementação.

Como estamos armazenando apenas a parte triangular inferior da matriz hessiana, que é a matriz de coeficientes de quase todos os sistemas lineares envolvidos nos algoritmos dos métodos de Newton e Halley, devemos escolher uma fatoração apropriada para esta matriz. Utilizamos a fatoração `ldl(·)` do MATLAB que realiza a fatoração de matrizes simétricas, não necessariamente positivas definidas, uma vez que a matriz hessiana pode ser indefinida. Essa rotina acessa apenas a parte triangular inferior da matriz e já assume que a matriz é simétrica, ideal para nossos propósitos.

Na Tabela 4.4 apresentamos os resultados dos testes em que foram utilizadas as mudanças propostas acima.

n	**Halley**		**Newton**	
	iterações	*tempo (s)*	*iterações*	*tempo (s)*
10	4	2.783	6	1.750
20	4	6.241	6	3.672
30	4	10.335	6	5.839
40	4	15.080	6	8.357
50	4	20.792	6	11.150
100	4	59.654	6	29.576
200	4	193.962	6	89.766
300	4	1073.977	6	380.597
400	4	1595.499	6	1188.384

Tabela 4.4: Resultados dos testes para funções de várias variáveis aproveitando a esparsidade e a simetria da hessiana e do tensor.

Na Figura 4.2 temos a representação gráfica dos tempos apresentados na Tabela 4.4.

Analisando os dados apresentados na Tabela 4.4 e na Figura 4.2, comparando com os dados da Tabela 4.1, percebemos uma grande melhora do método de Halley, pois além de apresentar um tempo de execução mais reduzido, ele apresenta uma variação pequena da taxa de crescimento do tempo, assim como acontece no método de Newton, à medida que a dimensão do problema cresce. Outro ponto positivo em trabalhar com os dados

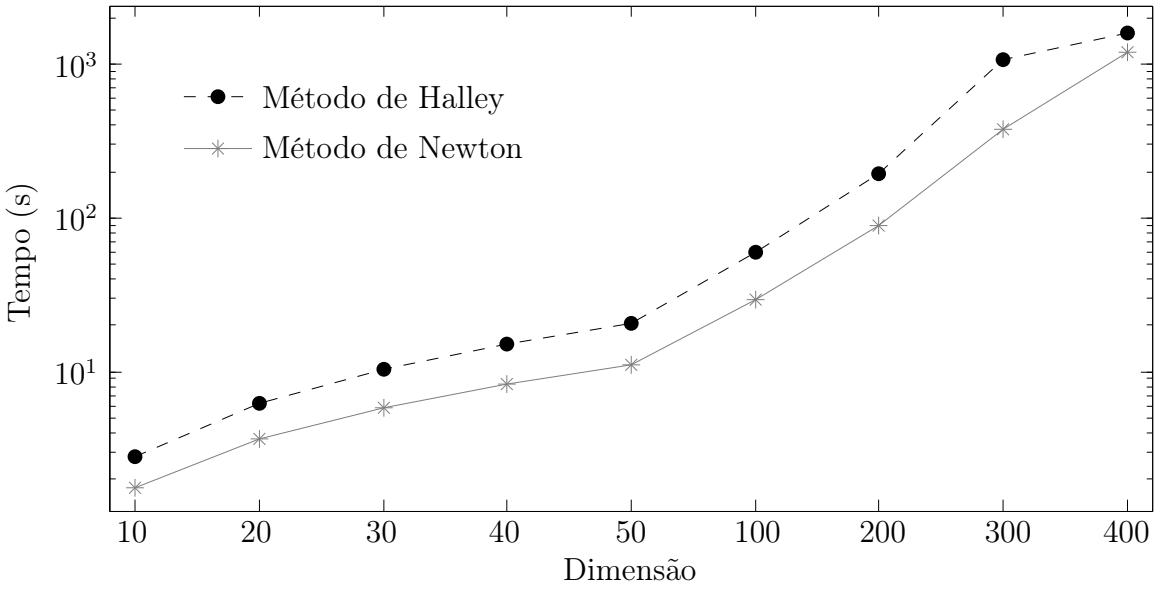


Figura 4.2: Testes númericos preliminares dos métodos de Newton e Halley com vetor de 'uns' como ponto inicial, utilizando nossa implementação para o cálculo das derivadas de mais alta ordem.

armazenados na forma esparsa é que conseguimos resolver problemas maiores, uma vez que além de não armazenarmos os elementos nulos, elementos iguais são armazenados uma única vez.

Na Figura 4.3 temos um gráfico que ilustra a melhoria que obtivemos na implementação do método de Halley ao levarmos em consideração a esparsidade das matrizes.

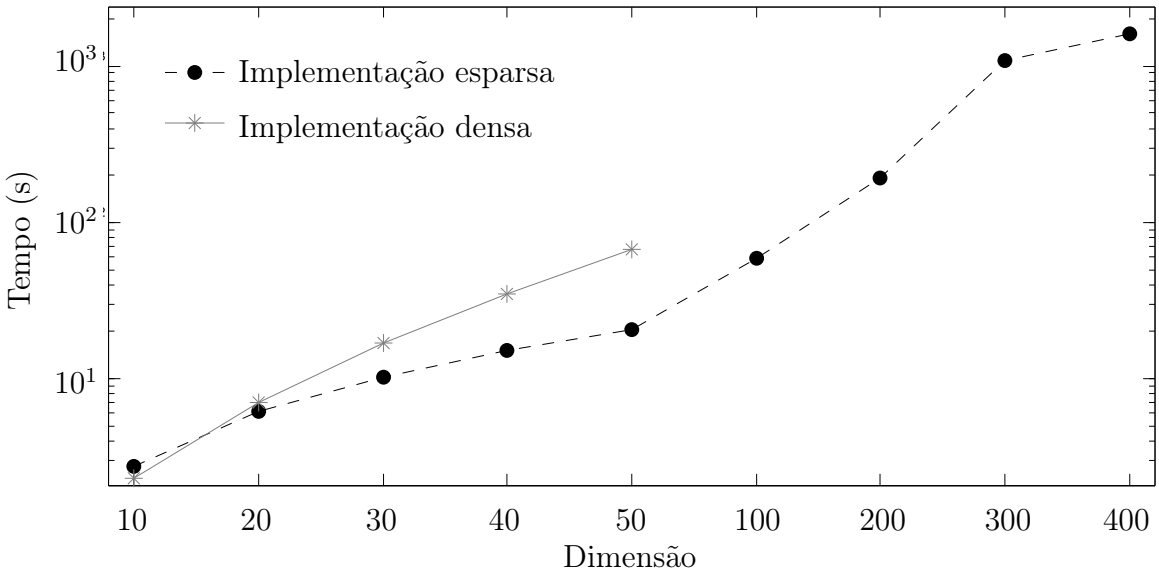


Figura 4.3: Método de Halley: Implementação densa × Implementação esparsa

No método de Newton, a princípio, parece que os resultados são piores quando estamos trabalhando com os dados esparsos. Isso acontece porque no caso denso estamos utilizando as rotinas prontas do MATLAB para calcular a hessiana, que é sempre mais rápida do que as nossas rotinas abertas. Mas à medida que a dimensão cresce, podemos notar a diferença também no método de Newton.

Na Figura 4.4 apresentamos uma comparação dos tempos de execução obtidos pelo método de Newton implementado na forma densa e esparsa, onde podemos constatar a observação feita acima.

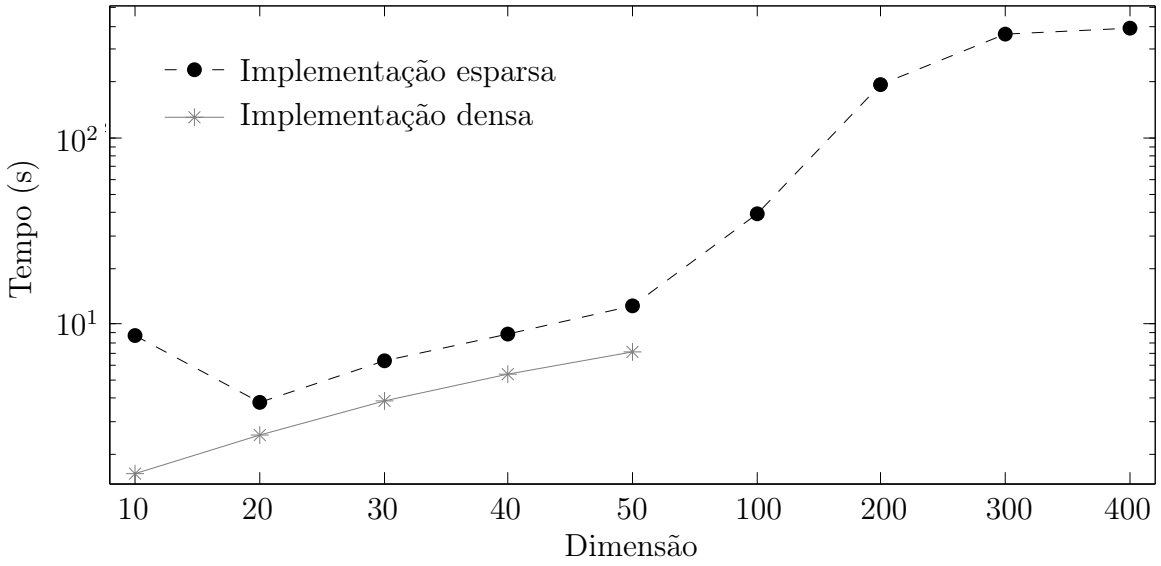


Figura 4.4: Método de Newton: Implementação densa × Implementação esparsa

4.4 Evitando calcular o tensor

Apesar de conseguirmos um desempenho melhor do método de Halley quando trabalhamos com dados esparsos, ele ainda apresenta um tempo de execução maior que o do método de Newton. Por outro lado, o número de iterações é menor na maioria dos casos e se conseguirmos diminuir o custo por iteração podemos tornar o método de Halley mais atrativo.

Um dos principais responsáveis pelo alto custo computacional do algoritmo do método de Halley é o cálculo do tensor. Por isso, propomos uma modificação ao algoritmo do método de Halley que, além de não necessitar do cálculo do tensor completo, ainda evita o cálculo do produto tensor · vetor, diminuindo o custo por iteração.

Vamos assumir que já temos o cálculo da hessiana simbólica $H(x)$ da função f e seja s o vetor que multiplica o tensor, que também vamos considerar como sendo um vetor de variáveis simbólicas. Nesse caso, podemos avaliar a hessiana no vetor $(x + ts)$

$$H(x + ts), \qquad t \in \mathbb{R} \tag{4.4}$$

e calcular a derivada parcial de (4.4) com relação a t,

$$\frac{\partial H(x+ts)}{\partial t} = T(x+ts)s. \tag{4.5}$$

Quando avaliamos (4.5) em $t = 0$ obtemos

$$T(x+ts)s\,|_{t=0} = \; T(x)s,$$

que é exatamente o produto do tensor pelo vetor utilizado no método.

Para uma análise mais aprofundada da melhoria obtida no algoritmo do método de Halley com a modificação acima proposta, vamos considerar o número de operações envolvidas nas duas formas de obtenção do produto tensor $\cdot$ vetor, tomando o pior caso, quando a hessiana e o tensor são cheios.

No algoritmo convencional realizamos:

- Cálculo de n^3 derivadas parciais de terceira ordem;
- Avaliação de n^3 entradas do tensor num ponto com n entradas, por iteração;
- Produto tensor $\cdot$ vetor necessitando de $2n^3 - n^2$ operações aritméticas, por iteração.

Já no algoritmo que evita o cálculo do tensor realizamos:

- Avaliação de n^2 entradas da hessiana no ponto $(x + ts)$;
- Cálculo de n^2 derivadas parciais;
- Avaliação de n^2 entradas do produto $P = T(x + ts)s$ em $t = 0$;
- Avaliação de n^2 entradas de P num ponto com $2n$ entradas, por iteração.

Assim, no pior caso, reduzimos um custo de $O(n^3)$ operações, a um custo de $O(n^2)$ e conseguimos diminuir consideravelmente o custo computacional do método de Halley quando utilizamos a modificação proposta nesta seção. Se pensarmos no custo por iteração, este resultado é ainda mais motivador pois, além de realizarmos um número menor de avaliações de função, as operações necessárias para o cálculo do produto tensor $\cdot$ vetor, $O(n^3)$ operações, não são mais necessárias

Buscando analisar o efeito das mudanças propostas acima, aplicamos a implementação modificada do método de Halley para algumas funções da classe (4.3), nas mesmas condições dos demais testes já apresentados neste capítulo, e os resultados obtidos podem ser encontrados na Tabela 4.5

n	Halley				Newton		
	iterações	t_1	t_2	t_3	*iterações*	t_1	t_2
10	4	2.321	2.783	3.763	6	1.579	1.750
20	4	7.039	6.241	4.900	6	2.523	3.672
30	4	17.051	10.335	8.195	6	3.881	5.839
40	4	34.686	15.080	11.531	6	5.368	8.357
50	4	66.756	20.792	14.379	6	7.110	11.150
100	4	–	59.654	78.316	6	–	29.576
200	4	–	193.962	121.507	6	–	89.766
300	4	–	1073.977	247.208	6	–	380.597
400	4	–	1595.499	824.060	6	–	1188.384

Tabela 4.5: Tempo (s) para: t_1 - implementação densa; t_2 - implementação esparsa/simétrica; t_3 - implementação que evita produto Tensor · vetor.

Na Figura 4.5 temos a representação gráfica dos dados da Tabela 4.5.

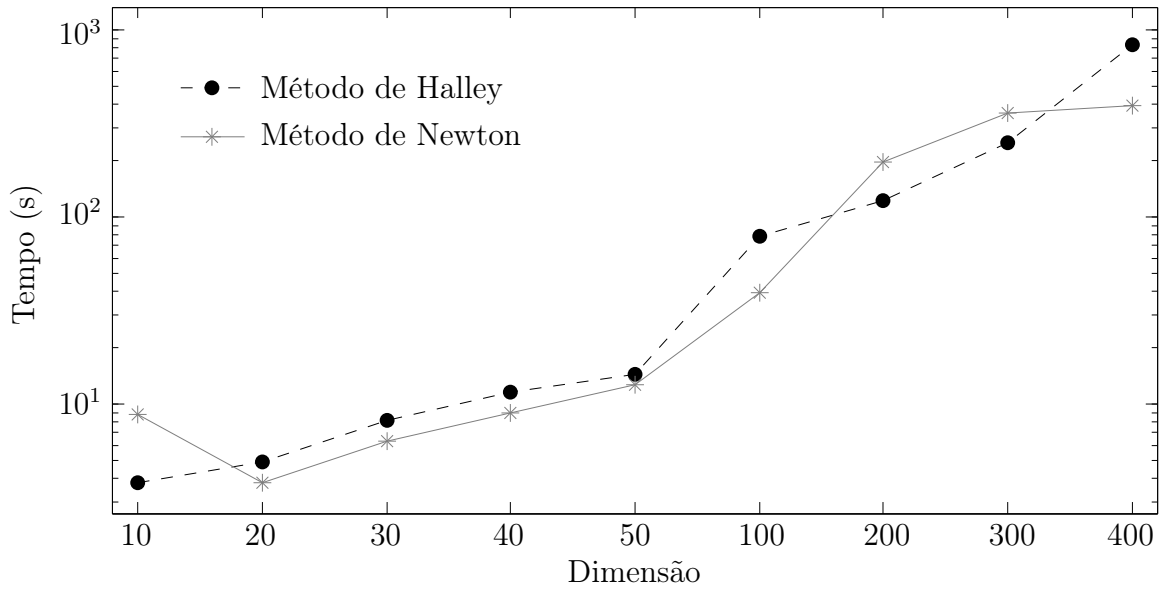


Figura 4.5: Testes numéricos preliminares dos métodos de Newton e Halley com vetor de 'uns' como ponto inicial, utilizando a implementação modificada do método de Newton.

Podemos ver que, com a modificação proposta, conseguimos reduzir a menos da metade o tempo de execução do método de Halley para um problema com 400 variáveis, obtendo resultados muito próximos aos obtidos com o método de Newton. Com o aumento da dimensão do problema, pode ser que o método de Halley venha a ser mais vantajoso pois a diferença no número de iterações pode compensar o maior custo por iteração. Infelizmente nossos testes estão condicionados às limitações impostas pela programação no MATLAB e não conseguimos atingir dimensões muito altas.

A representação gráfica dos tempos de execução do método de Halley nas três im-

plementações apresentadas neste capítulo torna mais fácil sua comparação, ver Figura 4.6.

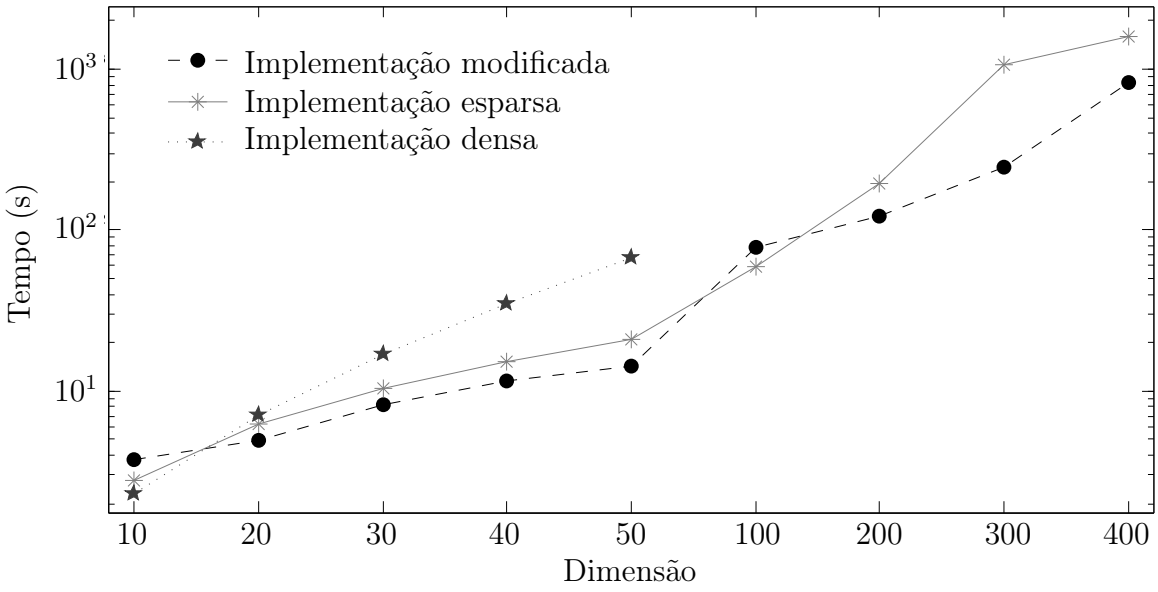


Figura 4.6: Método de Halley: Comparação do desempenho do método de Halley com respeito às modificações propostas neste capítulo.

Como os testes preliminares acima apresentados mostraram uma melhora significativa no desempenho do método de Halley, decidimos nos aprofundar nos testes utilizando os dados armazenados na forma esparsa, aproveitando a simetria da matriz hessiana e utilizando a modificação proposta nesta seção.

No próximo capítulo detalhamos os critérios de escolha dos problemas para os testes principais, assim como os resultados obtidos.

Capítulo 5

Testes Principais - Problema de Otimização Irrestrito

5.1 A escolha dos problemas teste

Em princípio nosso objetivo era selecionar 10 funções para realizarmos os testes com a implementação do método de Halley que evita o cálculo do tensor, proposto nesta dissertação, e todos os testes seriam retirados da coleção CUTE de problemas teste [7]. Entretanto, as restrições impostas para a função teste fizeram com que apenas 6 problemas fossem selecionados da coleção CUTE.

Precisávamos de problemas de classe pelo menos C^3, cuja dimensão pudesse ser variada. Demos preferência aos problemas cuja matriz hessiana não fosse mal-condicionada, uma vez que isso leva a uma convergência lenta dos métodos de Newton e Halley. Progamamos várias funções no MATLAB e selecionamos da coleção CUTE os problemas destacados abaixo:

ARWHEAD	DQRTIC	NONDQUAR	ENGVAL1	DIXMAANE	LIARWHD

Os 4 problemas que estavam faltando para completar a nossa coleção de testes foram retirados de [5], onde Andrei apresenta uma coleção de funções testes para otimização irrestrita onde estão incluídos também os problemas do CUTE. Foram programadas várias funções até encontrarmos 4 que atingissem nossos propósitos.

Portanto, os 4 problemas testes retirados de [5] são:

TRIDIAG1	HIMMELBLAU	PSC1	QF2

5.2 Resultados obtidos

Nesta seção apresentamos os resultados dos testes realizados com os 10 problemas selecionados. Relataremos sempre o número de iterações e o tempo gasto pelos algoritmos. No caso em que fazemos várias resoluções do mesmo problema com pontos iniciais diferentes, apresentaremos as médias destas medidas de performance. Para todos os problemas

fazemos uma resolução com o ponto inicial indicado na coleção e depois mais vinte, com pontos iniciais aleatórios, determinados utilizando o comando `rand` do MATLAB, com o qual selecionamos vetores com entradas reais entre -1000 e 1000. Além disso, para os problemas DIXMAANE e LIARWHD fazemos mais uma resolução com o ponto inicial indicado na coleção e utilizando estratégia de permutação para evitar o preenchimento na fatoração LDL^T. Esta estratégia foi testada para os outros problemas também, porém ela só implicou em alteração nestes dois.

Para a realização dos testes com pontos iniciais aleatórios, utilizamos os computadores do Laboratório de Informática da Pós-Graduação do IMECC. Estes computadores possuem processador *Core i7 - 3.40 GHz* , sistema operacional *Windows 7 Professional, 64 bits* e memória RAM de *4GB*. Para testes com ponto inicial dado na literatura utilizamos o computador com processador *Core i5 - 2.27 GHz*, sistema operacional *Windows 7 Home Premium, 32 bits* e memória RAM de *4GB*. Portanto, não são comparáveis os tempos de execução entre conjuntos diferentes de testes (ponto inicial aleatório e ponto inicial fornecido), mesmo que estejamos considerando o mesmo problema, na mesma dimensão.

Os testes com pontos iniciais aleatórios buscam modelar a situação mais realista, na qual não conhecemos as soluções dos problemas que tentamos resolver. Sob este ponto de vista, estes testes propiciam uma comparação mais válida da eficiência relativa dos dois algoritmos em situações práticas. A comparação com o ponto inicial fornecido na coleção permite uma padronização. Apenas um dos problemas (DIXMAANE) apresentou número de iterações alto começando com o este ponto, todos os restantes convergiram em um número bem reduzido de iterações.

Finalmente, verificamos que, para os problemas DIXMAANE e LIARWHD a versão simples da fatoração LDL^T (sem o uso de estratégias para evitar o *fill in*), usada para resolver o sistema cuja matriz de coeficientes é a hessiana, produzia mais preenchimento do que a versão usando a estratégia de permutação. Então, para estes dois problemas, a análise da matriz, para a escolha de uma permutação adequada de suas linhas e colunas, foi feita utilizando o comando `symamd(·)` do MATLAB. Este comando é aplicável a matrizes simétricas e acessa apenas a parte triangular inferior da matriz. Tem como saída um vetor de permutações que leva a um fator L mais esparso na fatoração da matriz.

Observamos um fenômeno interessante em praticamente todos os testes realizados: apesar de utilizarmos o mesmo critério de parada em ambos os algoritmos (valor da norma do gradiente menor do que ou igual a 10^{-8}) as soluções obtidas pelo algoritmo de Halley foram, em geral, mais precisas do que aquelas obtidas por Newton, em termos dos valores da função e da norma de seu gradiente no ponto final obtido.

Nos testes detalhados nas próximas seções não foram observadas falhas em nenhum dos algoritmos com os pontos iniciais aleatórios. Já com o ponto inicial fornecido na literatura, aconteceram falhas no problema DQRTIC para ambos os algoritmos.

Nos gráficos que traduzem os resultados sobre tempos apresentados nas tabelas, utilizamos sempre a escala logarítmica para os tempos, para melhor visualização.

5.2.1 Problema ARWHEAD

O problema ARWHEAD é um dos problemas da coleção CUTE [7] e é composto pela classe de funções

$$f_n(x) = \sum_{i=1}^{n-1}(-4x_i + 3) + \sum_{i=1}^{n-1}(x_i^2 + x_n^2)^2,$$

com ponto inicial

$$x_0 = (1, \ldots, 1). \tag{5.1}$$

Na Figura (5.1) podemos observar a estrutura de esparsidade da matriz hessiana desse problema para uma função com 40 variáveis. Note que os elementos não nulos estão distribuídos ao longo da diagonal principal, da última linha e da última coluna da matriz, formando o perfil de uma seta.

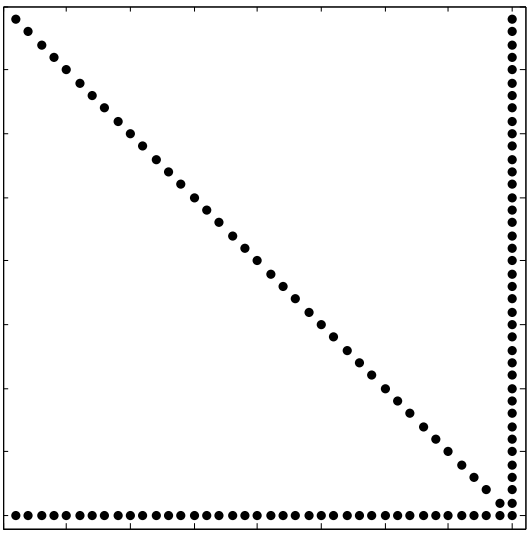
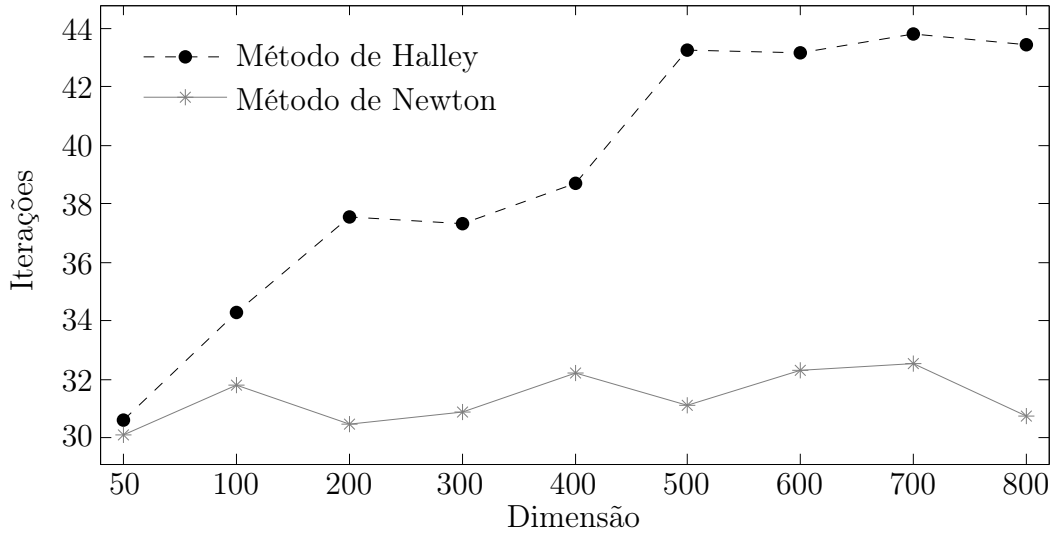


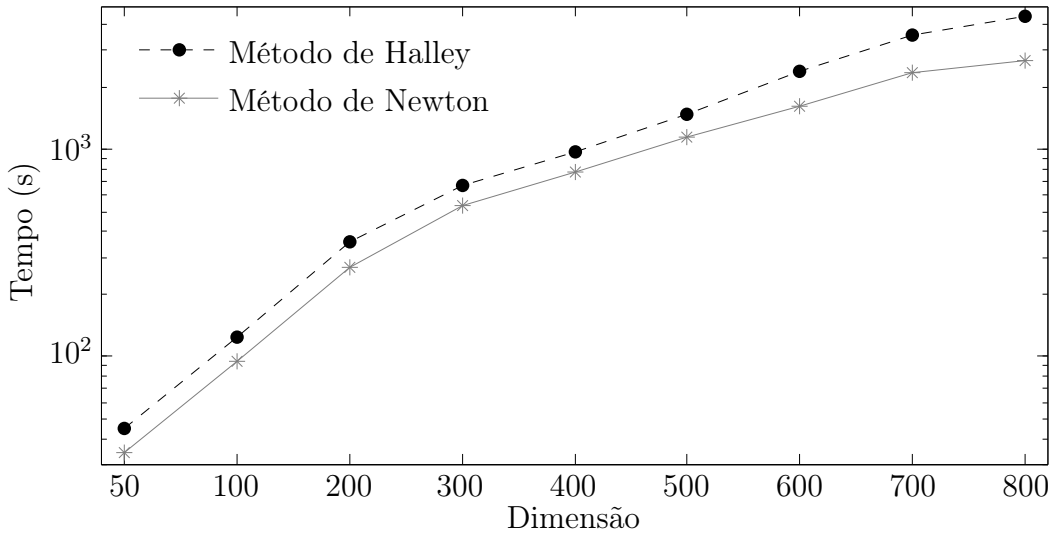
Figura 5.1: Estrutura da matriz hessiana do problema ARWHEAD, com $n = 40$.

Realizamos testes com a dimensão do problema variando entre 50 e 800 variáveis e, para cada dimensão, fizemos 20 testes com ponto inicial aleatório. Na Tabela 5.1 são apresentados a média aritmética do número de iterações e do tempo de execução em segundos dos testes para os algoritmos de Newton e Halley. Observamos que este último apresentou maior média de número de iterações e tempo de execução em todas as dimensões. Enquanto o algoritmo de Newton teve a média de iterações praticamente constante, variando entre 30 e 33, Halley apresentou maior variabilidade neste quesito. As médias dos tempos de execução de Halley acompanharam as de Newton, praticamente mantendo a proporção de 1.37 para 1.

Nas Figuras 5.2 (a) e (b) apresentamos a visualização gráfica dos dados da Tabela 5.1, facilitando a comparação do número de iterações e tempo gastos nos testes com Halley e Newton. Na escala logarítmica utilizada para as médias dos tempos na Figura 5.2 (b), a razão praticamente constante entre a média de Halley e a média de Newton se traduz em uma distância praticamente constante entre os pontos correspondentes no gráfico.



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.2: Problema ARWHEAD

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	31	44.789	30	34.271
100	34	124.200	32	94.720
200	38	357.395	30	267.401
300	37	664.550	31	533.763
400	39	967.924	32	779.050
500	43	1482.326	31	1146.049
600	43	2365.604	32	1609.275
700	44	3571.533	33	2331.957
800	43	4394.034	31	2696.219

Tabela 5.1: Problema ARWHEAD: média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

Como o método de Halley tem convergência cúbica, era esperado que ele apresentasse um número de iterações menor do que o método de Newton, que tem convergência quadrática. Quando escolhemos a aproximação inicial de forma aleatória isto não aconteceu, o que é compreensível, uma vez que o resultado de convergência é local.

Levando em consideração os resultados obtidos quando tomamos a aproximação inicial de forma aleatória, é interessante analisarmos também o comportamento dos algoritmos quando utilizamos o ponto inicial fornecido pela própria coleção. Sendo assim, realizamos testes variando a dimensão do problema entre 100 e 600 utilizando a aproximação inicial x_0 dada em (5.1). Os resultados são apresentados na Tabela 5.2.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	4	30.704	6	31.967
200	4	90.115	6	93.152
300	4	233.825	6	234.416
400	4	510.283	6	407.802
500	4	903.305	6	947.889
600	4	1020.200	6	993.362

Tabela 5.2: Problema ARWHEAD: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

Como pode ser visto na Tabela 5.2, tomando como aproximação inicial o ponto sugerido na coleção, além de obtermos uma convergência mais rápida, é possível observar a ordem de convergência dos métodos de Newton e Halley, uma vez que em todas as dimensões Halley convergiu com 4 iterações e Newton com 6.

Na Figura 5.3 temos a representação gráfica dos tempos de execução da Tabela 5.2 e

podemos observar que, com a redução no número de iterações, Halley foi mais rápido do que Newton em todos as dimensões, com exceção de 400 e 600.

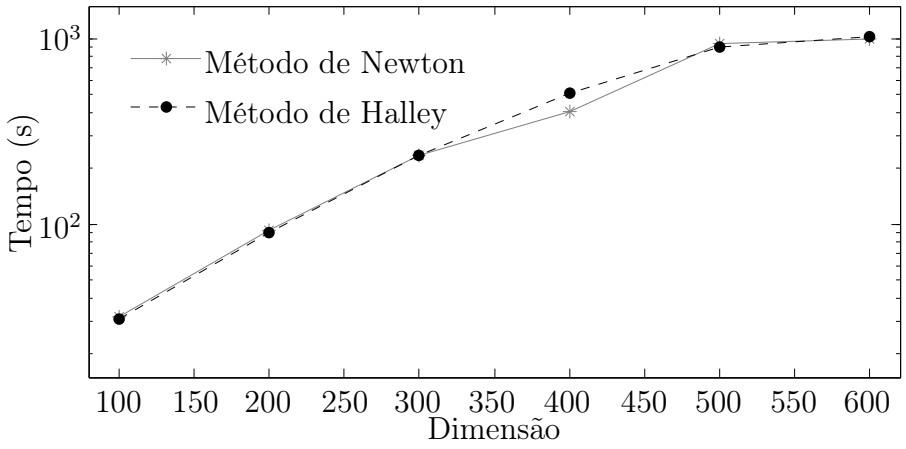


Figura 5.3: Problema ARWHEAD: tempos de execução de Newton e Halley com o ponto inicial fornecido na coleção.

5.2.2 Problema DQRTIC

O problema DQRTIC é outro problema que faz parte da coleção CUTE [7] e é composto pela classe de funções

$$f_n(x) = \sum_{i=1}^{n} (x_i - i)^4,$$

com ponto inicial

$$x_0 = (2, \ldots, 2). \tag{5.2}$$

Como f_n é uma função separável, sua hessiana é uma matriz diagonal, como podemos ver exemplificado na Figura 5.4, para $n = 40$.

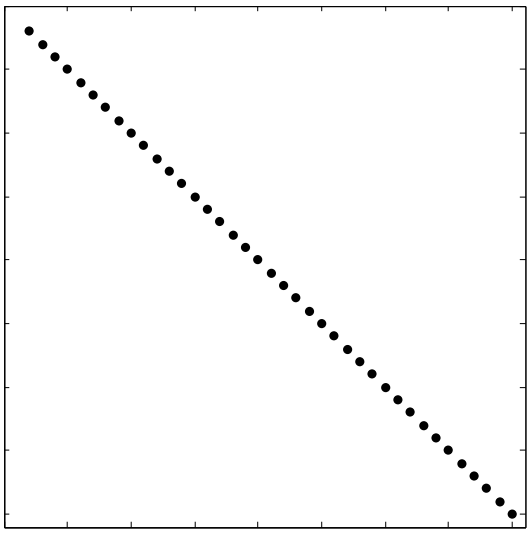


Figura 5.4: Estrutura da matriz hessiana do problema DQRTIC.

Neste problema temos um caso bem particular, onde o gradiente, a hessiana e o tensor possuem o mesmo número de entradas, igual a dimensão n do problema. Neste caso não é necessário realizar a fatoração da matriz para a resolução dos sistemas lineares, uma vez que a matriz já se encontra na forma mais simples para resolução.

Analogamente ao problema ARWHEAD, variamos a dimensão do problema entre 50 e 800 variáveis e para cada problema foram realizados 20 testes no MATLAB, com ponto inicial aleatório.

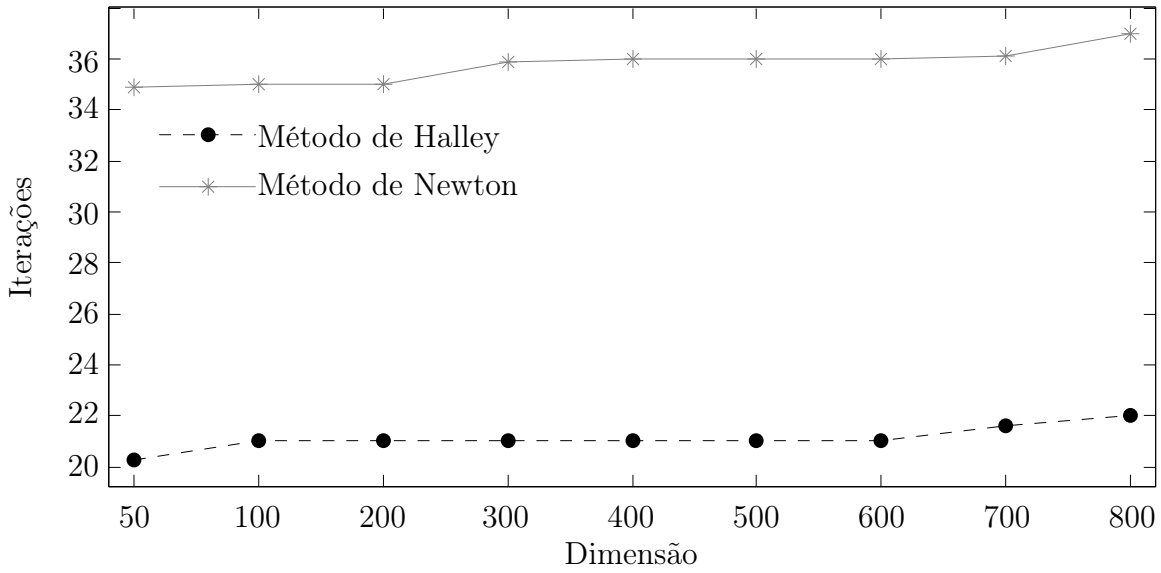
Na Tabela 5.3 apresentamos a média aritmética do número de iterações e do tempo de execução em segundos dos algoritmos de Newton e Halley para os 20 testes realizados. Note que para ambos os algoritmos a média do número de iterações permaneceu praticamente constante em todas as dimensões, variando entre 20 e 22 iterações para Halley e entre 35 e 37 para Newton. Neste problema, Halley apresentou, em média, 58% do número médio de iterações de Newton. Esta economia de iterações refletiu diretamente no tempo de execução dos algoritmos: Halley obteve o menor tempo na maioria das dimensões, embora tenha perdido eficiência para dimensões maiores, começando a apresentar um tempo superior para funções com mais de 600 variáveis.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	20	16.277	35	17.666
100	21	41.562	35	43.491
200	21	163.118	35	167.886
300	21	363.267	36	371.175
400	21	572.261	36	583.524
500	21	800.463	36	813.530
600	21	1478.642	36	1184.610
700	22	1711.409	36	1427.578
800	22	3752.160	37	2519.637

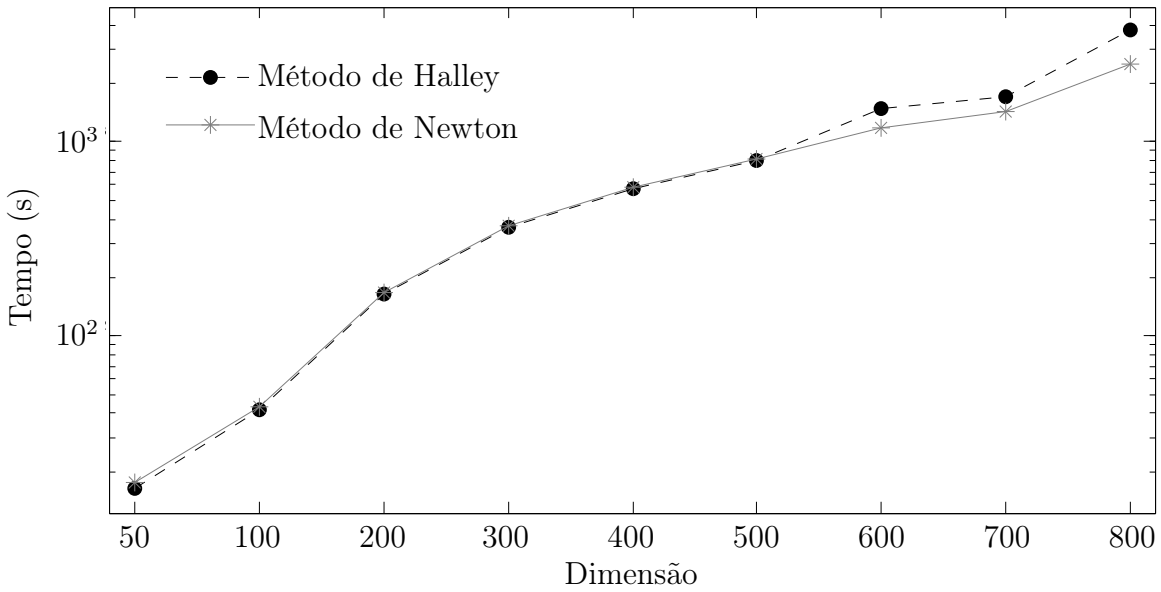
Tabela 5.3: Problema DQRTIC: média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

Nas Figuras 5.5 (a) e (b) apresentamos a representação gráfica dos dados da Tabela 5.3 para uma melhor visualização e comparação dos resultados obtidos. Analisando o gráfico da Figura 5.5 (b), vemos que, apesar de Halley apresentar um tempo menor que Newton para dimensões menores que 600, as curvas começam a se distanciar com o aumento da dimensão. Observe que a pouca variação no número de iterações está bem representada no gráfico da Figura 5.5 (a) pelas curvas praticamente horizontais ao eixo das abscissas.

Ambos os algoritmos falharam com o ponto inicial $x_0 = (2, \ldots, 2)$ dado em (5.2), pois neste ponto a hessiana de f é singular.



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.5: Problema DQRTIC.

5.2.3 Problema NONDQUAR

O problema NONDQUAR é formado pela seguinte classe de funções

$$f_n(x) = (x_1 - x_2)^2 + \sum_{i=1}^{n-2}(x_i + x_{i+1} + x_n)^4 + (x_{n-1} + x_n)^2,$$

com ponto inicial

$$x_0 = (1, -1, \ldots, 1, -1). \tag{5.3}$$

Sua matriz hessiana possui a estrutura de esparsidade indicada na Figura 5.6, onde os elementos não nulos estão distribuídos ao longo das três diagonais centrais, da primeira linha e da última coluna, no formato de uma seta.

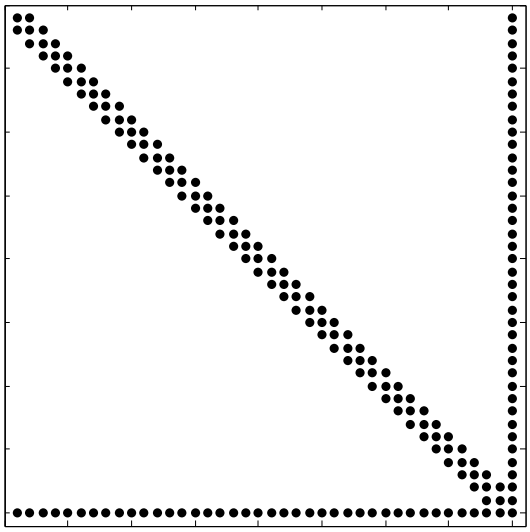


Figura 5.6: Estrutura da matriz hessiana do problema NONDQUAR.

Apresentamos na Tabela 5.4 a média aritmética do número de iterações e tempo de execução em segundos, para os algoritmos de Newton e Halley, dos 20 testes com ponto inicial aleatório, realizados no MATLAB com dimensão variando entre 50 e 800. Note que o número médio de iterações neste caso também permaneceu praticamente constante para ambos os métodos, variando entre 22 e 23 iterações para Halley e entre 37 e 40 iterações para Newton. A diferença na média de iterações dos dois métodos foi grande, com um desempenho melhor de Halley, cujo número médio de iterações obtido foi 59 % do de Newton. Essa diferença influenciou diretamente no desempenho dos algoritmos, tendo Halley apresentado um tempo de execução menor que Newton em todas as dimensões exceto $n = 600$. De fato, o comportamento de ambos os algoritmos nesta dimensão é anômalo com relação às demais.

Nas Figuras 5.7 (a) e (b) apresentamos a representação gráfica dos dados da Tabela 5.4. Observando a Figura 5.7 (a) podemos constatar a pouca variação no número de iterações apresentados pelos algoritmos de Newton e Halley, levando a curvas quase paralelas ao eixo das abscissas. Já na Figura 5.7 (b) podemos perceber que a curva que representa os

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	22	61.418	37	66.322
100	22	138.275	38	149.331
200	23	400.515	38	417.665
300	23	735.649	39	764.083
400	23	1228.872	39	1319.413
500	23	1775.557	39	1905.339
600	23	7252.614	39	6501.471
700	23	2114.400	39	2209.557
800	23	3102.908	40	3118.111

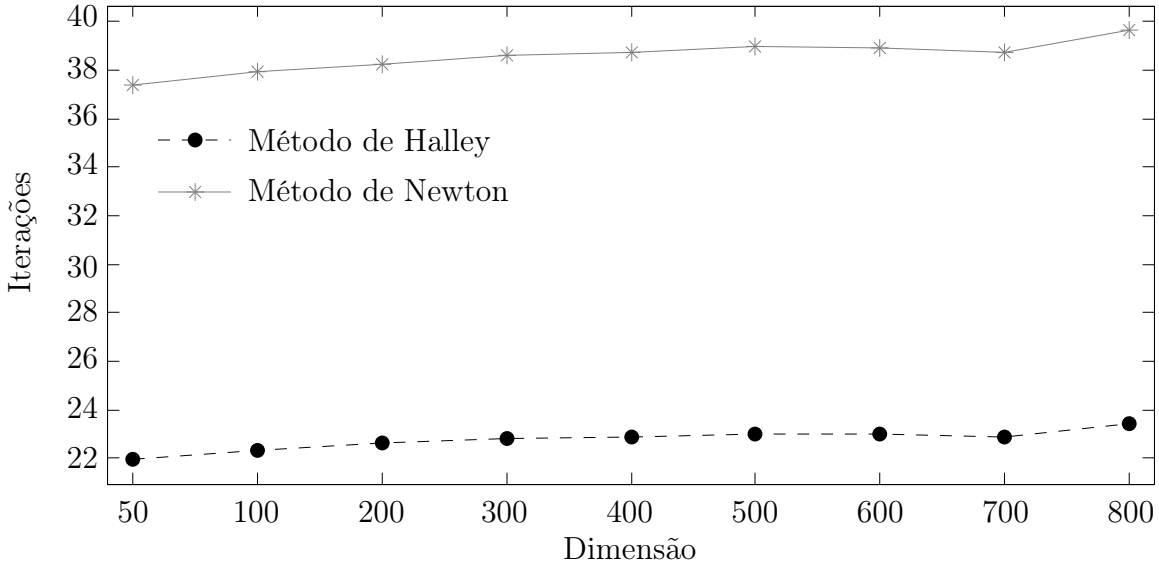
Tabela 5.4: Problema NONDQUAR: média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

dados de Halley se mantém abaixo da curva de Newton exceto para 600 variáveis. Para esta dimensão os pontos correspondentes aos dois algoritmos são claramente *outliers* com relação às curvas respectivas.

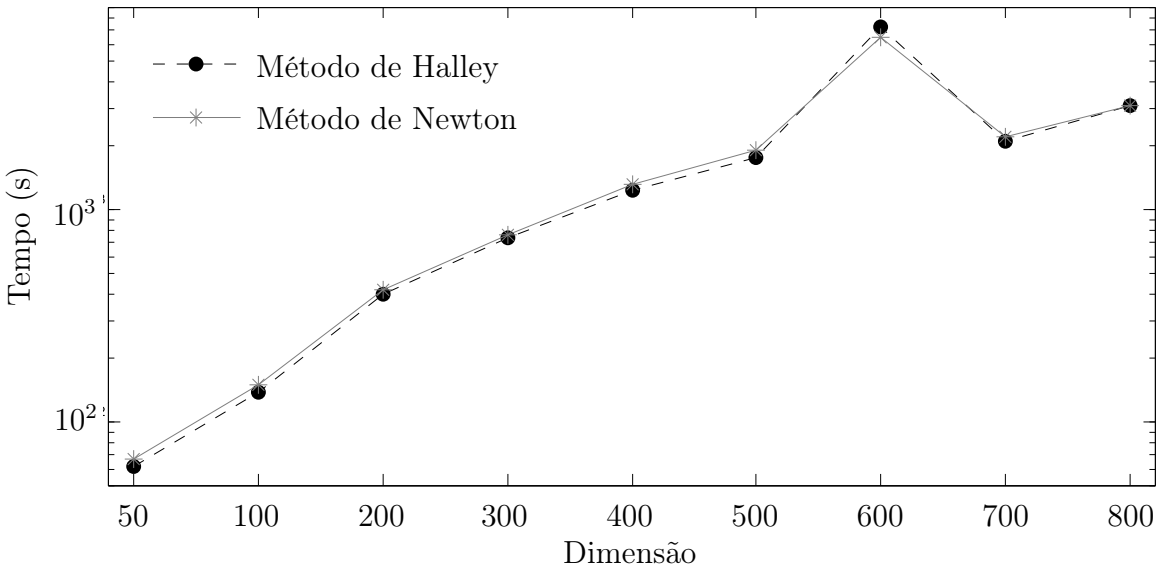
Utilizando a aproximação inicial dada em (5.3), fornecida na coleção, temos a uma convergência mais rápida dos métodos, como pode ser visto na Tabela 5.5. Para este ponto inicial também foi pequena a variação do número de iterações com relação à dimensão e, assim como nos testes com ponto inicial aleatório, pudemos verificar a influência da ordem de convergência dos métodos no número de iterações, uma vez que o método de Halley apresentou um número de iterações bem menor que o método de Newton. Os resultados neste caso apresentam uma diferença um pouco maior no tempo de execução do que com ponto inicial aleatório, para dimensões menores. Para os problemas com mais de 400 variáveis esta diferença continua muito pequena, sendo que o tempo de Halley para 400 e 600 variáveis foi um pouco maior do que o de Newton.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	12	144.667	21	157.677
200	13	222.286	21	390.377
300	13	668.421	21	718.262
400	13	654.143	22	651.976
500	13	995.225	22	1021.460
600	13	1481.220	22	1457.120

Tabela 5.5: Problema NONDQUAR: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.7: Problema NONDQUAR.

Na Figura 5.8 temos a representação gráfica dos tempos de execução da Tabela 5.5 para uma melhor visualização dos dados. Observando o gráfico fica evidente a superioridade do método de Halley para dimensões menores e uma difença praticamente imperceptível nos tempos para as dimensões maiores.

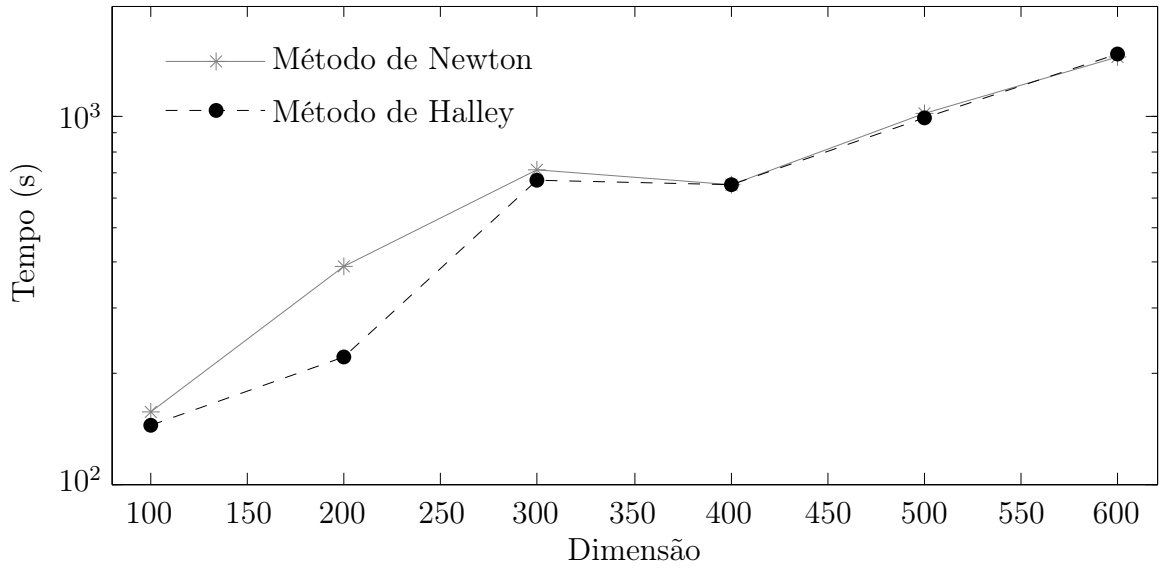


Figura 5.8: Problema NONDQUAR: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

5.2.4 Problema ENGVAL1

O problema ENGVAL1 é composto pela seguinte classe de funções

$$f_n(x) = \sum_{i=1}^{n-1}(x_i^2 + x_{i+1}^2)^2 + \sum_{i=1}^{n-1}(-4x_i + 3),$$

com ponto inicial

$$x_0 = (2, \ldots, 2). \tag{5.4}$$

Na Figura 5.9 ilustramos a estrutura de esparsidade da matriz hessiana da função com 40 variáveis, que é uma matriz tridiagonal.

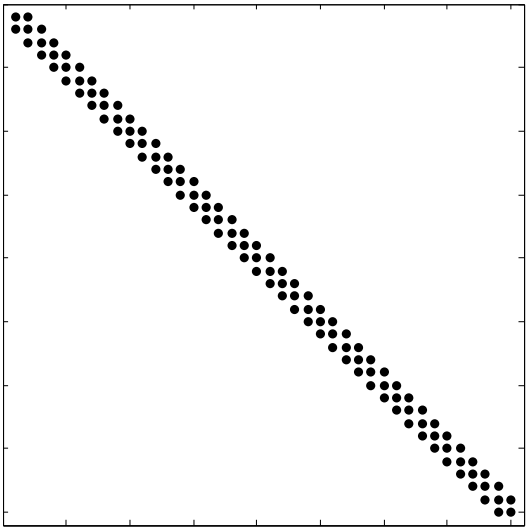


Figura 5.9: Estrutura da matriz hessiana do problema ENGVAL1.

Na Tabela 5.6 apresentamos a média aritmética do número de iterações e tempo de execução, dos algoritmos de Newton e Halley, para 20 testes com ponto inicial aleatório. Note que o número de iterações obtido por Halley variou entre 22 e 28 iterações, enquanto que para Newton variou entre 26 e 30 iterações. Além de Newton apresentar uma variação menor no número de iterações seu desempenho relativo ao método de Halley foi melhor, levando em consideração a ordem de convergência de ambos os métodos, uma vez que, em média, o número de iterações de de Halley foi 90 % do de Newton, quando deveria ser cerca de 66 % no caso de ambos os métodos convergirem nas taxas esperadas. Vemos que este melhor desempenho relativo de Newton interfere diretamente no tempo de execução dos métodos, uma vez que a média do tempo apresentado por Halley foi maior em todas as dimensões.

Nas Figuras 5.10 (a) e (b) temos a representação gráfica dos dados da Tabela 5.6, para o número de iteração e tempo de execução respectivamente. Observando os gráficos fica mais fácil a comparação dos resultados. Na Figura 5.10 (a) vemos a maior variação dos números de iterações apresentados por ambos os métodos, com a diferença no número de iterações diminuindo com o aumento da dimensão. Na Figura 5.10 (b) vemos que a

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	22	38.412	26	34.951
100	22	89.236	26	80.954
200	24	288.063	28	268.011
300	25	492.740	29	458.330
400	25	935.880	28	872.427
500	25	1325.745	29	1239.509
600	26	1894.234	29	1567.718
700	28	2817.477	30	2118.977

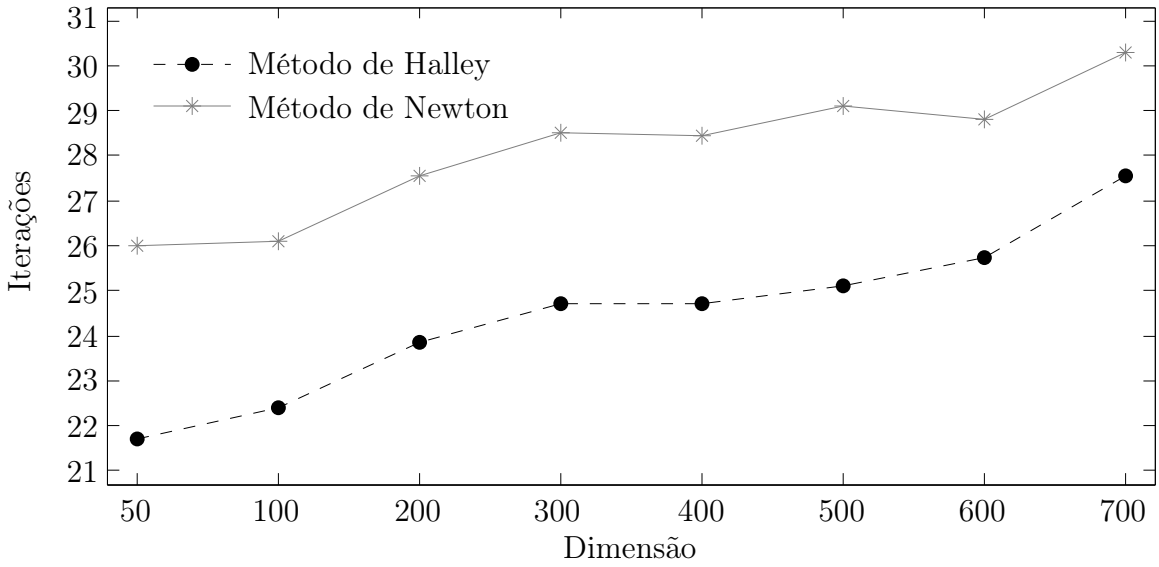
Tabela 5.6: Problema ENGVAL1: média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

curva do tempos de Halley permanece sempre acima da curva de Newton, embora estejam bem próximas, vemos que a diferença no tempo de execução cresce com o aumento da dimensão, o que é esperado uma vez que a diferença no número de iterações é menor.

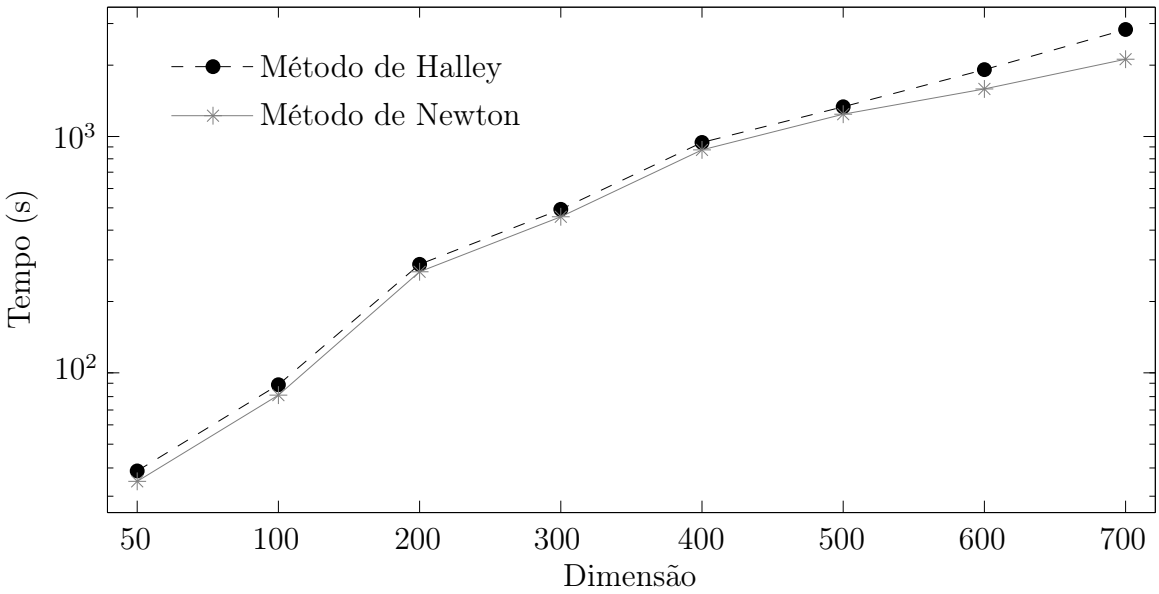
A pouca diferença entre os números de iterações obtidos pelos algoritmos de Newton e Halley ocorre porque, em alguns dos testes realizados, o método de Halley necessitou de um número maior de iterações do que o método de Newton para convergir, interferindo diretamente na média do número de iterações e tempo de execução do método de Halley. Em particular, nos testes em que o método de Halley convergiu com pelo menos 9 iterações a menos que o método de Newton, ele obteve um tempo de execução menor.

Na Tabela 5.7 apresentamos os resultados obtidos para os testes dos métodos de Newton e Halley com o ponto inical dado em (5.4), fornecido na coleção, com n variando de 100 a 600. Vemos que agora o número de iterações apresentado por Halley foi cerca de 62% do obtido por Newton, e este número se manteve constante em todas as dimensões, assim pudemos observar a ordem de convergência de ambos os métodos. Com isso, vemos que os tempos de execução dos métodos tornam-se próximos, sendo em algumas dimensões maiores para Newton do que para Halley e em outras maiores para Halley.

Na Figura 5.11 temos a representação gráfica dos tempos de execução da Tabela 5.7. Pelo gráfico fica clara a próximidade dos tempos de execução uma vez que as curvas para Newton e Halley estão sobrepostas, tendo uma diferença maior apenas para 600 variáveis, onde Halley apresentou um número de iterações maior.



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.10: Problema ENGVAL1

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	5	35.213	8	34.387
200	5	93.810	8	97.940
300	5	237.679	8	240.744
400	5	412.311	8	413.747
500	5	610.618	8	609.086
600	5	1076.670	8	885.163

Tabela 5.7: Problema ENGVAL1: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

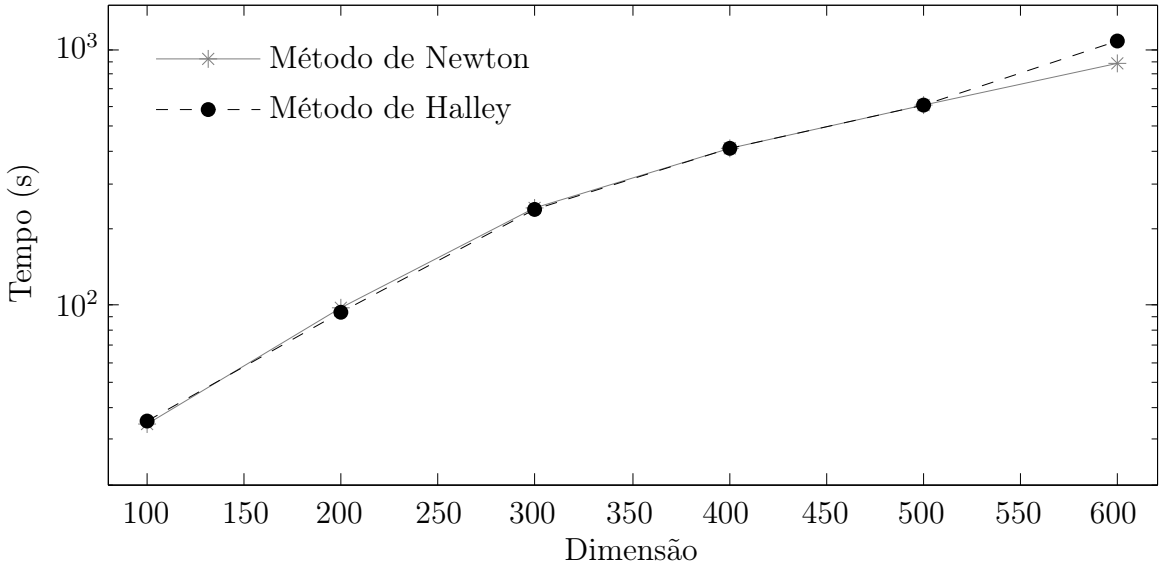


Figura 5.11: Problema ENGVAL1: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

5.2.5 Problema DIXMAANE

O problema DIXMAANE, com uma mudança em um dos parâmetros, é dado pela seguinte classe de funções

$$f_n(x) = 1 + \sum_{i=1}^{n} x_i^2 \left(\frac{i}{n}\right) + \sum_{i=1}^{n-1} \frac{1}{2} x_i^2 (x_{i+1} + x_{i+1})^2 + \sum_{i=1}^{2m} \frac{1}{8} x_i^2 x_{i+m}^4 + \sum_{i=1}^{m} \frac{1}{8} x_i x_{i+2m} \left(\frac{i}{n}\right)$$

$$m = (n/3)$$

com ponto inicial

$$x_0 = (2, \ldots, 2). \tag{5.5}$$

A matriz hessiana desse problema tem a estrutura de esparsidade representada na Figura 5.12 para $n = 40$. Note que os elementos não nulos estão localizados ao longo das três diagonais principais, e mais quatro diagonais mais afastadas, duas acima da diagonal principal e duas abaixo.

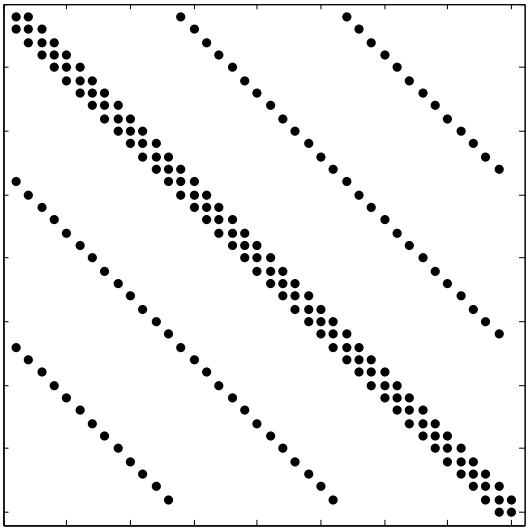


Figura 5.12: Estrutura da matriz hessiana do problema DIXMAANE.

Na Tabela 5.8 apresentamos a média aritmética do número de iterações e tempo de execução dos algoritmos de Newton e Halley para 20 testes com ponto inicial aleatório. Note que, tanto para Newton quanto para Halley, este problema apresentou uma variação grande no número de iterações, entre 54 e 110 iterações para Halley e entre 69 e 118 iterações para Newton. Embora na maioria dos casos o número de iterações obtido para Halley seja menor que o de Newton, não foi em todas as dimensões que esta diferença exemplifica a taxa de convergência dos métodos. Neste problema a média do tempo de execução de Halley foi maior que a de Newton em todas as dimensões e observamos diferenças bem significativas entre os resultados.

Nas Figuras 5.13 (a) e (b) temos a representação gráfica dos números de iteraçõe e tempos de execução da Tabela 5.8. Observando as Figuras fica mais fácil a comparação

n	Halley		Newton	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	110	260.157	69	80.258
100	54	578.850	86	431.137
200	67	1691.104	93	1087.265
300	76	3462.923	93	1933.208
400	65	5349.624	107	3649.208
500	80	10868.338	108	5119.347
600	73	13510.416	110	7351.394
700	68	18684.501	103	6297.788
800	80	22893.465	118	13827.263

Tabela 5.8: Problema DIXMAANE: média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

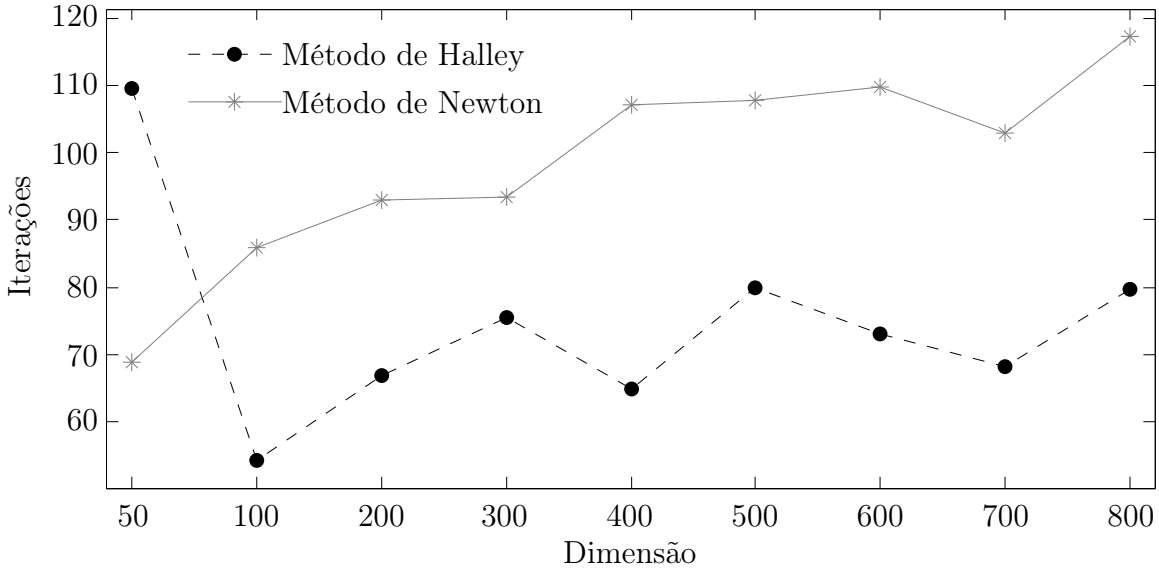
dos resultados. Na Figura 5.13 (a) fica evidente a grande variação do número de iterações obtidos pelos métodos de Newton e Halley, uma vez que as curvas de ambos os métodos não são suaves. Apesar da grande variação no número de iteraçõs dos métodos, as curvas para o tempo de execução, representadas na escala logaritmica, dadas na Figura 5.13 (b), não sofrem tanta variação e as curvas para Newton e Halley apresentam taxa de crescimento bem próximos com o Halley apresentando um desempenho inferior em todos os testes.

A diferença no tempo de execução dos métodos de Newton e Halley pode estar relacionada ao fato deste problema possuir uma matriz hessiana com uma proporção maior de elementos não nulos. Muitos destes elementos estão distribuídos ao longo de duas diagonais afastadas da diagonal principal, o que certamente leva *fill in* do fator L da fatoração LDL^T.

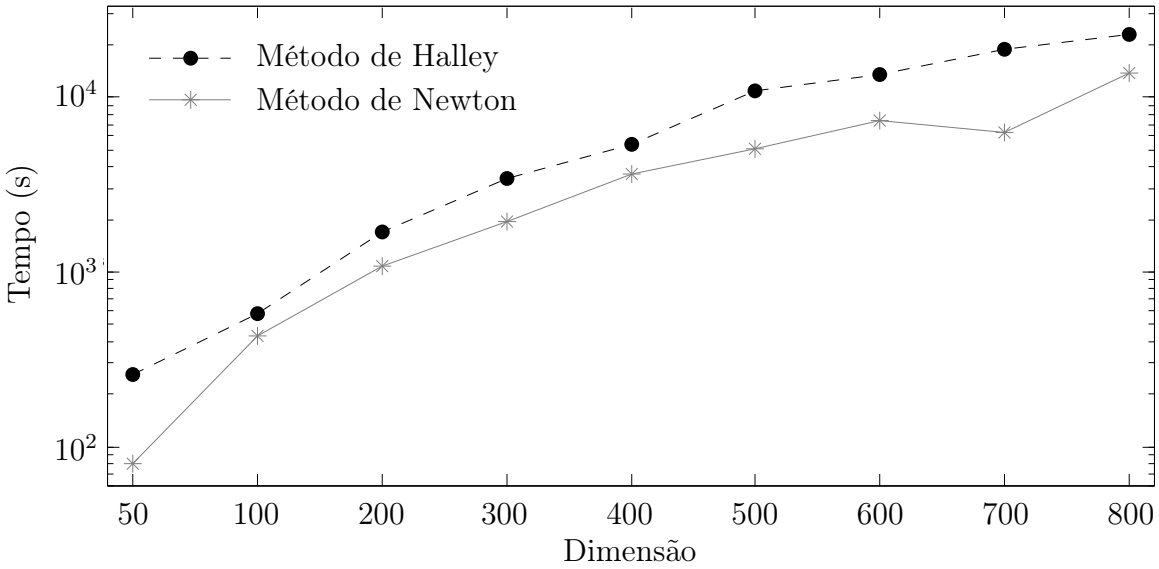
Para verificar este fato, realizamos testes com o ponto inicial fornecido na coleção, dado em 5.5. Variamos a dimensão entre 100 a 600 e aplicamos o mesmo algoritmo utilizado com ponto inicial aleatório. Para comparação, aplicamos também o algoritmo onde fizemos uso do comando `symamd(·)` para realizar a análise da esparsidade da matriz e liberar a permutação de linhas e colunas mais apropriada.

Na Tabela 5.9 apresentamos os resultados do número de iterações e tempo de execução em segundos obtidos com o algoritmo de Newton e Halley utilizando o ponto inicial fornecido na coleção. Este problema foi o único que apresentou grande variação do número de iterações quando tomamos o ponto inicial fornecido na bibliografia, variando entre 33 e 79 iterações para Halley e entre 49 e 93 para Newton, sendo que para 300 e 400 variáveis o número de iterações de Halley foi maior do que o de Newton. Devido a isso, observamos um desempenho do algoritmo de Halley, relativo ao de Newton, bem pior, apresentando um tempo de execução bem maior do o de Newton em todas as dimensões.

Os resultados apresentados na Tabela 5.9 para o tempo de execução podem ser ana-



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.13: Problema DIXMAANE

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	48	938.986	59	560.173
200	46	1916.790	49	823.300
300	79	5864.800	67	1901.710
400	76	4439.520	56	2604.960
500	35	3319.610	93	3118.270
600	33	4436.070	72	3405.950

Tabela 5.9: Problema DIXMAANE: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

lisados mais facilmente quando tomamos sua representação gráfica dada na Figura 5.14. Pelo gráfico fica claro que o desempenho do algoritmo de Halley foi bem inferior ao que obtivemos para os testes com ponto inicial aleatório para dimensões menores. Já para dimensões maiores verificamos exatamente o contrário pois quando tomamos o ponto inicial fornecido na coleção conseguimos tempos de execução mais próximos do que os obtidos com ponto inicial aleatório. Os resultados para este problema apresentam grande variação porque em algumas iterações dos métodos encontramos matrizes mal-condicionadas.

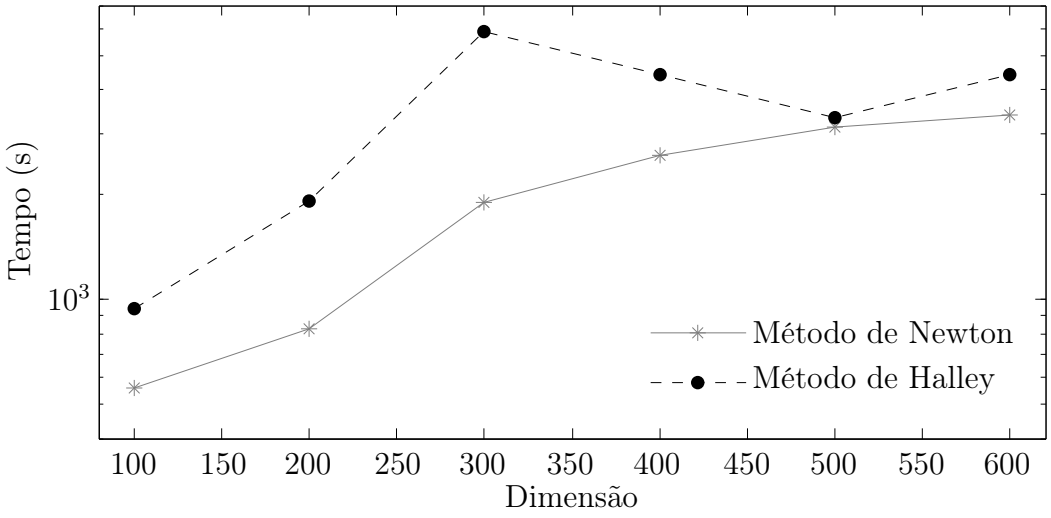


Figura 5.14: Problema DIXMAANE: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

Ao aplicarmos a estratégia de permutação de linhas e colunas da matriz hessiana deste problema, obtemos uma grande alteração na posição dos elementos não nulos, como pode ser verificada na Figura 5.15, onde apresentamos a estrutura da matriz hessiana do problema DIXMAANE após ser aplicada a estratégia de permutação.

Na Tabela 5.10 estão os resultados dos testes onde empregamos uma estratégia de permutação de linhas e colunas da matriz hessiana, buscando obter um fator L mais

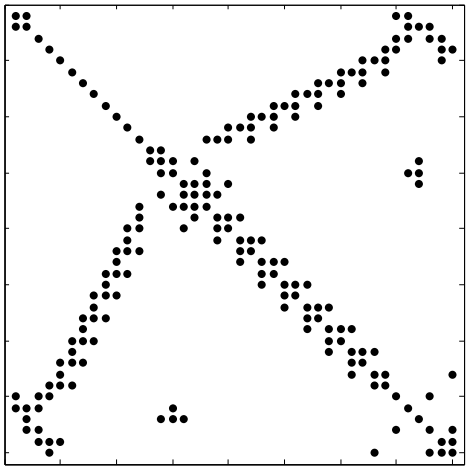


Figura 5.15: Estrutura da matriz hessiana do problema DIXMAANE após estratégia de permutação de suas linhas e colunas.

esparso quando realizamos a fatoração LDL^T. Com esta estratégia, percebemos que o número de iterações continua apresentando uma variação grande, embora ela seja menor do que sem permutação. Tivemos uma melhora significativa nos tempos de execução de Halley também e pela primeira vez ele apresenta um tempo menor do que o método de Newton, para o problema com 600 variáveis.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	40	386.871	57	260.324
200	39	895.854	55	573.467
300	65	2518.550	79	1391.480
400	79	4530.960	81	2038.730
500	39	5121.430	85	2892.810
600	33	4576.980	106	5363.340

Tabela 5.10: Problema DIXMAANE: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção e estratégia de permutação das linhas e colunas da matriz hessiana.

Para a melhor visualização dos tempos de execução da Tabela 5.10, apresentamos a representação gráfica dos dado na Figura 5.14. Vemos que conseguimos uma boa melhora no desempenho do método de Halley ao empregarmos uma estratégia de permutação na matriz hessiana do problema, principalmente para dimensões menores, cujo desempenho do algoritmo ficou próximo ao que tinhamos conseguido com os testes com ponto inicial aleatório. O algoritmo do método de Halley torna-se mais eficiente com o aumento da dimensão e chega a apresentar um tempo de execução menor do que o do método de Newton para a função com 600 variáveis.

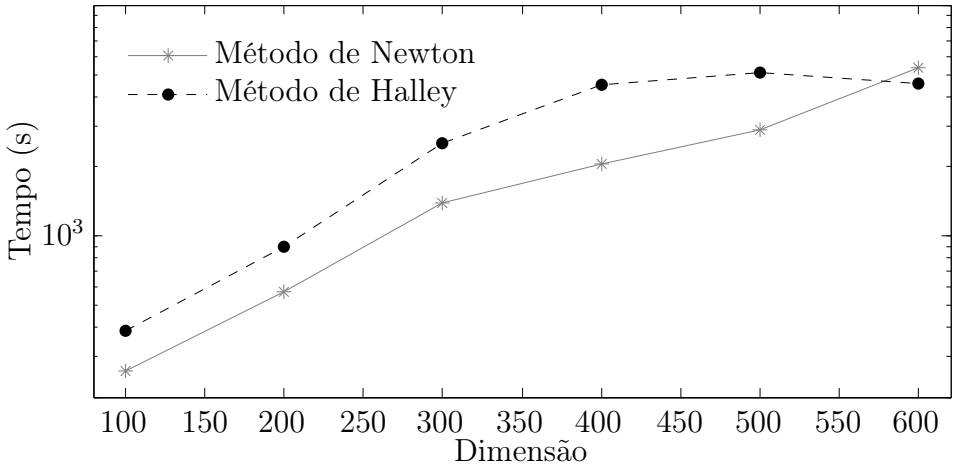


Figura 5.16: Problema DIXMAANE: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção e estratégia de permutação das linhas e colunas da matriz hessiana.

5.2.6 Problema LIARWHD

O Problema LIARWHD é o último problema selecionado da coleção CUTE, e sua expressão é dada por

$$f_n(x) = \sum_{i=1}^{n} 4(-x_1 + x_i^2)^2 + \sum_{i=1}^{n}(x_i - 1)^2,$$

com ponto inicial

$$x_0 = (4, \ldots, 4). \tag{5.6}$$

A estrutura de esparsidade da matriz hessiana da função com 40 variáveis pode ser visualizada na Figura 5.17, onde podemos perceber que seus elementos não nulos estão distribuídos ao longo da diagonal principal, da primeira linha e da primeira coluna da matriz, no formato de uma seta.

As médias aritméticas dos números de iterações e dos tempos de execução dos 20 testes realizados no MATLAB com ponto inicial aleatório estão apresentados na Tabela 5.11. Para este problema, variamos a dimensão entre 50 e 800 variáveis. Note que para este problema, assim como para o problema ARWHEAD, o número de iterações obtido pelo algoritmo de Halley é maior do que o de Newton. Além disso o número de iterações varia bastante quando tomamos dimensões diferentes, entre 35 e 102 iterações, enquanto que no algoritmo de Newton o número de iterações permanece praticamente constante, variando entre 29 e 31 iterações. Devido a isso, vemos que o método de de Halley apresenta uma média maior do tempo de execução para todas as dimensões.

Os resultados apresentados na Tabela 5.11 podem ser melhor visualizados se considerarmos a representação gráfica dos dados, dada nas Figuras 5.18 (a) e (b). Na Figura 5.18 (a) fica claro a grande variação no número de iterações apresentada pelo método de

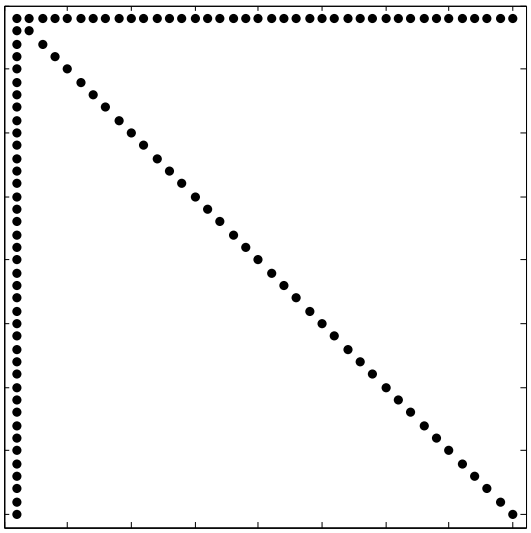


Figura 5.17: Estrutura da matriz hessiana do problema LIARWHD.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	51	56.565	29	29.555
100	35	59.952	31	45.910
200	41	215.403	33	171.708
300	40	382.514	30	308.696
400	36	669.387	31	544.706
500	48	913.120	30	689.921
600	62	2082.269	30	1472.728
700	102	3989.696	30	1780.795
800	68	4342.147	29	2332.416

Tabela 5.11: Problema LIARWHD : média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

Halley contra o número praticamente constante apresentado por Halley. Apesar da grande diferença no número de iterações a taxa de crescimento dos tempos de execução para os métodos de Newton e Halley é praticamente constante e as curvas dos tempos de execução de Newton e Halley crescem juntas, apesar de Halley sempre apresentar um tempo maior, como ilustrado na Figura 5.18. Nesta situação, o maior custo computacional do método de Halley deveria levar a um tempo de execução do método bem maior.

Vale ressalta que o fato do método de Halley apresentar um número maior de iterações é curioso uma vez que o método de Halley deveria apresentar um número menor de iterações, devido a sua ordem de convergência. Isso não contradiz o resultado de convergência apresentado no Capítulo 2, pois o resultado de ordem de convergência é local.

Os resultados obtidos para o número de iteração e tempo de execução, em segundos, dos métodos de Newton e Halley quando tomamos a aproximação inicial indicada na coleção, dada em 5.6 são apresentados na Tabela 5.12. Note que neste caso obtemos números de iterações constantes, tanto para Newton quanto para Halley e a taxa de convergência dos métodos também é observada, com Halley apresentando 63% do número de iterações de Newton. Com isso, Halley apresenta tempo de execução menor em todas as dimensões, embora a diferença nos tempos ainda seja pequena.

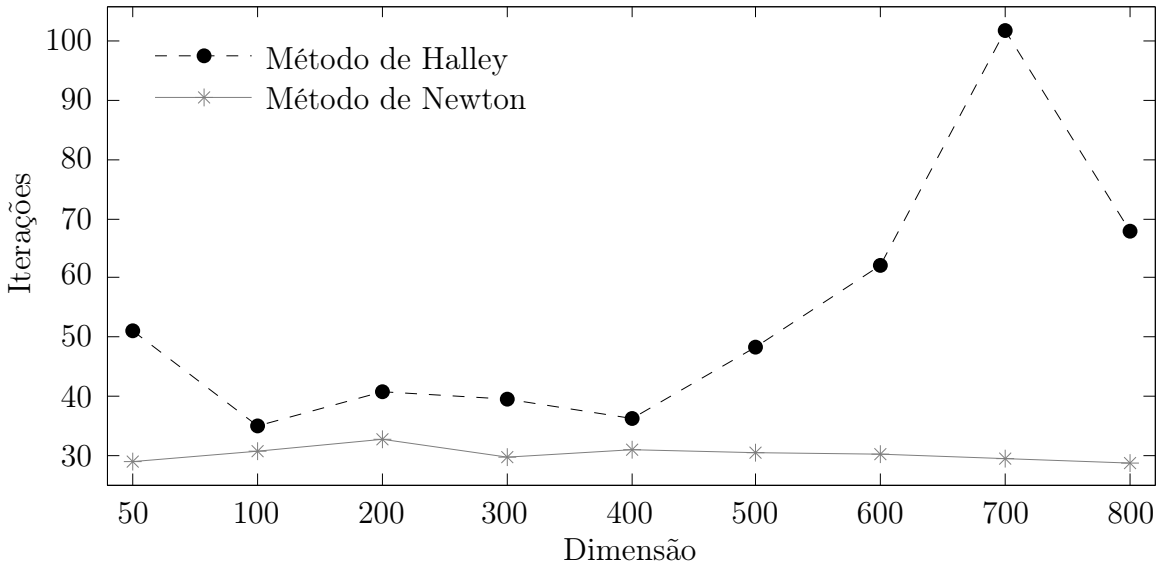
n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	7	31.207	11	34.084
200	7	96.239	11	96.490
300	7	240.926	12	249.697
400	7	418.982	12	428.929
500	7	623.779	12	635.111
600	7	924.237	12	937.389

Tabela 5.12: Problema LIARWHD: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

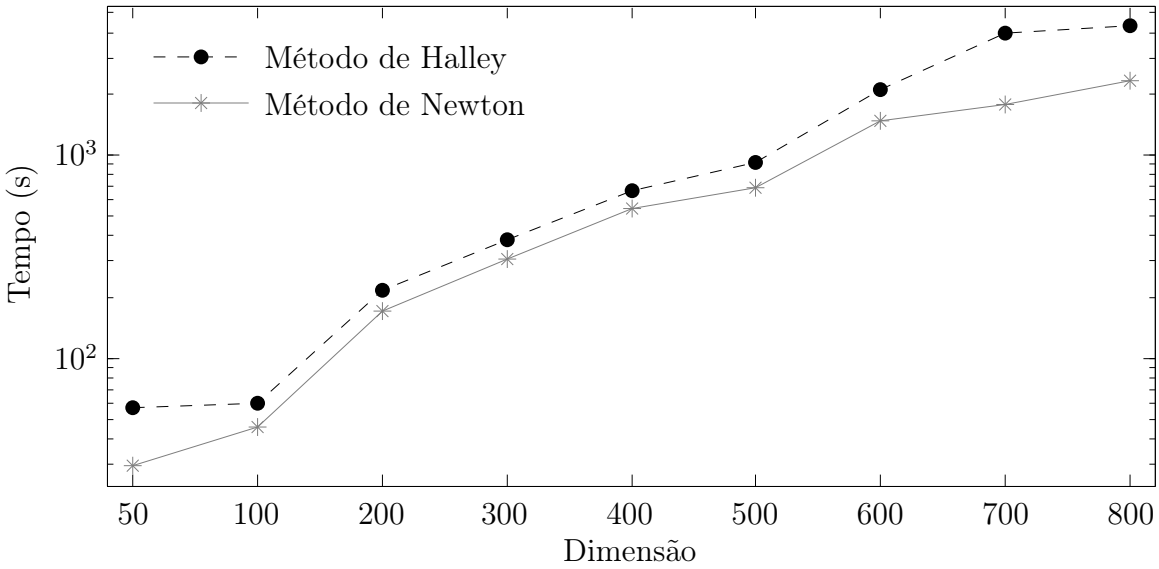
Para uma melhor comparação dos tempos de execução contidos na Tabela 5.12 apresentamos a representação gráfica dos dados na Figura 5.19. Observando o gráfico fica claro a proximidade nos tempos de execução, uma vez que as curvas que representamos tempos de execução de Newton e Halley estão praticamente sobrepostos.

Este é o segundo problema onde conseguimos um fator L, da fatoração LDL^T, mais esparso ao aplicar a estratégia de permutação das linhas e colunas da matriz, e aplicamos os métodos de Halley e Newton utilizando esta nova abordagem. Na Tabela 5.13 são apresentados os resultados obtidos para a dimensão variando entre 100 e 600. Note que não obtivemos diferença no número de iterações mas os tempos de execução são maiores. Com isso, vemos que o custo computacional necessário na permutação da matriz é maior do que o de realizar os cálculos com a matriz normal.

Com a representação gráfica dos tempos de execução da Tabela 5.13, dada em 5.20.A



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.18: Problema LIARWHD

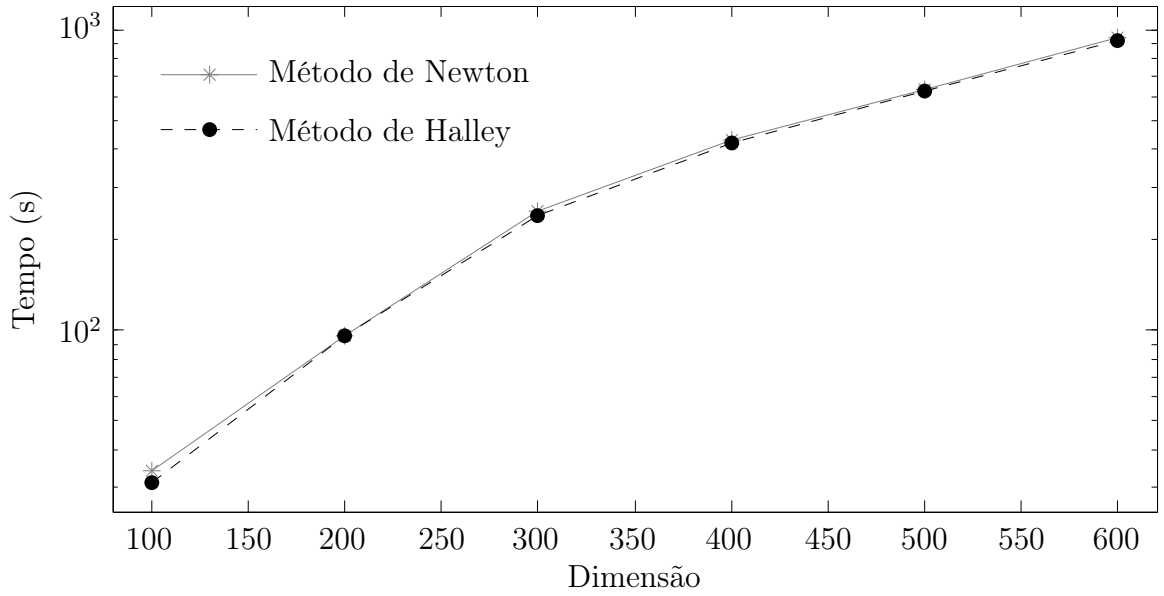


Figura 5.19: Problema LIARWHD: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	7	31.287	11	34.397
200	7	92.605	11	96.979
300	7	241.325	12	249.733
400	7	669.344	12	430.475
500	7	623.401	12	745.656
600	7	928.998	12	937.817

Tabela 5.13: Problema LIARWHD: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção e estratégia de permutação de linhas e colunas da matriz hessiana

piora nos tempos de execução é observada nos dois algoritmos, de Newton e de Halley, para 400 variáveis a piora é mais nítida no algoritmo de Halley enquanto que para 500 variáveis o tempo de Newton é maior.

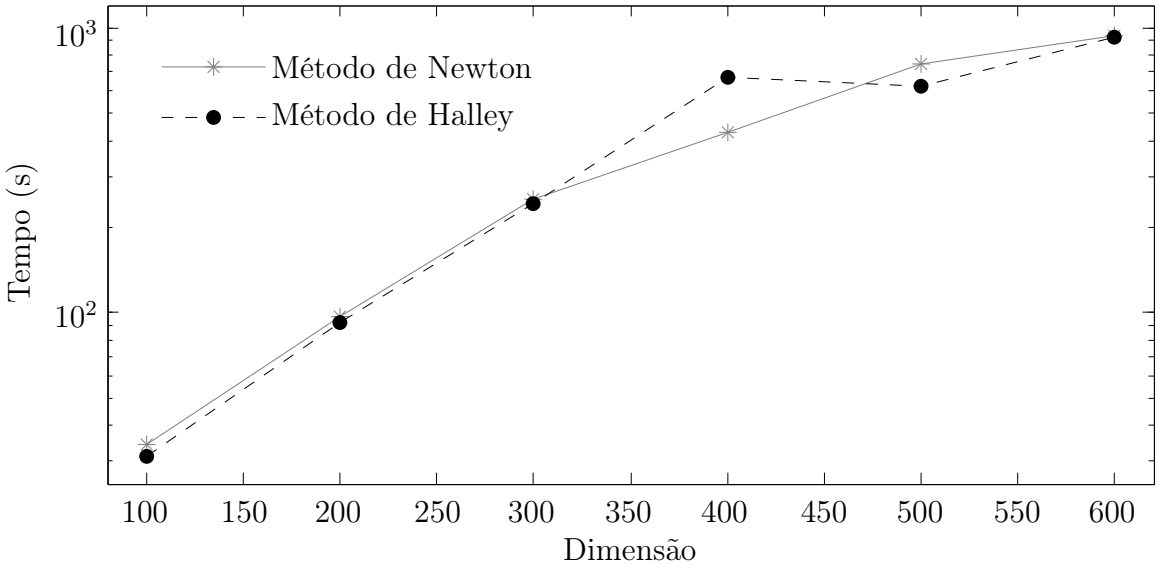


Figura 5.20: Problema LIARWHD: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção e estratégia de permutação das linhas e colunas da matriz hessiana.

5.2.7 Problema TRIDIAG1

O Problema TRIDIAG1 foi retirado da coleção de problemas [5] e sua expressão é dada por

$$f_n(x) = \sum_{i=1}^{n/2} (x_{2i-1} + x_{2i} - 3)^2 + (x_{2i-1} - x_{2i} + 1)^4,$$

com ponto inicial

$$x_0 = (2, \ldots, 2). \tag{5.7}$$

A estrutura da matriz hessiana do problema com 40 variáveis é apresentada na Figura 5.21, que como o próprio nome já diz, é uma matriz tridiagonal.

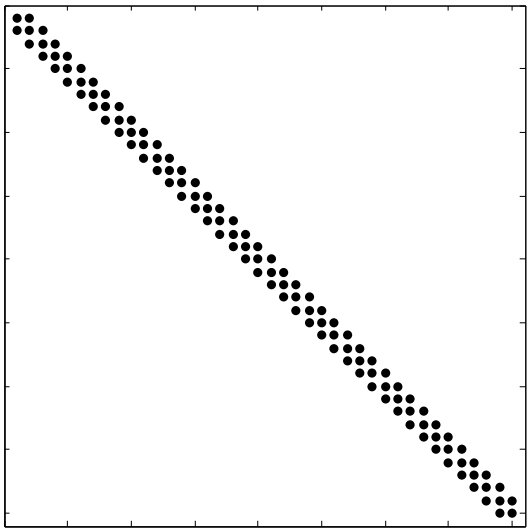


Figura 5.21: Estrutura da matriz hessiana do problema TRIDIAG1.

Foram realizados 20 testes no MATLAB com ponto inicial aleatório, variando a dimensão de 50 a 700 variáveis. A média aritmética do número de iteração e tempo de execução, em segundos, para os métodos de Newton e Halley são apresentadas na Tabela 5.14. Note que, assim como nos problema ARWHEAD e LIARWHD, este problema também apresenta um número de iterações maior para Halley, variando entre 24 e 42 iterações, enquanto que Newton só apresenta um número maior de iterações para $n = 500$, 29 iterações, para as demais dimensões apresentou 23 ou 24 iterações. Com o maior número de iterações de Halley, obviamente seu tempo de execução também foi maior para todas as dimensões, embora sua taxa de crescimento acompanhe a de Newton.

Nas Figuras 5.22 (a) e (b) apresentamos a representação gráfica dos números de iterações e tempos de execução dos métodos de Newton e Halley, para uma melhor comparação dos resultados. No gráfico da Figura 5.22 (a) podemos perceber um crescimento do número de iterações do método de Halley com o aumento da dimensão do problema, enquanto que para Newton apenas o ponto correspondente ao problema com 500 variáveis foge da curva horizontal. Com o crescimento no número de iterações esperase um crescimento ainda mais significativo no tempo de execução do total, mas como pode ser observado na

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	24	328.119	24	37.646
100	29	82.717	24	56.353
200	33	262.409	24	190.746
300	36	503.706	24	357.573
400	35	1057.908	23	732.693
500	39	1901.905	29	1337.987
600	40	6264.262	24	2724.998
700	42	10006.740	24	4200.962

Tabela 5.14: Problema TRIDIAG1: média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

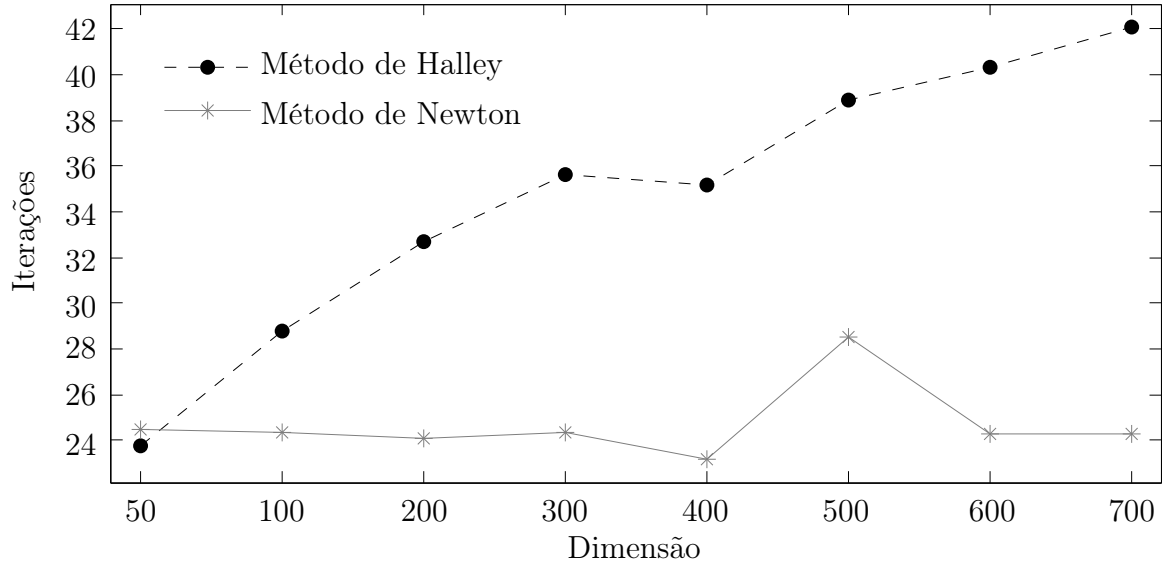
Figura 5.22 (b) o crescimento do tempo de Halley acompanha o crescimento de Newton, apresentando um crescimento um pouco maior apenas par 600 e 700 variáveis.

Nos testes apresentados na Tabela 5.15 utilizamos a aproximação inicial sugerida na coleção, dada em (5.7). Note que este caso exemplica claramente a ordem de convergência dos métodos de Newton e Halley, onde o número de iterações de Halley, constante em todas as dimensões, é 2/3 do número de iterações de Newton. Vemos que o tempo de execução de Halley também foi menor em quase todas as dimensões, apresentando tempo maior apenas para 400 e 600 variáveis, embora esta diferença seja pequena.

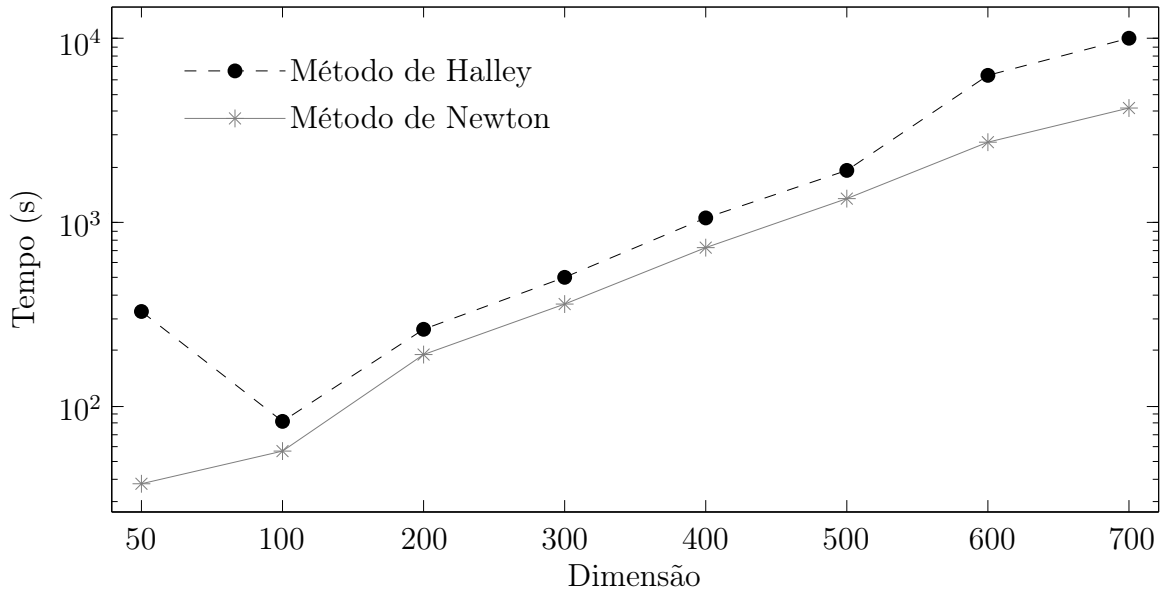
n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	3	29.413	5	31.781
200	3	88.579	5	92.983
300	3	248.661	5	246.206
400	3	454.864	5	447.295
500	3	1079.120	5	1286.110
600	3	1626.610	5	1550.510

Tabela 5.15: Problema TRIDIAG1: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

Para uma melhor visualização do tempo de execução obtido quando utilizamos o ponto inicial fornecido na coleção, apresentamos na Figura 5.23 a representação gráfica dos tempos de execução da Tabela 5.15. Note que agora as curvas dos métodos de Newton e Halley caminham juntas fungindo apenas para 500 variáveis, onde Halley apresenta um tempo de execução menor que Newton. Os resultados obtidos para Halley neste caso foram melhores porque agora conseguimos tirar proveito da principal característica que favorece



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.22: Problema TRIDIAG1.

o método de Halley, que é a ordem de convergência maior. A diferença no tempo de execução não foi muito grande pois ambos os métodos precisaram de um número pequeno de iterações para convergir.

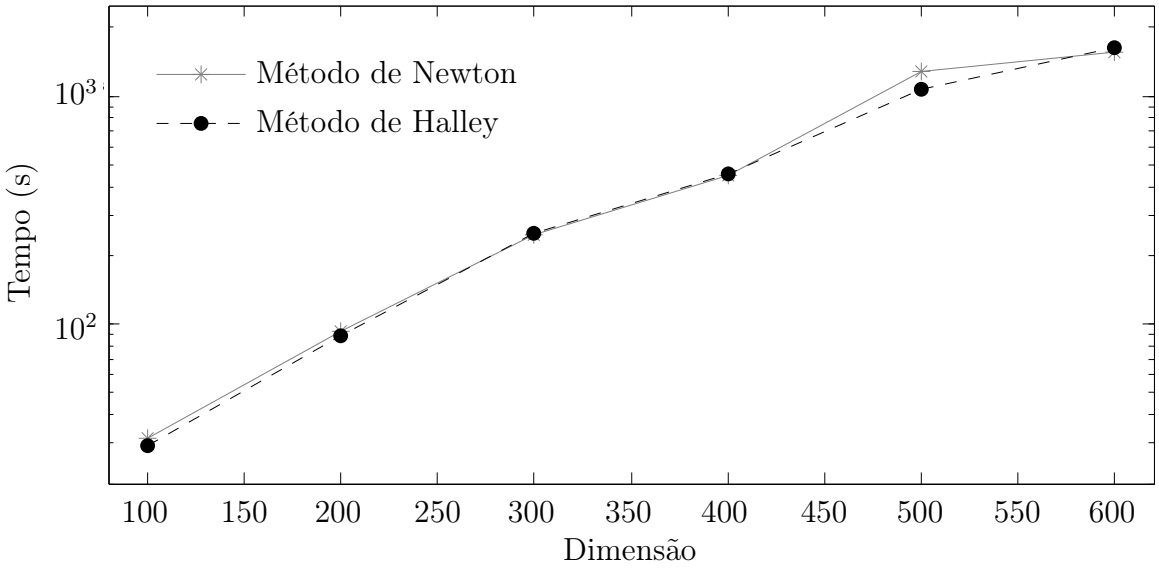


Figura 5.23: Problema TRIDIAG1: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

5.2.8 Problema QF2

O problema QF2 é dado pela expressão

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} i(x_i^2 - 1)^2 - x_n,$$

com ponto inicial

$$x_0 = (0.5, \ldots, 0.5). \tag{5.8}$$

A estrutura de esparsidade da matriz hessiana do problema QF2 com 40 variáveis é apresentado na Figura 5.24, onde podemos observar que a matriz é diagonal.

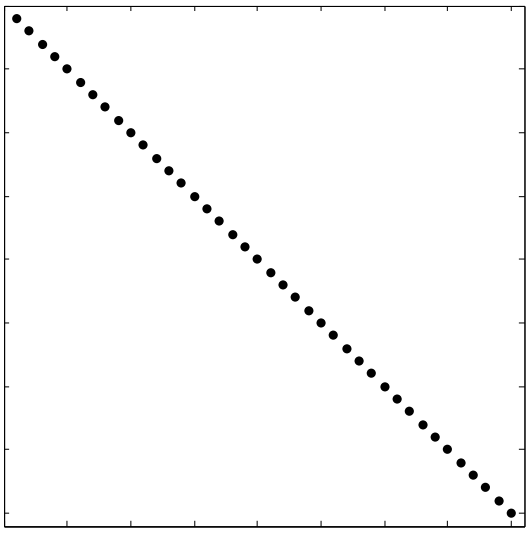
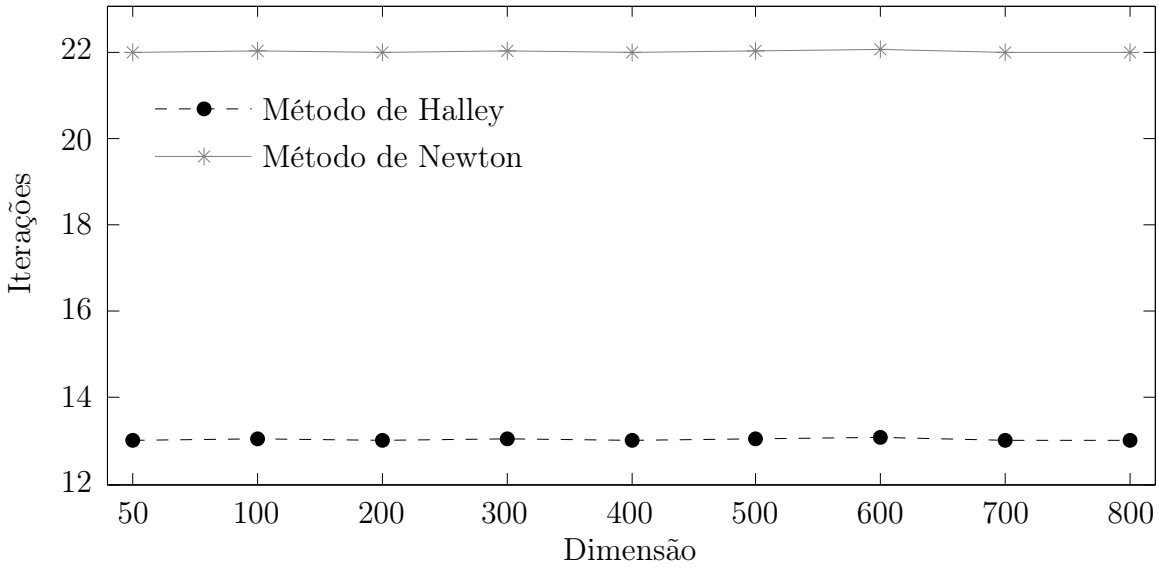


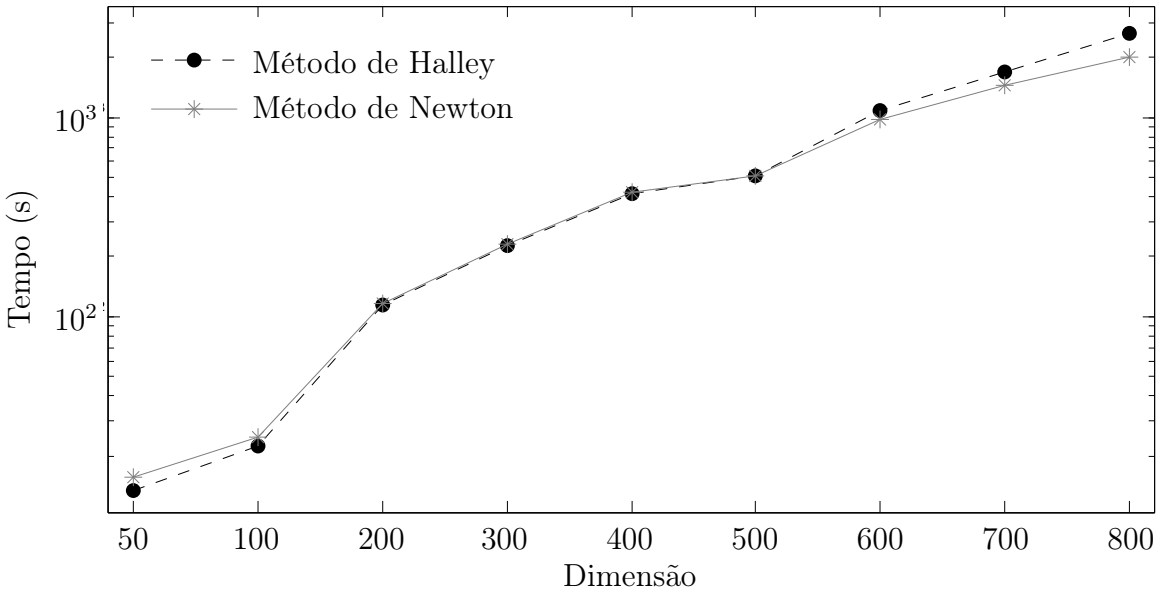
Figura 5.24: Estrutura da matriz hessiana do problema QF2.

Aplicamos os algoritmos de Newton e Halley, onde variamos a dimensão n de 50 a 800 variáveis, e para cada dimensão realizamos 20 testes com ponto inicial aleatório e apresentamos os resultados da média do número de iterações e tempo de execução em segundos, ver Tabela 5.16. Observe que para este problema o número de iterações foi constante para todas as dimensões, 13 iterações para Halley e 22 para Newton, com o número de iteração de Halley cerca de 59% do de Newto. Com essa grande diferença no número de iterações vemos que o método de Halley apresenta o menor tempo de execução na maioria dos testes, embora comece a apresentar um desempenho inferior ao de Newton com o aumento da dimensão.

A análise dos resultados obtidos fica mais fácil quando observamos a representação gráfica dos números de iteração e tempo de execução da Tabela 5.16, dada nas Figuras 5.25 (a) e (b). Na Figura 5.25 (a) observamos o número de iterações constantes para ambos os métodos representados pelas curvas paralelas ao eixo das abscissas. Já na Figura 5.25 (b) vemos que Halley apresenta uma diferença maior no tempo de execução para 50 e 100 variáveis. De 200 a 500 variáveis o tempo de Halley também é menor, mas como a



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.25: Problema QF2.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	13	13.458	22	15.517
100	13	22.520	22	24.828
200	13	115.009	22	115.784
300	13	228.327	22	231.178
400	13	414.321	22	424.537
500	13	512.555	22	507.213
600	13	1078.886	22	982.998
700	13	1698.595	22	1453.547
800	13	2630.590	22	2003.311

Tabela 5.16: Problema QF2: média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

diferença é pequena, torna-se imperceptível no gráfico. A partir de 600 variáveis o tempo de execução de Halley começa a crescer e apresenta tempos maiores que Newton.

A matriz hessiana deste problema é uma matriz diagonal, neste caso particular, tanto a hessiana quanto o tensor possuem o mesmo número de elementos não nulos, igual à dimensão do problema. Como a diferença no número de iterações é grande, era esperado que o método de Halley fosse mais eficiente.

Assim como nos outros problemas, também realizamos testes utilizando o ponto inicial fornecido na coleção, dado em 5.8. Variamos a dimensão n entre 100 e 600 e os resultados do número de iterações e tempo de execuções obtidos são apresentados na Tabela 5.17. Vemos que, apesar de Halley apresentar um número de iterações constante e menor que Newton, seu tempo de execução é maior do que o de Newton uma vez que a diferença no número de iterações é tão pequena que não chega a compensar o maior custo por iteração.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	5	57.655	8	36.847
200	5	145.473	8	102.133
300	5	339.416	8	300.035
400	5	581.452	8	562.022
500	5	906.705	8	834.470
600	5	809.442	8	806.599

Tabela 5.17: Problema QF2: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

Os resultados do tempo de execução apresentados na Tabela 5.16 podem ser analisados

melhor se considerarmos a representação gráfica dos dados, ver Figura 5.26. Observando a figura, fica claro que o método de Newton apresenta um desempenho melhor em todos os testes. Por outro lado, com o aumento da dimensão, os tempos de execução começam a ficar mais próximos, com uma diferença muito pequena para 600 variáveis, tudo indica que Halley seria mais eficiente para dimensões maiores.

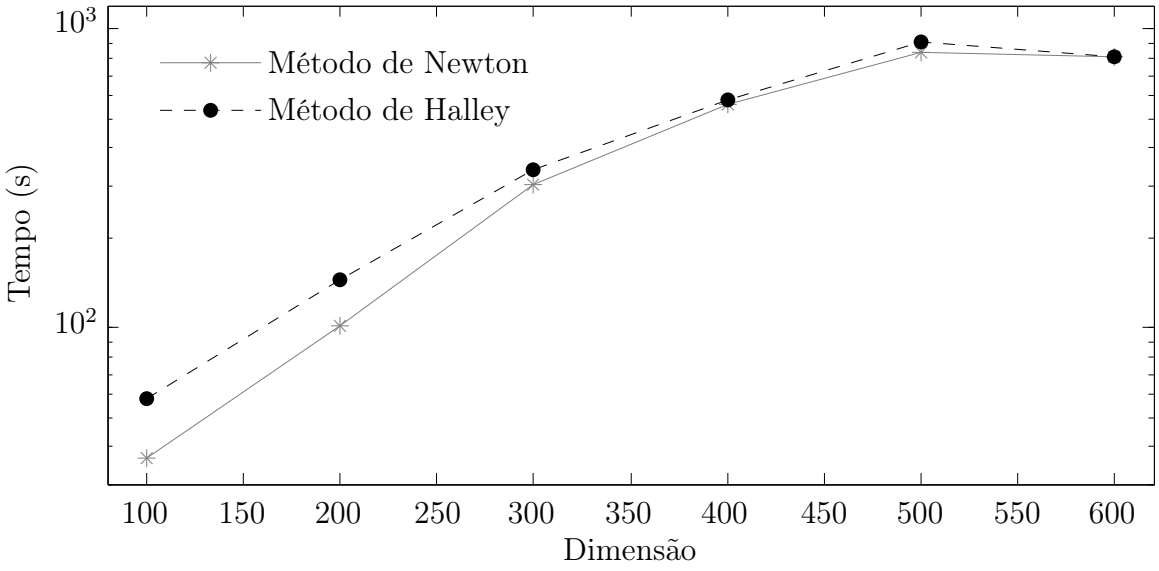


Figura 5.26: Problema QF2: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

5.2.9 Problema HIMMELBLAU

O problema HIMMELBLAU é dado pela expressão

$$f(x) = \sum_{i=1}^{n/2} (x_{2i-1}^2 + x_{2i} - 11)^2 + (x_{2i-1} + x_{2i}^2 - 7)^2,$$

$$x_0 = (1, \ldots, 1). \tag{5.9}$$

A estrutura da matriz hessiana do problema com 40 variáveis é apresentada na Figura 5.27. Vemos que os elementos não nulos estão distribuídos ao longo das três diagonais centrais.

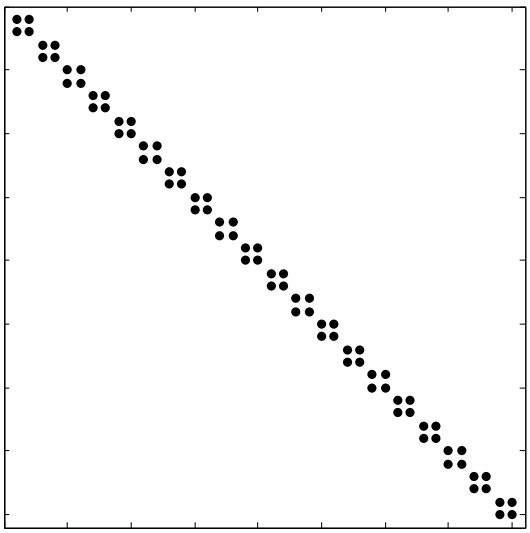
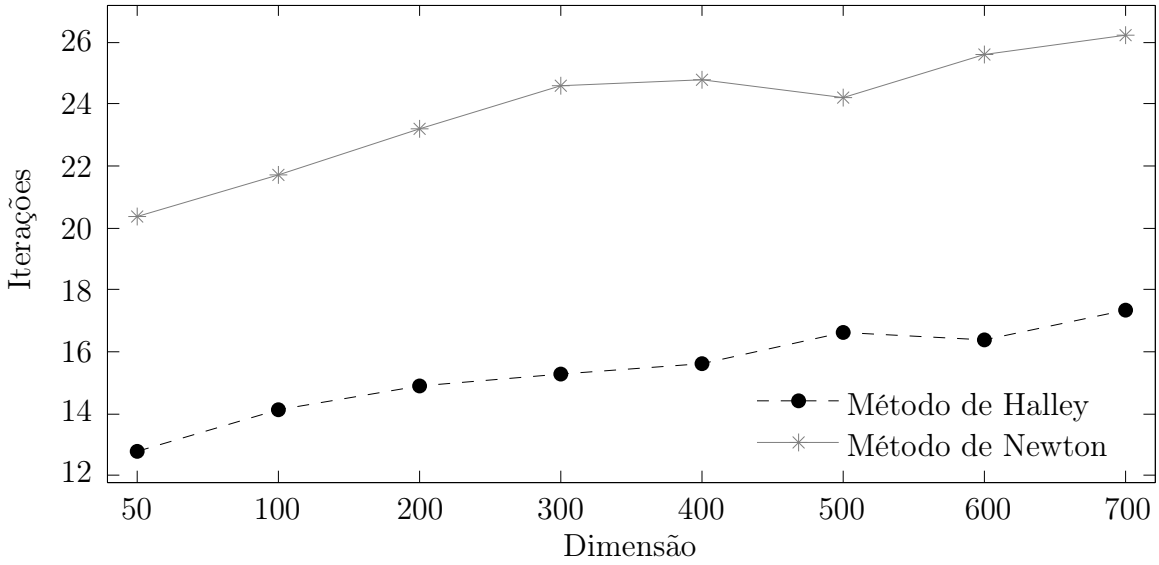


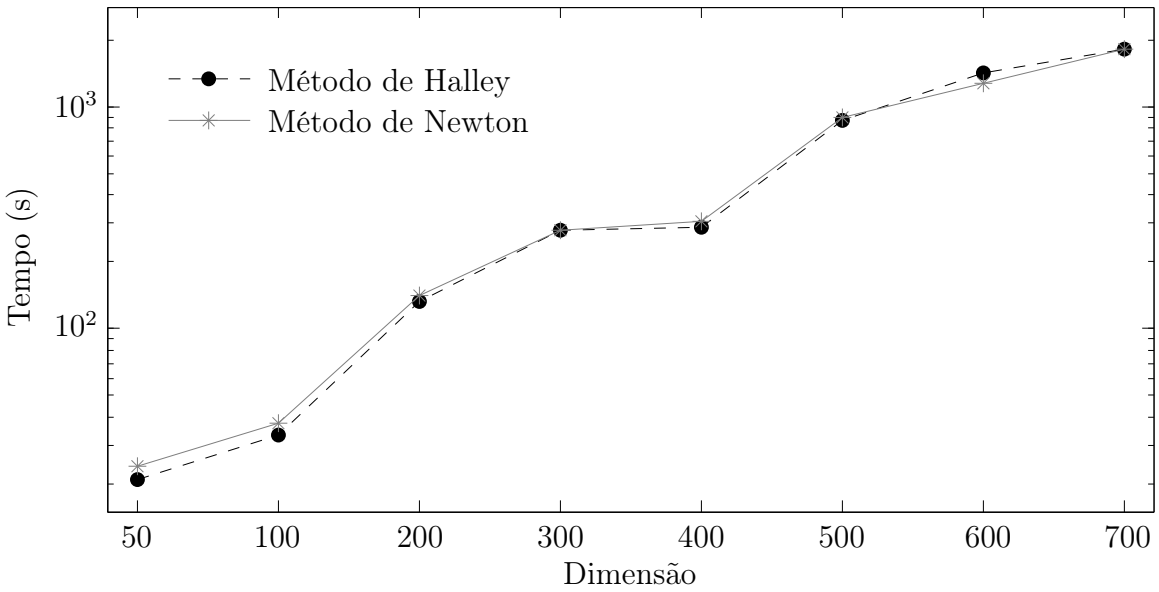
Figura 5.27: Estrutura da matriz hessiana do problema HIMMELBLAU.

Realizamos testes com os algoritmos dos métodos de Halley e Newton variando a dimensão entre 50 e 700. Para cada dimensão foram feitos 20 testes com ponto inical aleatório e apresentamos a média aritmética do número de iterações e tempo de execução na Tabela 5.18. Note que tanto para Halley quanto para Newton a média do número de iterações não apresentou muita variação, entre 13 e 17 iterações para Halley e entre 20 e 26 iterações para Newton, sendo o número de iterações de Halley aproximadamente 66 % do número de iterações de Newton. Com relação ao tempo de execução dos métodos, vemos que Halley apresentou o menor tempo de execução para todas as dimensões exceto para $n = 600$, onde Newton apresentou o menor tempo.

Nas Figuras 5.28 (a) e (b) apresentamos a representação gráfica dos resultados obtidos para o número de iterações e tempo de execução da Tabela 5.18. Na Figura 5.28 (a) podemos observar a pouca variação apresentada por ambos os métods, refletindo em curvas com apenas um leve crescimento para dimensões maiores. Já na Figura 5.28 (b) podemos observar que a curva do tempo de execução do método de Halley permaneceu sempre abaixo da curva do método de Newton, exceto para $n = 600$ o ponto da curva de



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.28: Problema HIMMELBLAU.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	13	21.033	20	24.132
100	14	33.426	22	37.460
200	15	133.405	23	142.293
300	15	276.505	25	276.505
400	16	288.060	25	306.366
500	17	872.682	24	894.287
600	16	1427.140	26	1272.063
700	17	1821.927	26	1821.927

Tabela 5.18: Problema HIMMELBLAU: média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

Halley está acima da de Newton. Mas embora Halley apresente um desempenho superior, a diferença entre os tempos de Newton e Halley ainda é pequena.

Aplicamos novamente os métodos de Newton e Halley tomando como aproximação inicial o ponto sugerido na coleção, dado em 5.9. Variamos a dimensão n de 100 a 600 variáveis e o número de iterações e tempo de execução obtidos estão apresentados na Tabela 5.19. Vemos que tanto para Newton quanto para Halley o número de iterações permaneceu constrante, mesmo com a variação da dimensão, sendo 5 iterações para Halley e 8 para Newton. Neste caso Halley também apresentou o menor tempo de execução dos algoritmos em todas as dimensões exceto para $n = 200$.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	5	27.604	8	30.041
200	5	89.276	8	88.599
300	5	221.519	8	225.285
400	5	386.538	8	391.869
500	5	575.572	8	578.781
600	5	843.518	8	848.958

Tabela 5.19: Problema HIMMELBLAU: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

Na Figura 5.29, temos a representação gráfica dos tempos de execução da Tabela 5.19 para uma melhor comparação dos tempos. Podemos observar que neste caso, a pequena diferença no tempo de execução é representada por curvas praticamente sobrepostas.

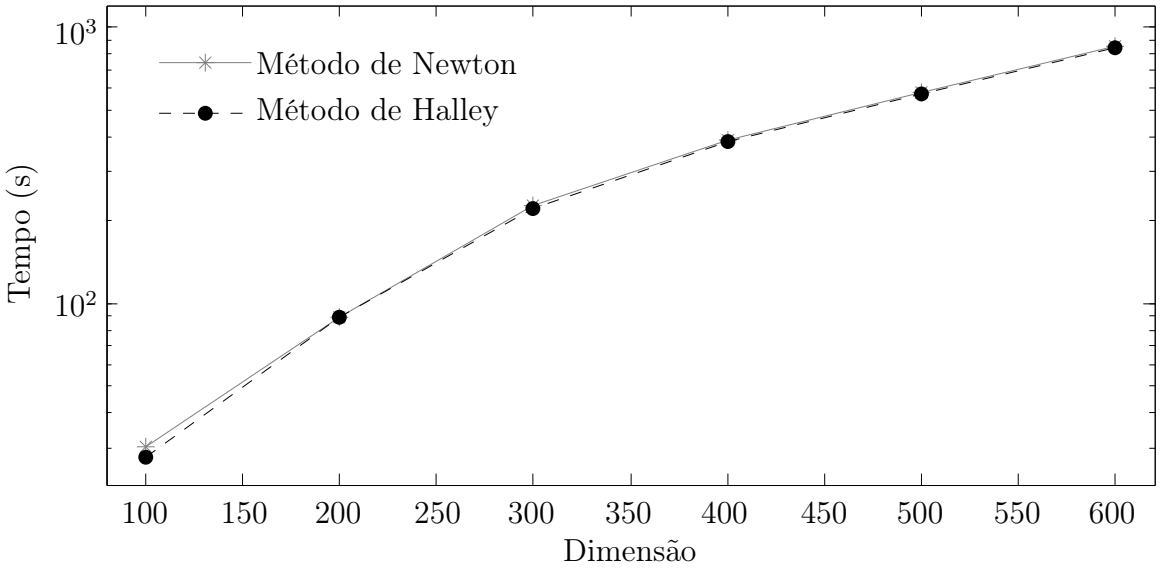


Figura 5.29: Problema HIMMELBLAU: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

5.2.10 Problema PSC1

O problema PSC1 é dado pela expressão

$$f_n(x) = \sum_{i=1}^{n-1} (x_i^2 + x_{i+1}^2 + x_i x_{i+1})^2 + \operatorname{sen}^2(x_i) + \cos^2(x_{i+1}),$$

com ponto inicial

$$x_0 = (3, 0.1, \ldots, 3, 0.1). \tag{5.10}$$

A estrutura de esparsidade da matriz hessiana do problema com 40 variáveis, está ilustrada na Figura 5.30. Note que os elementos não nulos estão distribuídos ao longo das três diagonais centrais.

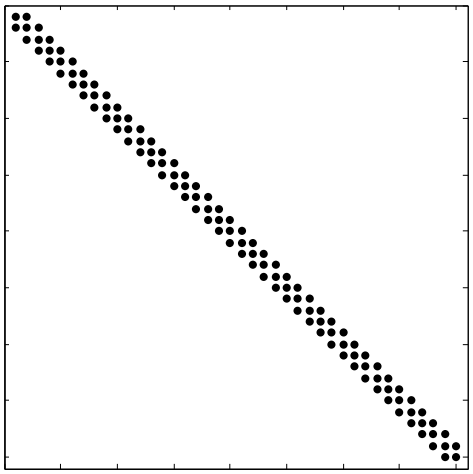


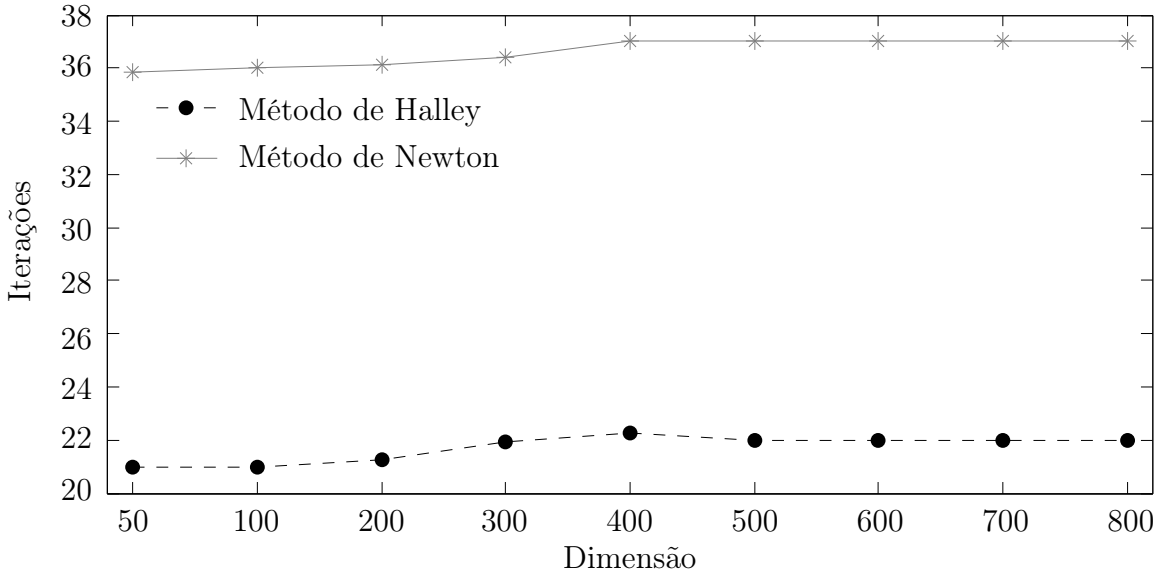
Figura 5.30: Estrutura da matriz hessiana do problema PSC1.

n	Halley		Newton	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
50	21	26.36332	36	31.39422
100	21	58.40058	36	67.43624
200	21	279.96310	36	305.95980
300	22	610.46570	36	652.64900
400	22	981.79110	37	1023.47400
500	22	1395.57650	37	1448.46900
600	22	1709.11940	37	1763.51930
700	22	2331.07050	37	2379.84500
800	22	3128.30050	37	3207.94950

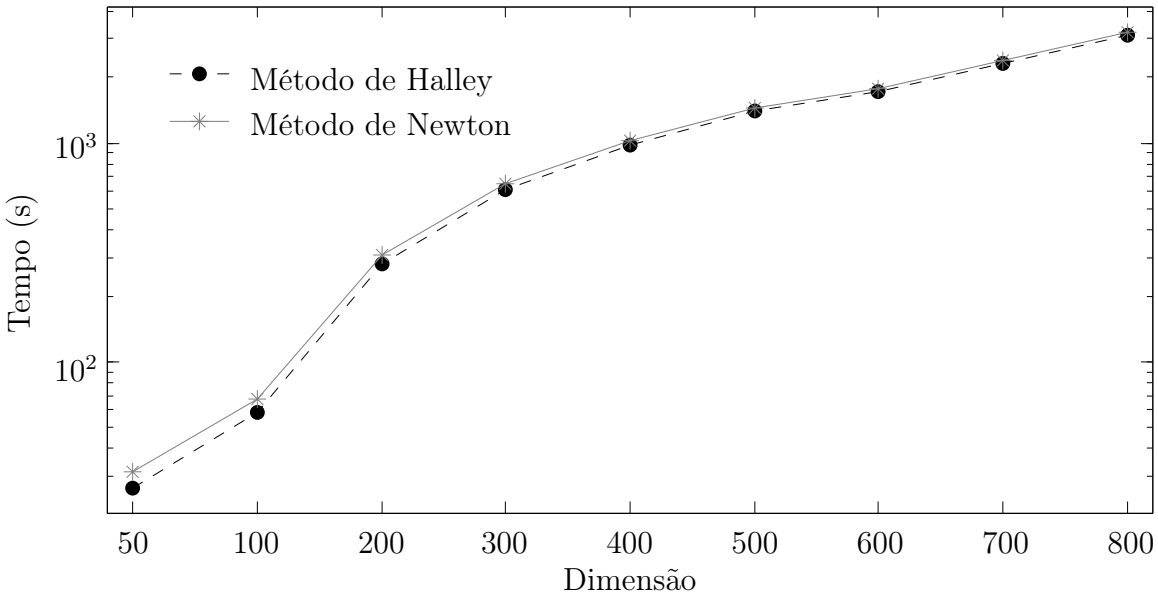
Tabela 5.20: Problema PSC1: média do número de iterações e tempo para 20 testes com ponto inicial aleatório.

Realizamos testes variando a dimensão entre 50 a 800 variáveis e para cada dimensão fizemos 20 testes com ponto inicial aleatório. Na Tabela 5.20 apresentamos a média aritmética do número de iterações e tempo de execução obtidos para estes testes. Note que para este problema o número de iterações permaneceu praticamente constante com a variação da dimensão, entre 21 e 22 iterações para Halley e entre 36 e 37 iterações para Newton. Devido à grande diferença entre os números de iterações apresentados por ambos os métodos, Halley apresenta um tempo de execução menor em todas as dimensões.

Nas Figuras 5.31 (a) e (b) apresentamos a representação gráfica dos dados da Tabela 5.20. O número de iterações praticamente constante está bem representado na Figura 5.31 (a) com curvas praticamente paralelas ao eixo das abscissas. Já na Figura 5.31 (b) podemos ver com mais clareza que Halley apresenta o menor tempo para todas as dimensões, embora as curvas estejam bem próximas para dimensões maiores.



(a): média do número de iterações dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.



(b): média do tempo de execução dos métodos de Newton e Halley para 20 testes com ponto inicial aleatório.

Figura 5.31: Problema PSC1.

Os resultados dos testes em que utilizamos o ponto inicial sugerido na coleção, dado em 5.10, estão apresentados na Tabela 5.21. Note que o número de iterações permaneceu praticamente constante com a variação da dimensão, 13 iterações no método de Halley e 22 ou 23 iterações no método de Newton, para ambos os métodos com Halley apresentando um melhor desempenho em todas as dimensões, exceto para 300 variÃ¡veis.

n	**Halley**		**Newton**	
	iteração	*tempo (s)*	*iteração*	*tempo (s)*
100	13	71.266	22	80.889
200	13	179.838	22	197.890
300	13	871.208	22	704.978
400	13	1005.200	22	1635.360
500	13	1236.420	23	1520.960
600	13	1672.200	23	2028.550

Tabela 5.21: Problema PSC1: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

Pela Tabela 5.21 vemos que o método de Halley apresentou um tempo de execução maior do que o método de Newton para a função com 300 variáveis. Com isso, obtivemos resultados melhores quando tomamos pontos iniciais aleatórios, pois nesse caso Halley convergiu em menos tempo para todas as dimensões. Esses resultados podem ser melhor comparados quando observamos sua representação gráfica, dada na Figura 5.32. Vemos que a curva dos tempos de execução de Halley permanece sempre abaixo da curva de Newton, estando acima apenas para 300 variáveis.

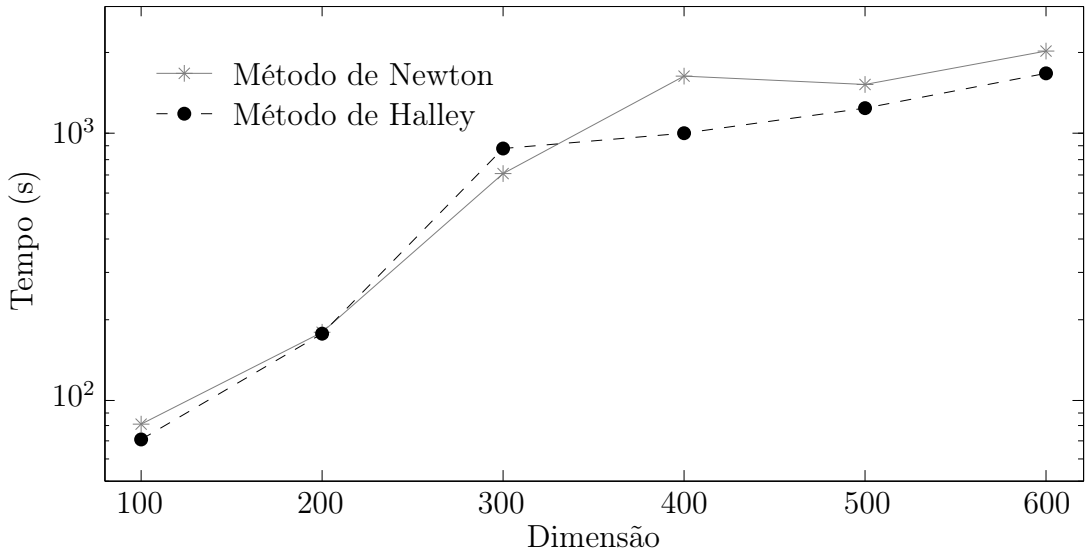


Figura 5.32: Problema PSC1: tempo de execução dos métodos de Newton e Halley com o ponto inicial fornecido na coleção.

Conclusões

Nosso foco neste trabalho foi o estudo do método de Halley para zero de função. Buscamos diferentes formas de implementá-lo, procurando torná-lo mais eficiente. Contrapomos os resultados obtidos para o método de Halley com os resultados obtidos para o método de Newton, que é o método mais utilizado na atualidade para resolver este tipo de problema.

Apesar do método de Halley apresentar uma ordem de convergência maior e, em geral, necessitar de um número menor de iterações para convergir, seu custo computacional por iteração é maior e uma análise superficial dos cálculos envolvidos o colocaria em desvantagem se comparado ao método de Newton. Para funções reais de n variáveis a complexidade computacional por iteração do método de Newton é, a princípio, $O(n^2)$ e, de Halley, $O(n^3)$. Além disso, os resultados de convergência do método de Halley são aplicáveis para funções de classe C^3 e por isso abrangem um conjunto menor de funções que os do método de Newton, que também se aplicam para funções de classe C^2.

Para realizar uma comparação empírica dos métodos, ambos foram implementados no MATLAB e escolhidos exemplares para teste. As funções de uma variável foram escolhidas em livros de Cálculo. Para os testes com funções de várias variáveis, procedemos em duas fases. Inicialmente criamos uma classe de funções com dimensão variável n e hessiana esparsa, para poder observar o comportamento do programa para dimensões arbitrárias. A análise da performance de nossa implementação nesta classe levou ao seu aperfeiçoamento. Testes posteriores foram realizados com problemas da biblioteca CUTE [7] e da coleção de problemas irrestritos de otimização de [5].

No caso de funções reais de uma variável o método de Halley mostrou-se mais rápido que o de Newton em mais de 50% dos testes realizados, um total de 600 (20 pontos iniciais diferentes para 30 funções). Além disso, apresentou maior robustez, falhando em apenas 8, em contraste com as 86 falhas observadas para o método de Newton. Em particular, encontramos duas funções para as quais Newton falhou para todos os pontos iniciais, enquanto que Halley sempre convergiu.

Nossa implementação preliminar do método de Halley indicou a inviabilidade de uma abordagem ingênua, que não explorasse simetria e esparsidade dos tensores envolvidos. Adicionalmente, nos inspiramos na expressão da derivada direcional para programar de modo mais eficiente o resultado do produto tensor $\cdot$ vetor que aparece na fórmula de Halley. A reformulação da implementação que incorporou todas estas modificações obteve um desempenho significativamente superior na classe de funções criada para a fase piloto de testes. Observamos que o número de iterações dos dois métodos, tanto Newton quanto

Halley, foi praticamente independente do valor de n, com Halley fazendo aproximadamente metade do número de iterações de Newton. Entretanto, os tempos de Halley foram superiores, embora não cheguem a ser 10% maiores que os de Newton.

Aperfeiçoada a implementação, escolhemos 6 problemas da coleção CUTE [7] e 4 problemas da coleção descrita em [5] para a fase principal de testes. Destes 10 problemas escolhidos, em 3 o número de iterações de Halley foi maior do que de Newton para pontos iniciais aleatórios. Nestes casos, Newton apresentou pouca variabilidade no número de iterações, enquanto Halley variou significativamente. Os tempos de Halley foram consideravelmente maiores que os de Newton, com a diferença aumentando com a dimensão do problema.

Para 4 problemas observamos que ambos os métodos necessitavam, em média, do mesmo número de iterações para convergir, mesmo com o aumento da dimensão do problema. Para estes problemas, o método de Halley apresentou menor tempo de execução em quase todas as dimensões.

Nos demais problemas, em geral, o método de Halley apresentou um número menor de iterações que o método de Newton, com tempos de execução bem próximos, em grande parte dos casos menores que os de Newton.

Efetuamos também testes com os pontos iniciais fornecidos nas respectivas coleções. Para estes pontos, Halley convergiu sempre em um número menor de iterações, e, para a maioria das funções, em tempo menor também. Estes testes exemplificam com perfeição as respectivas taxas de convergência dos métodos, com o número de iterações de Halley aproximadamente igual a $2/3$ do número de iterações de Newton.

A estratégia de permutação das linhas e colunas da matriz hessiana, para que ela tenha um fator mais esparso, quando obtermos a fatoração LDL^T da matriz, foi empregada para 2 funções. Apesar disso, os resultados permaneceram praticamente os mesmos, sendo observada uma pequena melhora apenas para o problema DIXMAANE.

Por fim, é importante destacar que nossos testes estão condicionados ao uso do ambiente do MATLAB e tudo indica que os resultados poderiam ser melhores se estivéssemos trabalhando em uma linguagem de programação dedicada.

Referências Bibliográficas

[1] P. J. Acklam. A small paper on Halley's method. `http://home.online.no/~pjacklam/notes/halley/halley.pdf`, 2002. Acesso em 11 de março de 2013.

[2] G. Alefeld. On the convergence of Halley's method. *Amer. Math. Monthly*, 88(7):530–536, 1981.

[3] M. Altman. Iterative methods of higher order. *Bull. Acad. Polon. Sci. Sér. Sci. Math. Astr. Phys.*, 9(2):63–68, 1961.

[4] M. Altman. Concerning the method of tangent hyperboles for operator equations. *Bull. Acad. Polon. Sci. Sér. Sci. Math. Astr. Phys.*, 9(9):633–637, 1961.

[5] N. Andrei. An unconstrained optimization test functions collection. *Adv. Model. Optim.*, 10(1):147–161, 2008.

[6] M. S. Bazaraa, H. D. Sherali e C. M. Shetty. *Nonlinear Programming, Theory and Applications*. John Wiley & Sons, New York, 2 ed., 1993.

[7] I. Bongartz, A. R. Conn, N. Gould e Ph. L. Toint. Cute: constrained and unconstrained testing environment. *ACM Trans. Math. Softw.*, 21(1):123–160, 1995.

[8] G. H. Brown Jr. On Halley's variation of Newton's method. *American Mathematical Monthly*, 84(9):726–728, 1977.

[9] R. L. Burden, J. Douglas Faires e Albert C. Reynolds. *Numerical analysis*. Prindle, Weber & Schmidt, Boston, Mass., 1978.

[10] V. Candela e A. Marquina. Recurrence relations for rational cubic methods: I. The Halley method. *Computing*, 44(2):169–184, 1990.

[11] S. D. Conte. Elementos de Análise Numérica. *Enciclopédia Técnica Universal Globo*, 1977.

[12] J. A. Ezquerro e M. A. Hernández. New Kantorovich-type conditions for Halley's method. *Appl. Numer. Anal. Comput. Math.*, 2(1):70–77, 2005.

[13] A. Galántai. The theory of Newton's method. *J. Comput. Appl. Math.*, 124(1-2):25–44, 2000.

[14] W. Gander. On Halley's iteration method. *Amer. Math. Monthly*, 92(2):131–134, 1985.

[15] H. L. Guidorizzi. Um curso de cálculo. *LTC - Livros Técnicos e Científicos Editora S.A.*, 5 ed., 1, 2001.

[16] G. Gundersen e T. Steighaus. Sparsity in higher order methods in optimization. Technical Report 327, Departamento de Informática, Universidade de Bergen, 2006.

[17] G. Gundersen e T. Steighaus. Halley and Newton are one step apart. *Proc. Appl. Math. Mech.*, 7(1):2060011–2060012, 2007.

[18] G. Gundersen e T. Steighaus. On large-scale unconstrained optimization problems and higher order methods. *Optimization Methods & Software*, 25(3):337 – 358, 2010.

[19] Dan-fu Han. The convergence on a family of iterations with cubic order. *J. Comp. Math.*, 19(5):467–474, 2001.

[20] L. V. Kantorovich. The method of successive approximations for functional analysis. *Acta Math.*, 71:63–97, 1939.

[21] L. V. Kantorovich. Functional analysis and applied mathematics. *Uspehi Matem. Nauk (N.S.)*, 3(6(28)):89–185, 1948.

[22] L. V. Kantorovich. On Newton's method for functional equations. In *Dokl. Akad. Nauk SSSR*, 59:1237–1240, 1948.

[23] L. V. Kantorovich. On Newton's method. *Trudy Mat. Inst. Steklov*, 28:104–144, 1949.

[24] L. V. Kantorovich. The majorant principle and Newton's method. In *Dokl. Akad. Nauk SSSR (NS)*, 76:17–20, 1951.

[25] L. V. Kantorovich. On some further applications of the Newton approximation method. *Vestnik Leningrad University. Mathematics*, 12(7):68–103, 1957.

[26] Y. Ling e X. Xu. Semilocal convergence behavior of Halley's method using Kantorovich's majorants principle. arXiv:1212.0719v1 [math.NA], 2012.

[27] A. Melman. Geometry and convergence of Euler's and Halley's methods. *SIAM Rev.*, 39(4):728–735, 1997.

[28] G. Miel. An updated version of the Kantorovich theorem for Newton's method. *Computing*, 27(3):237–244, 1981.

[29] J. M. Ortega e W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics, Philadephia, PA, 2000.

[30] R. A. Safiev. The method of tangent hyperbolas. In *Sov. Math. Dokl*, 4:482–485, 1963.

[31] T. R. Scavo e J. B. Thoo. On the geometry of Halley's method. *Amer. Math. Monthly*, 102(5):417–426, 1995.

[32] T. Steihaug e S. Suleiman. Rate of convergence of higher order methods. *Applied Numerical Mathematics*, 67:230–242, 2013.

[33] J. Stewart. Calculus early transcendentals. *Thomson Brooks/Cole*, 6 ed., 2008.

[34] S. T. M. Suleiman. Solving systems of nonlinear equations using methods in the Halley class. *Thesis for the degree of Master of Science.* Departamento de Informática, Universidade de Bergen, 2009.

[35] E. Süli e D. F. Mayers. An Introduction to Numerical Analysis. *Cambridge University Press*, 2003.

[36] T. Yamamoto. On the method of tangent hyperbolas in Banach spaces. *J. Comput. Appl. Math.*, 21(1):75–86, 1988.

[37] T. Yamamoto. Historical developments in convergence analysis for Newton's and Newton-like methods. *J. Comput. Appl. Math.*, 124(1-2):1–23, 2000.

[38] Wikipedia. *Edmond Halley*, `http://en.wikipedia.org/w/index.php?title=Edmond_Halley&oldid=540331098` Acesso em 11 de março de 2013.