Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

O uso da estratégia de aceleração em métodos sem derivadas para a resolução de sistemas não lineares

Autor: Sérgio de Almeida Cipriano Júnior
Orientador: Prof. Dr. John Lenon Cardoso Gardenghi

Brasília, DF
2023

Sérgio de Almeida Cipriano Júnior

O uso da estratégia de aceleração em métodos sem derivadas para a resolução de sistemas não lineares

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. John Lenon Cardoso Gardenghi

Brasília, DF

2023

Sérgio de Almeida Cipriano Júnior
O uso da estratégia de aceleração em métodos sem derivadas para a resolução de sistemas não lineares/ Sérgio de Almeida Cipriano Júnior. – Brasília, DF, 2023-
42 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. John Lenon Cardoso Gardenghi

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2023.

1. Otimização-sem-derivadas. 2. Sistemas-de-equações-não-lineares. I. Prof. Dr. John Lenon Cardoso Gardenghi. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. O uso da estratégia de aceleração em métodos sem derivadas para a resolução de sistemas não lineares

Sérgio de Almeida Cipriano Júnior

O uso da estratégia de aceleração em métodos sem derivadas para a resolução de sistemas não lineares

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 11 de dezembro de 2023 – Data da aprovação do trabalho:

Prof. Dr. John Lenon Cardoso Gardenghi
Orientador

Prof. Dr. Bruno César Ribas
Convidado 1

Prof. Dr. Edson Alves da Costa Júnior
Convidado 2

Brasília, DF
2023

Resumo

A resolução de um sistema de equações é um problema usual em disciplinas quantitativas e consiste em encontrar uma valoração para as grandezas que descrevem o comportamento desse sistema. Nesse sentido, em métodos para resolução de sistemas de equações é comum o cálculo de derivadas, especialmente em se tratando de equações não lineares, o que, para muitos cenários, é uma abordagem eficiente. Contudo, para solucionar sistemas de equações não lineares com muitas variáveis, o cálculo de derivadas pode ser inviável por possuir um alto custo computacional ou consumo de memória. Neste trabalho, para solucionar sistemas não lineares com muitas variáveis, emprega-se um método residual espectral que não faz o uso do cálculo de derivadas. O objetivo é empregar uma nova técnica de aceleração, proposta por E. G. Birgin e J. M. Martínez, *Secant Acceleration of Sequential Residual Methods for Solving Large-Scale Nonlinear Systems of Equations, SIAM Journal on Numerical Analysis*, 2022, numa implementação consolidada em R do método residual espectral, verificando a eficácia da implementação em comparação com a mesma versão do algoritmo sem aceleração.

Palavras-chave: otimização sem derivadas, sistemas de equações não lineares, aceleração, método residual espectral.

Abstract

Solving a system of equations is a common problem in decision science and in the analysis of physical systems. It consists of finding a quantitative measure that describes the behavior of the given system. In this sense, it is common to calculate derivatives for solving systems of equations, especially when it comes to nonlinear equations which, for many scenarios, is a efficient approach. However, to solve systems of nonlinear equations with many variables, it may be impracticable to make use of derivatives since it has a high computational cost or memory consumption. In this work, to solve nonlinear systems with many variables, a spectral residual method is employed that does not make use of derivatives. The goal is to employ a new acceleration technique, proposed by E. G. Birgin and J. M. Martínez, Secant Acceleration of Sequential Residual Methods for Solving Large-Scale Nonlinear Systems of Equations, SIAM Journal on Numerical Analysis, 2022, in a consolidated R implementation of the method spectral residual, checking the effectiveness of the implementation compared to the same version of the algorithm without acceleration.

Keywords: derivative-free optimization, systems of nonlinear equations, acceleration, spectral residual methods.

Sumário

1 INTRODUÇÃO 11
1.1 Justificativa 11
1.2 Objetivos 12
1.3 Estrutura do Documento 12

2 FUNDAMENTAÇÃO TEÓRICA 13
2.1 Fundamentos da Otimização 13
2.2 Métodos de Otimização 15
2.3 Método Residual Espectral sem Derivadas (DF-SANE) 16
2.4 Implementação computacional do DF-SANE no BB 19
2.5 Estratégia de aceleração 19

3 PROPOSTA METODOLÓGICA 21
3.1 Aplicação da estratégia de aceleração 21
3.2 Abordagem da Experimentação Numérica 23

4 EXPERIMENTAÇÃO NUMÉRICA 27
4.1 Resultados obtidos 27
4.2 Análise de resultados 34

5 CONSIDERAÇÕES FINAIS 39
5.1 Trabalhos futuros 39

REFERÊNCIAS 41

1 Introdução

Otimização matemática consiste em encontrar a melhor solução possível dentro de um conjunto de opções, seja minimizando ou maximizando uma quantidade específica, encontrando os valores ótimos de um conjunto de variáveis ou atendendo a um série de restrições (GILL; MURRAY; WRIGHT, 1981).

Por consequência, otimização é amplamente utilizada em uma variedade de campos, como ciência da computação, engenharia, economia e ciências sociais; em virtude de ser aplicável para resolver problemas práticos, como planejar rotas de entrega eficientes, minimizar o custo de produção de um produto ou encontrar a combinação de investimentos que maximiza o retorno (NOCEDAL; WRIGHT, 2006).

Em síntese, um problema de otimização consiste em medidas quantitativas que descrevem o comportamento de um sistema e variáveis que caracterizam as particularidades desse sistema. O objetivo é encontrar valores para essas variáveis de forma a otimizar o sistema (DENNIS; SCHNABEL, 1996).

1.1 Justificativa

Métodos para otimização funcionam com base em dois pilares principais: direção e passo, que é o quanto percorrer na direção definida. O uso de derivadas para calcular essa direção é muito comum, visto que o oposto do gradiente é uma direção de descida. No entanto, algoritmos sem uso de derivadas são relevantes, haja vista que nem toda modelagem computacional do problema fornece meios de se calcular derivadas. O algoritmo empregado nesse trabalho, *Derivative-Free Spectral Algorithm for Nonlinear Equations* (DF-SANE) (CRUZ; MARTÍNEZ; RAYDAN, 2006) não faz o uso de derivadas e é utilizado para solucionar problemas de equações não lineares de grande porte.

Um dos procedimentos mais comuns para calcular o passo α_k é chamado de busca linear. Propostas de melhorias ao DF-SANE envolvendo mudanças na busca linear já foram apresentadas (CRUZ, 2017; MELI et al., 2020). Isso porque, tal como exposto por Birgin e Martínez (2022), o algoritmo DF-SANE sem modificações pode ser ineficiente para problemas de grande porte nos quais é necessário retroceder muitas vezes na busca linear para obter uma descida suficiente, isto é, uma redução no valor da função a ser minimizada. Assim, novas abordagens para escolher o passo e estratégias de aceleração são desejadas.

Varadhan e Gilbert (2009) apresentaram o pacote BB[1] implementado em R, que

[1] https://cran.r-project.org/web/packages/BB/index.html

disponibiliza funções para solucionar sistemas de equações não lineares. Esse pacote inclui uma implementação do DF-SANE, baseada no código Fortran de Birgin, Martínez e Raydan (2001), que será utilizada como base para a implementação do DF-SANE apresentada neste trabalho.

Dado o contexto, na próxima seção é definido o objetivo deste trabalho.

1.2 Objetivos

O objetivo deste trabalho é aplicar a estratégia de aceleração apresentada por Birgin e Martínez (2022) no método residual spectral DF-SANE implementado em R por Varadhan e Gilbert (2009).

Para atender o objetivo geral foram definidos os seguintes objetivos específicos:

- **OE1:** Aplicar a estratégia de aceleração de Birgin e Martínez na implementação do DF-SANE feita por Varadhan e Gilbert;
- **OE2:** Realizar experimentos numéricos comparando a versão modificada com a implementação do Varadhan e Gilbert.

1.3 Estrutura do Documento

O presente trabalho foi estruturado em capítulos: neste capítulo foi introduzido o trabalho e definido o objetivo; no Capítulo 2 são apresentados os trabalhos relacionados e o algoritmo proposto; no Capítulo 3 é abordada a proposta metodológica a ser empregada para o alcance dos objetivos definidos, com um detalhamento da experimentação numérica; no Capítulo 4 é mostrado os resultados obtidos nos experimentos e a análise desses resultados; no Capítulo 5 são feitas as considerações finais e ideias de trabalhos futuros são apresentadas.

2 Fundamentação Teórica

Neste capítulo, são apresentados os conceitos fundamentais para o embasamento deste trabalho. Para isso, são explorados os trabalhos mais avançados e atualizados sobre o algoritmo DF-SANE, bem como literaturas clássicas de otimização matemática.

O capítulo foi dividido do seguinte modo: a Seção 2.1 apresenta o que é um problema de otimização e os pilares que caracterizam um método de otimização; a Seção 2.2 traz métodos clássicos de otimização para contextualizar a importância de métodos como o DF-SANE; na Seção 2.3 é apresentado o método DF-SANE e algumas de suas diferentes versões; na Seção 2.4 é apresentado como Varadhan e Gilbert (2009) implementaram o algoritmo DF-SANE; a Seção 2.5 traz a estratégia de aceleração proposta por Birgin e Martínez (2022) e como é integrada ao algoritmo DF-SANE.

Notação. O símbolo $\|\|\|$ denota a norma euclidiana; $J(x)$ representa a matriz jacobiana de $F : \mathbb{R}^n \to \mathbb{R}^n$ avaliada em x; ∇f é o gradiente de f; $\nabla^2 f$ é a hessiana de f; A^T representa a matriz transposta de A; x_k^i representa a i-ésima componente de x elevada a k-ésima potência; o negrito em escalares como $\mathbf{0}$ indica que $\mathbf{0} \in \mathbb{R}^n$; $\max(a, b)$ é o valor máximo entre a e b; $\min(a, b)$ é o valor mínimo entre a e b; e $|a|$ refere-se ao valor absoluto de a.

2.1 Fundamentos da Otimização

Matematicamente falando, dado uma função objetivo f, otimizar essa função significa escolher valores da variável x com a finalidade de minimizar a função objetivo (GILL; MURRAY; WRIGHT, 1981).

É possível utilizar a otimização matemática para encontrar o ponto ótimo em um sistema não linear. E, dado que muitos problemas do cotidiano podem ser representados como sistemas de equações não lineares com muitas variáveis, a otimização é uma ferramenta valiosa na solução de problemas deste tipo. Assim, neste trabalho serão considerados sistemas de equações não lineares na forma:

$$F(x_*) = \mathbf{0}, \tag{2.1}$$

sendo $F : \mathbb{R}^n \to \mathbb{R}^n$ continuamente diferenciável em $\mathbb{R}^n$ e $x_* \in \mathbb{R}^n$. O interesse deste trabalho está em solucionar problemas de grande porte para os quais a Jacobiana de F não é calculável ou tem um custo computacionalmente alto (BIRGIN; MARTíNEZ, 2022). A Jacobiana é considerada não calculável quando a modelagem do problema de otimização não fornece os meios necessários para realizar o seu cálculo, esses problemas são classificados como caixa preta.

Conforme apresentado por Dennis e Schnabel (1996) a Equação (2.1) é apenas uma das formas de se representar um sistema de n equações não lineares com n variáveis. Um exemplo é

$$F(x_1, x_2) = \begin{bmatrix} x_1^2 + x_2^3 + 7 \\ x_1 + x_2 + 1 \end{bmatrix}, \tag{2.2}$$

que tem $F(x_*) = \mathbf{0}$ para $x_* = (1, -2)^T$, visto que o mesmo x_* que resolve (2.2) é também minimizador de

$$\sum_{i=1}^{n} f_i^2(x), \tag{2.3}$$

onde $f_i(x)$ denota a i-ésima componente da função F. A Equação (2.3) é classificada como um problema de minimização sem restrições, descrito como

$$\begin{aligned} &\text{Dado } f : \mathbb{R}^n \to \mathbb{R}, \\ &\text{encontrar } x_* \in \mathbb{R}^n \text{ para } f(x_*) \leq f(x) \text{ para todo } x \in \mathbb{R}^n, \end{aligned}$$

ou

$$\min_{x \in \mathbb{R}^n} f(x) : \sum_{i=1}^{n} f_i^2(x), \tag{2.4}$$

que normalmente é apenas abreviada para

$$\min_{x \in \mathbb{R}^n} f(x) : \mathbb{R}^n \to \mathbb{R}.$$

Um exemplo desse problema é

$$\min_{x \in \mathbb{R}^n} f(x_1, x_2, x_3) = (x_1 - 3)^2 + (x_2 + 5)^2 + (x_3 - 8)^2,$$

que tem como solução

$$x_* = \begin{bmatrix} 3 \\ -5 \\ 8 \end{bmatrix}.$$

Tal como descrito por Nocedal e Wright (2006), métodos para otimização numérica funcionam na base de iterações. Assim, a cada iteração k do algoritmo é calculado uma direção d_k e um passo α_k, que são usados para atualizar o valor das variáveis x_k. Desse modo, na primeira iteração é definido um x_0 arbitrário e o objetivo é executar k iterações gerando uma sequência $x_1, x_2, ..., x_k$ tal que

$$x_{k+1} = x_k + \alpha_k d_k \tag{2.5}$$

até encontrar um $x_* = x_k$ satisfatório, sendo $\alpha_k \in \mathbb{R}$ o quanto percorrer na direção $d_k \in \mathbb{R}^n$. A definição de satisfatório varia de acordo com o critério de parada do algoritmo.

A estratégia utilizada para definir a direção e o passo é o que difere os métodos de otimização. Muitas abordagens fazem uso da primeira e segunda derivada da função objetivo f, visto que oferecem informações sobre o declive e a curvatura de f em um ponto

específico (DENNIS; SCHNABEL, 1996). O algoritmo empregado neste trabalho, DF-SANE, utiliza resíduos, que são valores de variáveis e do sistema, calculados em iterações anteriores, para construir a sua direção de busca sem a necessidade de efetuar avaliações de derivadas.

2.2 Métodos de Otimização

Inúmeros problemas do cotidiano podem ser representados como sistemas de equações não lineares de grande porte. A escolha clássica para solucionar esses problemas é o método de Newton-Raphson, que demanda a resolução de um sistema linear em cada iteração do algoritmo principal (BIRGIN; MARTíNEZ, 2022).

Tal como apresentado por Dennis e Schnabel (1996), a direção do método de Newton é proveniente da derivada da aproximação de segunda ordem da série de Taylor para $f(x_k + d_k)$, que é

$$f(x_k + d_k) \approx f_k + d_k^T \nabla f_k + \frac{1}{2} d_k^T \nabla^2 f_k d_k = m_k(d_k). \tag{2.6}$$

No caso da hessiana $\nabla^2 f_k$ ser uma matriz positiva definida, a direção de Newton é obtida ao encontrar o vetor d que minimiza o modelo quadrático $m_k(d_k)$. De outro modo, almeja-se definir a direção d de modo que a derivada direcional de $m_k(d_k)$ seja nula, isto é,

$$d_k = -(\nabla^2 f_k)^{-1} \nabla f_k. \tag{2.7}$$

Quando $\nabla^2 f_k$ não é positiva definida, não há garantias de que a direção de Newton existe, em virtude de a inversa $(\nabla^2 f_k)^{-1}$ poder não existir. Para lidar com cenários onde a direção de Newton não pode ser obtida ou quando não corresponde à uma direção de descida, alguns métodos modificam a definição de d_k enquanto retém o benefício de informações de segunda ordem.

Nesse seguimento, Gill, Murray e Wright (1981) apresentam como calcular uma hessiana aproximada por meio da fatoração de Cholesky ou LDL^T. Em contrapartida, métodos diretos como Cholesky, LU e QR apresentam alto consumo de memória quando a matriz do sistema é de várias variáveis, pois calculam a decomposição de uma matriz.

Outros métodos, como os Quasi-Newton, buscam solucionar o sistema linear newtoniano por aproximação, como o GMRES (*generalized minimal residual algorithm for solving nonsymmctric linear systems*), com o potencial de realizar muitos produtos matriz-vetor por iteração. Esses produtos matriz-vetor na forma $J(x)v$, sendo a Jacobiana $J(x)$ de f e $v \in \mathbb{R}^n$, são normalmente substituídos pelo quociente incremental $[f(x + hv) - f(x)]/h$, um procedimento que não deteriora a performance de tais métodos. Entretanto, quando GMRES realiza muitos produtos para encontrar uma aproximação que solucione o sistema linear Newtoniano, o número de resíduos (usados para determinar se a

aproximação da atual iteração satisfaz o sistema linear) necessários aumenta bastante (BIRGIN; MARTíNEZ, 2022).

Com isso, foram introduzidos algoritmos cuja quantidade de avaliações de resíduos, o consumo de memória para armazenar as direções e o custo computacional nas operações algébricas fosse mínimo. O DF-SANE, então, foi introduzido como alternativa para a resolução desses problemas de várias variáveis.

2.3 Método Residual Espectral sem Derivadas (DF-SANE)

Inicialmente, Cruz e Raydan (2003) introduziram o algoritmo *Spectral Algorithm for Nonlinear Equations* (SANE) para solucionar sistemas não lineares na forma (2.1). Nesse método, cada iteração é definida pela Equação (2.5), sendo $d_k = \pm\sigma_k F(x)$ a direção de busca e σ_k o coeficiente espectral. Dessa forma, a partir da equivalência de (2.5) para $x = x - \sigma F(x)$ quando $\sigma_k > 0$, o primeiro ponto de teste em cada iteração é

$$x_{k+1} = x_k - \sigma_k F(x_k). \tag{2.8}$$

Com o propósito de atingir a convergência global, Cruz e Raydan (2003) utilizam a técnica de busca linear não monótona de Grippo, Lampariello e Lucidi (1986), descrita como:

$$f(x_{k+1}) \leq \max_{0\leq j\leq M-1} f(x_{k-j}) + \gamma\alpha_k \nabla f(x_k)^T d_k, \tag{2.9}$$

onde $f(x) = F(x)^T F(x)$ e $\gamma \in \mathbb{R}$ é um pequeno número positivo. Na Equação (2.9), M é um real positivo responsável por definir o grau de não monotonicidade de f, no qual $M = 1$ define um modelo estritamente monótono. Essa abordagem requer direções de descida em relação à norma quadrada do resíduo $F(x)$, como consequência, é necessário o cálculo de uma derivada direcional ou uma boa aproximação dela a cada iteração. Isso porque o termo $\nabla f(x_k)^T d_k$ no SANE é substituído por $\pm F(x_k)^T J(x_k) F(x_k)$, onde $J(x_k)$ é a Jacobiana de F no ponto x_k. Entretanto, para evitar o uso de derivadas, SANE aplica a seguinte aproximação (BIRGIN; MARTíNEZ, 2022)

$$J(x_k)F(x_k) \approx \frac{F(x_k + hF(x_k)) - F(x_k)}{h}, \ h > 0,$$

que demanda uma avaliação de função extra em todas as iterações. Segue-se que se $\nabla f(x_k)^T d_k < 0$, então a condição (2.9) é satisfeita para α_k suficientemente próximo de zero e é possível calcular um passo α_k por um processo de retrocesso finito. No entanto, quando $\nabla f(x_k)^T d_k = 0$, não há garantias de que existe um valor de α_k que satisfaça a condição (2.9) (CRUZ; MARTÍNEZ; RAYDAN, 2006).

Com a finalidade de diminuir a quantidade de avaliações dos resíduos e para encontrar um passo α_k independentemente da escolha da direção d_k, Cruz, Martínez e Raydan

(2006) apresentaram o método DF-SANE com uma nova estratégia de busca linear não monótona sem o uso de derivadas:

$$f(x_{k+1}) \leq \max_{0\leq j\leq M-1} f(x_{k-j}) + \eta_k - \gamma\alpha_k^2 f(x_k), \tag{2.10}$$

onde $\eta_k > 0$ decresce a medida que k aumenta, de forma que, $\sum_{k=0}^{\infty}\eta_k = \eta < \infty$. Além disso, o termo $\eta_k > 0$ na desigualdade (2.10) garante que todas as iterações sejam bem definidas ao assegurar a escolha de um tamanho do passo α_k que faça a função objetivo ter um descrécimo a cada iteração. Especificamente, o termo implica que a próxima iteração, x_{k+1}, possui um valor de função objetivo menor do que a iteração atual, x_k, ou pelo menos menor do que alguma iteração anterior, x_{k-j}. Assim, η_k impede o algoritmo de ficar infinitamente em repetição ou de divergir, e faz as iterações convergirem para um mínimo local ou um ponto estacionário (CRUZ; MARTÍNEZ; RAYDAN, 2006). Outra mudança do DF-SANE é que a direção de busca passa a ser $d_k = -\sigma_k F(x_k)$, agora que o sinal associado a $F(x_k)$ não é responsável por definir uma direção de descida.

Ademais, diferente da condição (2.9), a condição (2.10) não envolve o produto quadrático com a Jacobiana, responsável por gerar uma avaliação de função adicional a cada iteração. Consequentemente, DF-SANE é geralmente mais econômico que SANE ao comparar o número de avaliações de F (VARADHAN; GILBERT, 2009).

Quanto ao cálculo do passo σ_{k+1}, o único passo espectral considerado nos trabalhos Cruz e Raydan (2003) e Cruz, Martínez e Raydan (2006) é:

$$\sigma_{k+1} = \frac{s_k^T s_k}{s_k^T y_k}, \tag{2.11}$$

para $s_k = x_{k+1} - x_k$ e $y_k = F(x_{k+1}) - F(x_k)$. Tal como exposto por Birgin e Martínez (2022), a Equação (2.11) fornece uma aproximação diagonal da Hessiana.

Como alternativa, o pacote BB (VARADHAN; GILBERT, 2009) também utiliza duas outras estratégias de passo espectral, uma proposta por Barzilai e Borwein (1988):

$$\sigma_{k+1} = \frac{s_k^T y_k}{y_k^T y_k} \tag{2.12}$$

e outra apresentada por Varadhan e Roland (2008):

$$\sigma_{k+1} = sgn(s_k^T y_k)\frac{\|s_k\|}{\|y_k\|}, \tag{2.13}$$

onde $sgn(x) = x/|x|$, quando $x \neq 0$, e zero quando $x = 0$.

Em resumo, o DF-SANE é descrito no Algoritmo 1.

Algoritmo 1: DF-SANE - Método Residual Espectral sem Derivadas para Equações Não Lineares

Entrada. Dado $x_0 \in \mathbb{R}^n$, $\gamma \in (0,1)$, $0 < \sigma_{min} < \sigma_{max} < \infty$, $0 < \tau_{min} < \tau_{max} < 1$, o inteiro positivo M, $\sigma_0 = 1$, uma sequência η_k tal que $\eta_k > 0 \; \forall \, k \in \mathbb{N}$ e $\lim_{k\to\infty} \eta_k = 0$.

Saída. $x \in \mathbb{R}^n$.

Passo 1. Inicialização de $k \leftarrow 0$.

Passo 2. Se $\|F(x_k)\| = 0$, finalizar a execução do algoritmo e retornar x_k como saída.

Passo 3.

- Calcular $\sigma_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}$ de modo que $|\sigma_k| \in [\sigma_{min}, \sigma_{max}]$ se $k > 0$;
- Caso $|\sigma_k| \notin [\sigma_{min}, \sigma_{max}]$, o coeficiente espectral é substituído por:

$$\sigma_k = \begin{cases} 1 & \text{caso} \quad \|F(x_k)\| > 1, \\ \|F(x_k)\|^{-1} & \text{caso} \quad 10^{-5} \leq \|F(x_k)\| \leq 1, \\ 10^{-5} & \text{caso} \quad \|F(x_k)\| < 10^{-5}. \end{cases} \tag{2.14}$$

Passo 4. Calcular $\bar{f}_k = \max\{f(x_k), \ldots, f(x_{\max\{0,k-M+1\}})\}$;

Passo 5.

- Definir $\alpha \leftarrow 1$ e $d \leftarrow -\sigma_k F(x_k)$;
- Considerar a condição:

$$f(x_k + \alpha d) \leq \bar{f}_k + \eta_k - \gamma \alpha^2 f(x_k); \tag{2.15}$$

- Se (2.15) for satisfeita, definir $d_k = d$ e $\alpha_k = \alpha$;
- Caso contrário, escolher $\alpha \in [\tau_{min}\alpha, \tau_{max}\alpha]$ e reavaliar a condição (2.15).

Passo 6. Calcular $x_{k+1} = x_k + \alpha_k d_k$.

Passo 7. Incrementar $k \leftarrow k + 1$ e voltar ao Passo 2.

Algumas considerações em relação ao Algoritmo 1:

1. Implementações computacionais substituem o critério de parada $\|F(x_k)\| = 0$ do Passo 1 por $\|F(x_k)\| \leq \epsilon$, dado a tolerância $\epsilon > 0$, como é o caso da implementação feita por Birgin e Martínez (2022);

2. No Passo 2, a substituição do coefiente espectral em (2.14) ocorre raramente caso seja utilizado valores grandes para σ_{max} e $1/\sigma_{min}$, por exemplo, 10^{10} e 10^{-10}, respectivamente. Cruz, Martínez e Raydan (2006) exploram essa ocorrência em seus experimentos;

3. As constantes τ_{min} e τ_{max}, utilizadas no Passo 4, são responsáveis por reduzir α de modo que ao calcular o mínimo entre α e $\tau_{min}\alpha$ evita-se reduções muito grandes e ao calcular o máximo entre α e $\tau_{max}\alpha$ evita-se reduções muito pequenas.

2.4 Implementação computacional do DF-SANE no BB

Nesta seção é apresentado detalhes da implementação do DF-SANE em R do Pacote BB. Varadhan e Gilbert (2009) apresentam as seguintes mudanças em relação ao código Fortran disponibilizado em Raydan (2009):

1. É possível escolher entre os coeficientes espectrais (2.11), (2.12), (2.13), enquanto originalmente apenas o coeficiente (2.11) é adotado;

2. O primeiro passo do BB é redimensionado como $\alpha_0 = \min\{1, 1/\|F(x_0)\|\}$, ao invés de $\alpha_0 = 1$;

3. Quando o DF-SANE está próximo da solução, isto é, quando $f(x_k) < 10^{-4}$, é utilizado uma estratégia de retardo dinâmico publicada por Luengo e Raydan (2003), o qual propõe o uso do passo espectral de duas iterações prévias na iteração atual;

4. É implementado um critério de parada adicional em que o algoritmo é encerrado caso não ocorra a diminuição na função objetivo $f(x)$ no decorrer de $noimp$ iterações, onde é definido um valor padrão de $noimp = 100$. Esse critério é tido como essencial para casos em que $M \geq 100$;

5. É fornecido a opção de aprimorar os valores iniciais ao habilitar o uso do método simplex não linear de Nelder e Mead (1965);

6. É disponibilizado a função BBsolve para quando o DF-SANE falhar em convergir à uma solução. Essa função implementa a seguinte estratégia:

   a) Tentar um parâmetro de não monotonicidade diferente M. Como o padrão é $M = 10$, tente $M = 50$;

   b) Tentar outras escolhas de passo espectral. A escolha padrão é (2.12);

   c) Tentar executar o método com a inicialização Nelder-Mead.

2.5 Estratégia de aceleração

Estratégias de aceleração incorporam elementos de iterações prévias, ao invés de gastarem recursos realizando avaliações na iteração atual. Birgin e Martínez (2022) apresentam uma versão generalizada e acelerada do DF-SANE que permite direções não residuais, embora contenham informações relacionadas aos resíduos.

Tal como descrito por Birgin et al. (2021), a estratégia de aceleração aplicada no DF-SANE consiste em computar um x_{k+1}^{accel} antes de definir o ponto de teste x_{k+1}^{trial}, resultado do processo de busca linear retroativa, como próximo iterador x_{k+1}. O algoritmo de aceleração é descrito no Algoritmo 2.

Algoritmo 2: Estratégia de aceleração do DF-SANE

Entrada. $p \geq 1$, um inteiro que delimita a quantidade de iterações anteriores a se considerar; $x_{k+1}^{trial} \in \mathbb{R}^n$, a variável que será acelerada.

Saída. $x \in \mathbb{R}^n$.

Passo 1.

- Definir $\bar{k} = \max\{0, k - p + 1\}$,
- $s_j = x_{j+1} - x_j$ para $j = \bar{k}, \ldots, k-1$,
- $y_j = F(x_{j+1}) - F(x_j)$ para $j = \bar{k}, \ldots, k-1$,
- $s_k = x_{k+1}^{trial} - x_k$,
- $y_k = F(x_{k+1}^{trial}) - F(x_k)$,
- $S_k = (s_{\bar{k}}, \ldots, s_{k-1}, s_k)$,
- $Y_k = (y_{\bar{k}}, \ldots, y_{k-1}, y_k)$ e

$$x_{k+1}^{accel} = x_k - S_k Y_k^{\dagger} F(x_k), \tag{2.16}$$

dado que $Y_k^{\dagger}$ é a pseudo-inversa de Moore-Penrose de Y_k;

Passo 2. Escolher $x_{k+1} \in \{x_{k+1}^{trial}, x_{k+1}^{accel}\}$ que satisfaça

$$\|F(x_{k+1})\| = \min\{\|F(x_{k+1}^{trial})\|, \|F(x_{k+1}^{accel})\|\}. \tag{2.17}$$

Passo 3. Retorna x_{k+1} definido no passo anterior como saída.

Na Seção 3.1 são apresentados detalhes da implementação computacional do Algoritmo 2.

3 Proposta Metodológica

Este capítulo descreve os passos para atingir os objetivos específicos descritos na Seção 1.2. Para isso, é descrito a aplicação da estratégia de aceleração e a abordagem da experimentação numérica.

3.1 Aplicação da estratégia de aceleração

Para atender o objetivo específico **OE1**, foi aplicado a estratégia de aceleração do Algoritmo 2 na implementação computacional descrita na Seção 2.4. O objetivo desta seção é descrever como foi feita a implementação prática da estratégia de aceleração.

Assim, conforme exposto por Birgin e Martínez (2022), computar x_{k+1}^{accel} como exposto em (2.16) é equivalente à encontrar a solução $\bar{\omega}$ para o sistema linear $Y_k\omega = F(x_{k+1}^{trial})$ e avaliar $x_{k+1}^{accel} = x_{k+1}^{trial} - S_k\bar{\omega}$. Na prática, soluciona-se o sistema linear com a ortogonalização de Y_k. No entanto, fatorar uma matriz a cada iteração do método é um processo computacionalmente caro, por isso, o ponto chave é que o cálculo de Y_k corresponde a remover e/ou adicionar uma coluna à matriz Y_{k-1} e, neste caso, atualizar a fatoração ortogonal da matriz é possível e mais barato do que calcular a fatoração novamente. Os passos desse procedimento estão descritos por completo no Algoritmo 3.

Algoritmo 3: Implementação da estratégia de aceleração do DF-SANE.

Entrada. Dado $0 < h_{small} < h_{large}$ e $p \geq 1$.

Saída. $x \in \mathbb{R}^n$.

Passo 1. Inicialização $r_{max} \leftarrow 0$ e $l \leftarrow 1$.

Passo 2. Definir $x_{k+1}^{trial} = x_k + \alpha_k d_k$ no Passo 6 do Algoritmo 1.

Passo 3. Caso $k = 0$ (primeira iteração do DF-SANE), S_k e Y_k são definidos como matrizes de zero colunas. Caso contrário, $S_k \leftarrow S_{k-1}$ e $Y_k \leftarrow Y_{k-1}$;

Passo 4. Caso S_k e Y_k tiverem p colunas, remover a coluna mais a esquerda de S_k e Y_k.

Passo 5. Adicionar $s_k = x_{k+1}^{trial} - x_k$ e $y_k = F(x_{k+1}^{trial}) - F(x_k)$ como colunas mais a direita de S_k e Y_k, respectivamente. Definir $r_{max} \leftarrow \max\{r_{max}, rank(Y_k)\}$, dado que $rank(A)$ representa o posto da matriz A.

Passo 6. Caso $rank(Y_k) < r_{max}$, executar os passos 6.1-6.2:

Passo 6.1. Caso S_k e Y_k tiverem p colunas, remover a coluna mais a esquerda de S_k e Y_k.

Passo 6.2. Adicionar $s_{extra} = x_{extra} - x_k$ e $y_{extra} = F(x_{extra}) - F(x_k)$ como colunas mais a direita de S_k e Y_k, respectivamente, onde $x_{extra} = x_k + h_{small}e_l$. Definir $r_{max} \leftarrow \max\{r_{max}, rank(Y_k)\}$ e $l \leftarrow \mod(l, n) + 1$.

Passo 7. Caso $rank(Y_k) \neq 0$, executar os passos 7.1-7.3:

Passo 7.1. Calcular $\bar{\omega}$ que solucione o sistema linear $Y_k\omega = F(x_{k+1}^{trial})$ e definir $x_{k+1}^{accel} = x_{k+1}^{trial} - S_k\bar{\omega}$.

Passo 7.2. Se o Passo 6.2 foi executado, remover a coluna mais a direita de S_k e Y_k, isto é, as colunas s_{extra} e y_{extra}.

Passo 7.3. Caso $x_{k+1}^{accel} \neq x_k$, $\|x_{k+1}^{accel}\| \leq 10\max\{1, \|x_k\|\}$ e $\|F(x_{k+1}^{trial})\| < \|F(x_{k+1}^{accel})\|$, redefinir $x_{k+1}^{trial} = x_{k+1}^{accel}$, substituir a coluna mais a direita de S_k e Y_k, isto é, colunas s_k e y_k, por $s_{accel} = x_{k+1}^{accel} - x_k$ e $y_{accel} = F(x_{k+1}^{accel}) - F(x_k)$, respectivamente. Definir $r_{max} \leftarrow \max\{r_{max}, rank(Y_k)\}$.

Passo 8. Caso $rank(Y_k) = 0$, executar os passos 8.1-8.3:

Passo 8.1. Redefinir as matrizes S_k e Y_k como matrizes de zero colunas.

Passo 8.2. Executar $p - 1$ vezes:

- Adicionar $s_{extra} = x_{extra} - x_{k+1}^{trial}$ e $y_{extra} = F(x_{extra}) - F(x_{k+1}^{trial})$ como colunas mais a direita de S_k e Y_k, respectivamente, onde $x_{extra} = x_k + h_{large}e_l$. Definir $r_{max} \leftarrow \max\{r_{max}, rank(Y_k)\}$ e $l \leftarrow \mod(l, n) + 1$.

Passo 8.3. Adicionar $s_k = x_{k+1}^{trial} - x_k$ e $y_k = F(x_{k+1}^{trial}) - F(x_k)$ como colunas mais a direita de S_k e Y_k, respectivamente. Definir $r_{max} \leftarrow \max\{r_{max}, rank(Y_k)\}$.

Passo 9. Calcular $\bar{\omega}$ que solucione o sistema linear $Y_k\omega = F(x_{k+1}^{trial})$ e definir $x_{k+1}^{accel} = x_{k+1}^{trial} - S_k\bar{\omega}$.

Passo 10. Caso $x_{k+1}^{accel} \neq x_k$, $\|x_{k+1}^{accel}\| \leq 10\max\{1, \|x_k\|\}$ e $\|F(x_{k+1}^{trial})\| < \|F(x_{k+1}^{accel})\|$, redefinir $x_{k+1}^{trial} = x_{k+1}^{accel}$, substituir a coluna mais a direita de S_k e Y_k, isto é, colunas s_k e y_k, por $s_{accel} = x_{k+1}^{accel} - x_k$ e $y_{accel} = F(x_{k+1}^{accel}) - F(x_k)$, respectivamente. Definir $r_{max} \leftarrow \max\{r_{max}, rank(Y_k)\}$.

Passo 11. Retornar x_{k+1}^{trial} como saída.

Algumas considerações em relação ao Algoritmo 3:

1. Birgin e Martínez (2022) afirmam que, no geral, o Passo 8 não é executado. Caso a iteração k for tal que o Passo 8 não foi executado nas p iterações anteriores, as matrizes S_k e Y_k correspondem a remoção da coluna mais a esquerda e a adição de uma nova coluna mais a direita para S_{k-1} e Y_{k-1}, respectivamente. Entretanto, a coluna mais a esquerda não é removida quando o número máximo de colunas p não foi atingido;

2. A decomposição QR de Y_k deve ser recalculada do zero sempre que o Passo 8 for executado, isso porque a matriz Y_k é uma matriz nula;

3. Quando $k = 0$, Y_k é uma matriz de coluna única, logo, as matrizes P, Q e R da decomposição QR, em que $Y_kP = QR$, podem ser obtidas trivialmente;

4. Os números reais h_{small} e h_{large} são utilizados nos passos 6.2 e 8.2 para perturbar o l-ésimo componente do vetor x_{extra};

5. Quando o Passo 6 é executado, isso significa que na iteração k foi construída uma matriz Y_k com posto $rank(Y_k)$ menor que o maior posto r_{max} calculado nas iterações anteriores, ou seja, no Passo 5 foi adicionada uma coluna y_k linearmente dependente à outra coluna de Y_k. Diante disso, nos Passos 6.1 e 6.2 são realizadas modificações na matriz Y_k.

3.2 Abordagem da Experimentação Numérica

Para atender o objetivo específico **OE2**, a experimentação numérica utiliza como base as seguintes propriedades apresentadas por Nocedal e Wright (2006) para validar o algoritmo proposto:

1. Robustez: ter a capacidade de solucionar uma grande variedade de problemas da mesma classe, para todas escolhas razoáveis de variáveis iniciais;

2. Eficiência: método não deve exigir alto consumo de memória nem muito tempo;

3. Acurácia: deve ser capaz de identificar uma solução com precisão, com pouca sensibilidade a erros nos dados e erros de arredondamento.

Os experimentos numéricos foram feitos para o conjunto de problemas propostos por Moré, Garbow e Hillstrom (1981), fornecidos pelo banco de testes da biblioteca CUTEst (GOULD; ORBAN; TOINT, 2014). As funções desses problemas estão na forma de mínimos quadrados, ou seja, definidas por $f_1, f_1, \ldots, f_m$. É possível obter um problema de minimização irrestrita ao definir nossa função objetivo tal como feito em (2.4). Solucionar

o problema (2.4) é o mesmo que solucionar o problema para sistemas de equações não lineares (2.1).

A partir desse ponto, DF-SANEacc irá se referir à implementação feita neste trabalho e DF-SANE irá se referir à implementação apresentada por Varadhan e Gilbert (2009).

Os testes foram organizados da seguinte forma, é feita uma comparação entre o método de DF-SANEacc e o DF-SANE para os 70 problemas de sistemas de equações não lineares da coleção do CUTEst, com suas dimensões e valores iniciais padrão utilizados.

Os critérios que são utilizados para avaliar as implementações estão descritos no Quadro 1.

Quadro 1 - Critérios de avaliação adotados na experimentação numérica.

Sigla	Descrição
avalf	Quantidade de avaliações da função objetivo.
it	Quantidade de iterações do algoritmo.
cpu	Tempo de execução (em segundos).
$\|\|f(x^*)\|\|$	Norma de $f(x^*)$, função definida em (2.4) sendo x^* o vetor solução.

Fonte: Autor, 2022.

Na Tabela 1 são listados todos os 70 problemas de sistemas de equações não lineares da coleção do CUTEst com suas dimensões n. Foram criadas siglas para cada problema para facilitar sua referência.

A comparação entre os algoritmos envolve verificar o seguinte critério:

$$\frac{fm - fmin}{\max(1, |fmin|)} \leq \epsilon, \tag{3.1}$$

onde fm é a norma final da função objetivo, isto é, quando avaliada em $\|f(x^*)\|$; sendo x^* o k-ésimo ponto calculado pelo algoritmo em questão; $fmin$ é o mínimo entre os fm de cada solução, ou seja, $min(fm_{DF-SANEacc}, fm_{DF-SANE})$; e ϵ é a tolerância usada no critério de parada.

Tabela 1 – Os 70 problemas de sistemas de equações não lineares do CUTEst.

Nome do problema	**n**	**Nome do problema**	**n**
BOOTH	2	⋮	⋮
CLUSTER	2	TRIGON1NE	10
CUBENE	2	INTEQNE	12
DENSCHNCNE	2	HATFLDG	25
DENSCHNFNE	2	HYDCAR6	29
FREURONE	2	METHANB8	31
GOTTFR	2	METHANL8	31
HIMMELBA	2	HYDCAR20	99
HIMMELBC	2	LUKSAN21	100
HIMMELBD	2	MANCINONE	100
HS8	2	QINGNE	100
HYPCIR	2	ARGTRIG	200
POWELLBS	2	BROWNALE	200
POWELLSQ	2	CHANDHEU	500
PRICE3NE	2	10FOLDTR	1000
PRICE4NE	2	KSS	1000
RSNBRNE	2	MSQRTA	1024
SINVALNE	2	MSQRTB	1024
WAYSEA1NE	2	EIGENAU	2550
WAYSEA2NE	2	EIGENB	2550
DENSCHNDNE	3	EIGENC	2652
DENSCHNENE	3	NONMSQRTNE	4900
HATFLDF	3	BROYDN3D	5000
HATFLDFLNE	3	BROYDNBD	5000
HELIXNE	3	BRYBNDNE	5000
HIMMELBE	3	NONDIANE	5000
RECIPE	3	SBRYBNDNE	5000
ZANGWIL3	3	SROSENBRNE	5000
POWELLSE	4	SSBRYBNDNE	5000
POWERSUMNE	4	TQUARTICNE	5000
HEART6	6	OSCIGRNE	100000
HEART8	8	CYCLIC3	100002
COOLHANS	9	YATP1CNE	123200
MOREBVNE	10	YATP1NE	123200
OSCIPANE	10	YATP2CNE	123200
⋮	⋮	YATP2SQ	123200

Fonte: Autor, 2022.

4 Experimentação Numérica

Este capítulo apresenta os resultados obtidos ao realizar os experimentos numéricos descritos na Seção 3.2. Todos os experimentos foram realizados numa arquitetura x86, 64 bits, com 8 CPUs, 16 MB de memória RAM, com 100 GB disponíveis no disco e com o sistema operacional Debian GNU/linux 12.

Ademais, foram considerados 69 dos 70 problemas da Tabela 1. Não foi possível obter resultados para o problema HYDCAR20 ao testar o DF-SANEacc, pois foi interrompido por atingir um tempo de execução superior a 12 horas e por isso esse problema foi excluído das comparações.

4.1 Resultados obtidos

Nesta seção, são apresentados os resultados obtidos pelo DF-SANE e pelo DF-SANEacc para os 69 problemas de sistemas de equações não lineares. Detalhes de como reproduzir os testes estão disponíveis em um repositório do GitLab[1].

A Tabela 2 apresenta os resultados dos dois algoritmos com informações sobre a norma e sobre a quantidade de avaliações de função. Em complemento, Tabela 3 apresenta o resultado para os mesmos testes mas com informações sobre a quantidade de iterações e o tempo de execução. Em ambas, o sufixo dfsane se refere a implementação de Varadhan e Gilbert (2009) enquanto o sufixo dfsaneacc se refere a implementação feita para este trabalho. Além disso, está destacado em negrito a menor norma, a menor quantidade de avaliações de função, a menor quantidade de iterações e o menor tempo gasto em CPU.

Tabela 2 – Parte um dos resultados dos 70 problemas para o algoritmo DF-SANE e para DF-SANEacc.

problema	norma_dfsane	norma_dfsaneacc	avalf_dfsane	avalf_dfsaneacc
BOOTH	2.4 E-07	**1.8 E-15**	8	**5**
CLUSTER	**2.4 E-07**	1.0 E-06	42	**24**
CUBENE	1.0 E-06	**6.0 E-15**	**26**	35
DENSCHNCNE	**1.4 E-07**	1.6 E-07	17	**13**
DENSCHNFNE	2.5 E-07	**3.1 E-08**	40	**21**
FREURONE	1.1 E+01	**7.0 E+00**	**123**	135735

[1] https://gitlab.com/sergiosacj/dfsaneacc

GOTTFR	2.6 E-02	**1.6 E-06**	154606	**36**
HIMMELBA	**1.3 E-07**	0.0 E+00	8	**5**
HIMMELBC	7.0 E-07	**1.5 E-07**	11	11
HIMMELBD	2.4 E+00	2.4 E+00	**211**	221
HS8	2.1 E-07	**1.5 E-07**	15	**11**
HYPCIR	1.2 E-06	**1.0 E-09**	14	**11**
POWELLBS	8.4 E-07	**6.2 E-08**	367	**208**
POWELLSQ	**9.8 E-03**	7.8 E+00	6522441	**2415**
PRICE3NE	9.0 E-07	**7.4 E-07**	16	**11**
PRICE4NE	2.0 E-08	**4.0 E-12**	39	**24**
RSNBRNE	**3.7 E-07**	0.0 E+00	**564**	766
SINVALNE	2.1 E+00	**1.7 E-20**	52078	**341**
WAYSEA1NE	1.0 E-06	**1.6 E-07**	3466	**28**
WAYSEA2NE	4.8 E-07	**3.4 E-07**	1420604632	**315631**
DENSCHNDNE	1.8 E-06	**2.0 E-08**	85	**56**
DENSCHNENE	9.8 E-01	**6.6 E-08**	112	**24**
HATFLDF	**9.6 E-07**	8.5 E-03	**907**	2354
HATFLDFLNE	8.2 E-03	**8.1 E-03**	**251**	444
HELIXNE	3.1 E+01	**1.4 E-08**	574	**42**
HIMMELBE	2.1 E+00	**1.1 E-14**	128	**17**
RECIPE	**1.4 E-06**	1.5 E-06	57	**29**
ZANGWIL3	1.3 E-08	**2.1 E-14**	27	**8**
POWELLSE	1.5 E+01	**3.6 E-06**	240	**67**
POWERSUMNE	2.0 E-02	**1.4 E-03**	**485**	2325
HEART6	1.9 E+01	**5.1 E+00**	**476**	3684
HEART8	1.3 E+01	**6.5 E+00**	**332**	933
COOLHANS	3.5 E-02	**2.7 E-06**	124	**22**
MOREBVNE	**8.9 E-06**	9.5 E-06	**64**	69
OSCIPANE	1.0 E+00	1.0 E+00	**113**	1942
TRIGON1NE	7.6 E-06	**3.4 E-06**	31	**27**
INTEQNE	1.2 E-06	**9.2 E-07**	**6**	7
HATFLDG	5.0 E+00	**1.9 E-02**	**189**	44184
HYDCAR6	2.5 E+01	**1.1 E-04**	**430**	119840962
METHANB8	9.9 E-01	**3.1 E-05**	**109**	35474830
METHANL8	6.5 E+01	**4.4 E-05**	**490**	126293080
LUKSAN21	8.7 E-05	**7.2 E-05**	**78**	97
MANCINONE	5.2 E-06	**2.6 E-06**	**8**	11

QINGNE	**4.5 E-06**	7.0 E-05	**36**	44
ARGTRIG	**1.3 E-04**	1.7 E-04	**69**	93
BROWNALE	**1.2 E-07**	1.4 E-05	16	**15**
CHANDHEU	2.9 E-04	**2.4 E-04**	64	**27**
10FOLDTR	**1.8 E+05**	9.7 E+07	1167	**899**
KSS	7.5 E-06	**3.8 E-06**	**12**	17
MSQRTA	8.6 E+01	**6.6 E+01**	**585**	1777
MSQRTB	8.6 E+01	**5.6 E+01**	**615**	1895
EIGENAU	1.8 E+02	**1.7 E+02**	**563**	1596
EIGENB	9.9 E+00	**9.3 E+00**	**7459**	21312
EIGENC	1.0 E+02	1.0 E+02	**545**	754
NONMSQRTNE	**2.2 E+02**	2.3 E+02	362828	**3945**
BROYDN3D	4.4 E-03	4.0 E-03	**10**	15
BROYDNBD	3.6 E+01	**3.1 E+00**	**327**	641
BRYBNDNE	3.6 E+01	**3.1 E+00**	**327**	641
NONDIANE	6.4 E+02	6.4 E+02	483	**233**
SBRYBNDNE	2.6 E+02	**2.5 E+02**	**897**	40414
SROSENBRNE	2.9 E-03	**2.3 E-04**	16	16
SSBRYBNDNE	**1.3 E+02**	1.5 E+02	**1192**	6644
TQUARTICNE	8.9 E-01	**8.7 E-01**	**3991**	873298
OSCIGRNE	9.4 E-02	**5.7 E-02**	**16**	31
CYCLIC3	**8.0 E-02**	9.1 E-02	11751	**1793**
YATP1CNE	**8.4 E+03**	8.8 E+03	865	**690**
YATP1NE	**8.4 E+03**	8.8 E+03	865	**573**
YATP2CNE	7.2 E+04	**3.6 E+04**	**830**	1511
YATP2SQ	4.5 E+04	**4.0 E+04**	**115**	2187

Fonte: Autor, 2023.

Tabela 3 – Parte dois dos resultados dos 70 problemas para o algoritmo DF-SANE e para DF-SANEacc.

problema	it_dfsane	it_dfsaneacc	cpu_dfsane	cpu_dfsaneacc
BOOTH	7	**2**	0.009677	**0.071090**
CLUSTER	40	**11**	**0.012481**	0.065467
CUBENE	24	**16**	**0.010722**	0.068262
DENSCHNCNE	16	**6**	**0.010610**	0.067065
DENSCHNFNE	27	**7**	**0.012112**	0.074197

FREURONE	**103**	9664	**0.015577**	4.795574
GOTTFR	24196	**12**	3.536324	**0.068866**
HIMMELBA	7	**2**	**0.008975**	0.065403
HIMMELBC	10	**5**	**0.009745**	0.067633
HIMMELBD	**188**	191	**0.019576**	0.095881
HS8	14	**5**	**0.009763**	0.073121
HYPCIR	13	**5**	**0.009430**	0.070058
POWELLBS	106	**28**	**0.022314**	0.080151
POWELLSQ	665188	**343**	89.302055	**0.162580**
PRICE3NE	15	**5**	**0.010608**	0.067344
PRICE4NE	37	**11**	**0.011259**	0.071052
RSNBRNE	428	**339**	**0.030767**	0.150036
SINVALNE	5063	**152**	1.154021	**0.092579**
WAYSEA1NE	785	**11**	0.121268	**0.067344**
WAYSEA2NE	81301108	**37854**	18139.287895	**10.716876**
DENSCHNDNE	82	**25**	**0.015293**	0.068122
DENSCHNENE	107	**11**	**0.015287**	0.068175
HATFLDF	**586**	637	**0.041050**	0.204480
HATFLDFLNE	170	**132**	**0.023284**	0.097500
HELIXNE	102	**15**	**0.033012**	0.067867
HIMMELBE	127	**8**	**0.019777**	0.072884
RECIPE	56	**11**	**0.014227**	0.072016
ZANGWIL3	25	**3**	**0.010080**	0.065017
POWELLSE	101	**26**	**0.021262**	0.070560
POWERSUMNE	**411**	946	0.031091	**0.258941**
HEART6	**116**	681	**0.025791**	0.267418
HEART8	**101**	302	**0.023622**	0.151187
COOLHANS	120	**9**	**0.016577**	0.069199
MOREBVNE	61	**34**	**0.013931**	0.072839
OSCIPANE	**100**	220	**0.015156**	0.159600
TRIGON1NE	28	**12**	**0.010238**	0.070980
INTEQNE	5	**3**	**0.009550**	0.070150
HATFLDG	**102**	5355	**0.019627**	2.217885
HYDCAR6	**102**	8267391	**0.027560**	3237.022552
METHANB8	**102**	2830566	**0.016875**	1004.405806
METHANL8	**101**	8625813	**0.029145**	3407.543103
LUKSAN21	61	**42**	**0.015170**	0.076424

MANCINONE	7	**5**	**0.016313**	0.074277
QINGNE	30	**20**	**0.010976**	0.058789
ARGTRIG	62	**45**	**0.017194**	0.109913
BROWNALE	15	**7**	**0.007080**	0.061283
CHANDHEU	59	**13**	**0.164447**	0.168745
10FOLDTR	183	**145**	1.342815	**1.200500**
KSS	9	**7**	**0.031289**	0.096375
MSQRTA	**129**	638	**0.369329**	1.487940
MSQRTB	**123**	624	**0.243893**	0.973042
EIGENAU	**118**	447	**1.549025**	3.570246
EIGENB	**856**	3400	**14.516752**	37.817333
EIGENC	**112**	161	**1.603910**	2.246373
NONMSQRTNE	52286	**691**	1285.238467	**15.347828**
BROYDN3D	9	**7**	**0.009313**	0.088166
BROYDNBD	**124**	272	**0.277133**	1.061755
BRYBNDNE	**124**	272	**0.226941**	1.063594
NONDIANE	102	**101**	**0.203326**	0.406652
SBRYBNDNE	**319**	3970	**0.549103**	20.165811
SROSENBRNE	14	**7**	**0.012108**	0.073420
SSBRYBNDNE	**302**	1377	**0.762183**	4.886198
TQUARTICNE	**790**	59861	**1.305786**	260.667837
OSCIGRNE	15	15	**0.210080**	1.203669
CYCLIC3	11396	**719**	68.583235	**27.944267**
YATP1CNE	103	**102**	18.501421	**17.199562**
YATP1NE	103	**102**	16.757265	**14.347459**
YATP2CNE	**114**	166	**15.212269**	33.324985
YATP2SQ	**104**	219	**2.580487**	46.940917

Fonte: Autor, 2023.

A comparação entre a norma de $f(x^*)$ dos algoritmos seguiu o critério 3.1. A partir dele é possível determinar que o DF-SANE resolveu 36 dos 69 problemas enquanto o DF-SANEacc resolveu 59 dos 69 problemas. A Tabela 4 apresenta quais problemas cada algoritmo resolveu, sendo que estão em negrito os 26 problemas que ambos resolveram. A tolerância definida para cada problema foi de

$$\epsilon = 1.e - 06 * \sqrt{n},$$

sendo n as dimensões que aparecem na Tabela 1.

Considera-se que um algoritmo resolveu o problema caso atenda o critério 3.1, ou seja, o critério de parada do algoritmo não é considerado.

Tabela 4 – Quais problemas cada algoritmo solucionou. Em negrito, os 26 problemas que ambos solucionaram, sendo que o DF-SANE resolveu 36 problemas no total e o DF-SANEacc 59 problemas.

problem	dfsane	dfsaneacc
BOOTH	Sim	Sim
CLUSTER	Sim	Sim
CUBENE	Sim	Sim
DENSCHNCNE	Sim	Sim
DENSCHNFNE	Sim	Sim
FREURONE	Não	Sim
GOTTFR	Não	Sim
HIMMELBA	Sim	Sim
HIMMELBC	Sim	Sim
HIMMELBD	Sim	Sim
HS8	Sim	Sim
HYPCIR	Sim	Sim
POWELLBS	Sim	Sim
POWELLSQ	Sim	Não
PRICE3NE	Sim	Sim
PRICE4NE	Sim	Sim
RSNBRNE	Sim	Sim
SINVALNE	Não	Sim
WAYSEA1NE	Sim	Sim
WAYSEA2NE	Sim	Sim
DENSCHNDNE	Não	Sim
DENSCHNENE	Não	Sim
HATFLDF	Sim	Não
HATFLDFLNE	Não	Sim
HELIXNE	Não	Sim
HIMMELBE	Não	Sim
RECIPE	Sim	Sim
ZANGWIL3	Sim	Sim
POWELLSE	Não	Sim

POWERSUMNE	Não	Sim
HEART6	Não	Sim
HEART8	Não	Sim
COOLHANS	Não	Sim
MOREBVNE	Sim	Sim
OSCIPANE	Sim	Sim
TRIGON1NE	Não	Sim
INTEQNE	Sim	Sim
HATFLDG	Não	Sim
HYDCAR6	Não	Sim
METHANB8	Não	Sim
METHANL8	Não	Sim
LUKSAN21	Não	Sim
MANCINONE	Sim	Sim
QINGNE	Sim	Não
ARGTRIG	Sim	Não
BROWNALE	Sim	Sim
CHANDHEU	Não	Sim
10FOLDTR	Sim	Não
KSS	Sim	Sim
MSQRTA	Não	Sim
MSQRTB	Não	Sim
EIGENAU	Não	Sim
EIGENB	Não	Sim
EIGENC	Sim	Sim
NONMSQRTNE	Sim	Não
BROYDN3D	Não	Sim
BROYDNBD	Não	Sim
BRYBNDNE	Não	Sim
NONDIANE	Sim	Sim
SBRYBNDNE	Não	Sim
SROSENBRNE	Não	Sim
SSBRYBNDNE	Sim	Não
TQUARTICNE	Não	Sim
OSCIGRNE	Não	Sim
CYCLIC3	Sim	Não
YATP1CNE	Sim	Não

YATP1NE	Sim	Não
YATP2CNE	Não	Sim
YATP2SQ	Não	Sim

Fonte: Autor, 2023.

4.2 Análise de resultados

Nesta seção será feita uma comparação gráfica com o uso de perfis de desempenho. A justificativa do uso desse método é porque Dolan e Moré (2002) introduziram perfis de desempenho como uma ferramenta para avaliar e comparar o desempenho de softwares para otimização. Perfis de desempenho fornecem um meio de visualizar a diferença de desempenho esperada entre algoritmos, evitando escolhas arbitrárias de parâmetros para a comparação.

Ao todo foram gerados seis perfis. Os perfis 1, 2 e 3 utilizam todos os 69 problemas apresentados e os perfis 4, 5 e 6 utilizam para comparação apenas os 26 problemas que ambos algoritmos solucionaram, baseado na Tabela 4.

Diante desses resultados podemos fazer algumas observações:

- O tempo gasto de CPU pelo DF-SANEacc se mostrou pior em ambos perfis 2 e 5. A estratégia de aceleração depende da fatoração de matrizes, cujo custo computacional é de $O(n^3)$. Além disso, a implementação da fatoração foi desenvolvida em Fortran por Birgin et al. (2021), para poder reutilizar essa implementação foi empregado chamadas em R, usando a ferramenta `.Call`, para uma sub-rotina C que chama uma sub-rotina Fortran que realiza a decomposição QR. A escolha natural seria chamar as rotinas Fortran diretamente, no entanto, experimentos numéricos feitos por Birgin et al. (2021) mostraram que a combinação de `.Call` com a interface C é mais rápido. Ainda assim, essas chamadas para outras linguagens impacta no tempo de execução. Esse tempo pode ser melhorado ao paralelizar as operações com matrizes, algo que não foi explorado neste trabalho;

- A quantidade de iterações mostra que o emprego da aceleração aumentou a eficiência do algoritmo DF-SANE. Na Figura 3, quando a abscissa equivale a um, a proporção de problemas resolvidos pelo DF-SANE é de 35% enquanto o DF-SANEacc resolveu 67% dos 69 problemas. Na Figura 6, quando a abscissa vale um, o algoritmo DF-SANE resolveu 11% enquanto DF-SANEacc resolveu 88% dos 26 problemas. Além disso, quando consideramos os problemas em que ambos resolveram, o DF-SANEacc resolve 100% dos problemas com menos de $5.6(2^{2.5})$ iterações, já o DF-SANE atinge o mesmo valor com mais de $1024(2^{10})$ iterações;

- A quantidade de avaliações da função objetivo mostra que o uso da aceleração também aumentou a robustez do DF-SANE. Quando consideramos os 69 problemas, Figura 1, o desempenho é similar. No entanto, quando observamos para a Figura 4, o DF-SANE resolveu 38% dos problemas quando o valor no eixo das abscissas é um, à medida que o DF-SANEacc resolveu 65% dos problemas.

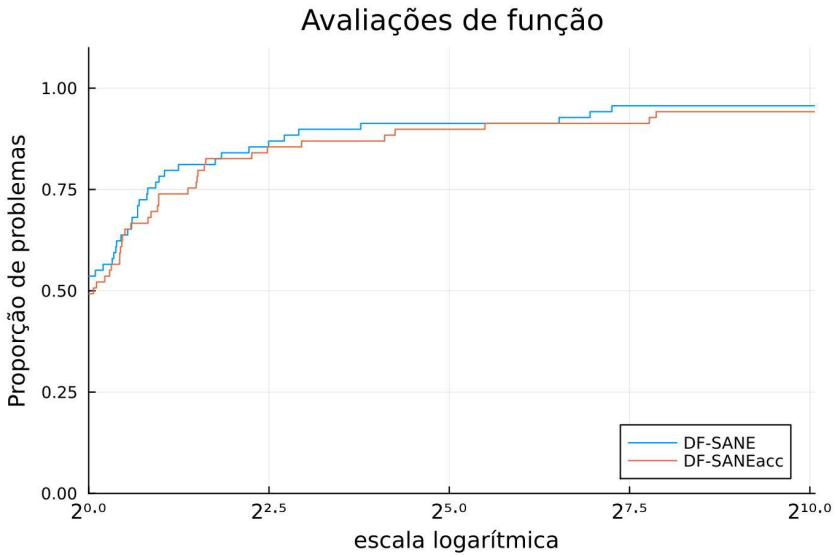

Figura 1 – Perfil de desempenho para os 69 problemas, comparação da quantidade de avaliações de função.

Fonte: Autor, 2023.

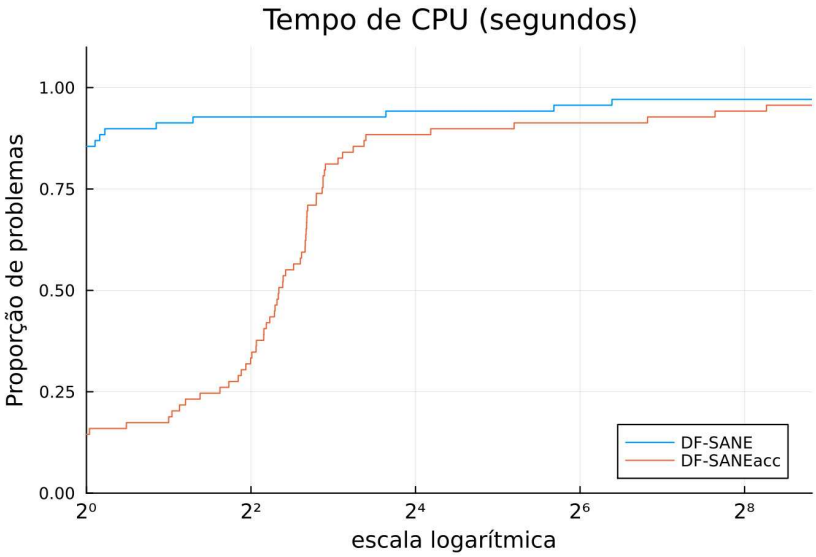

Figura 2 – Perfil de desempenho para os 69 problemas, comparação do tempo gasto em CPU.

Fonte: Autor, 2023.

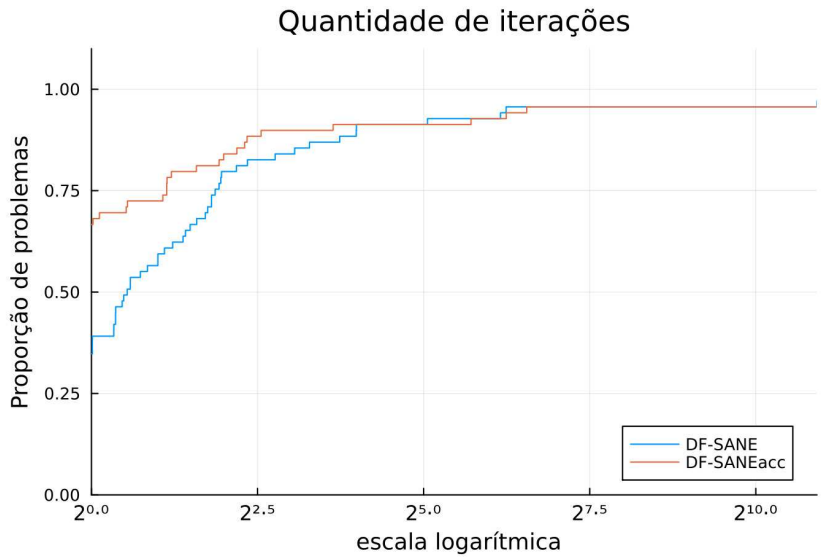

Figura 3 – Perfil de desempenho para os 69 problemas, comparação da quantidade de iterações.

Fonte: Autor, 2023.

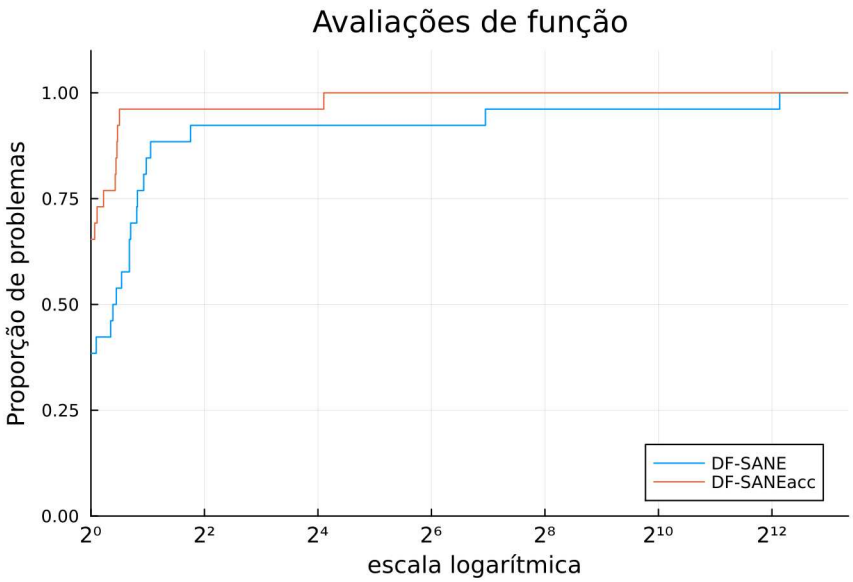

Figura 4 – Perfil de desempenho para problemas resolvidos por ambos algoritmos, comparação da quantidade de avaliações de função.

Fonte: Autor, 2023.

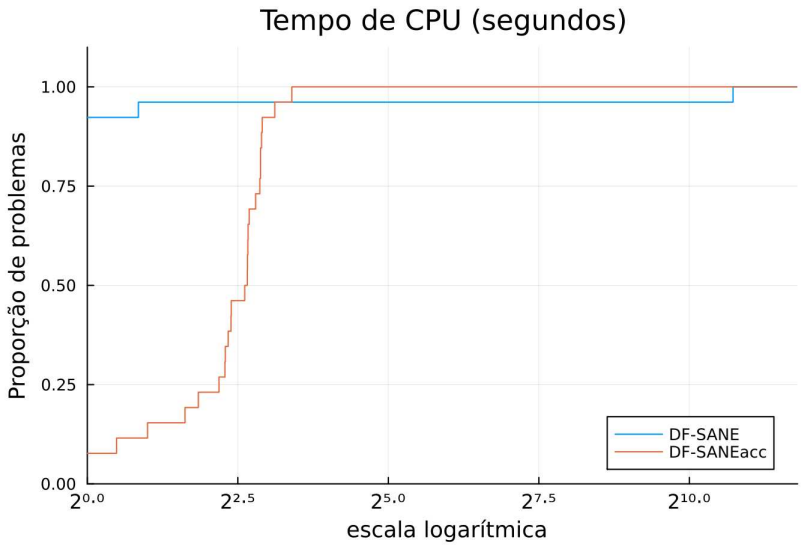

Figura 5 – Perfil de desempenho para problemas resolvidos por ambos algoritmos, comparação do tempo gasto em CPU.

Fonte: Autor, 2023.

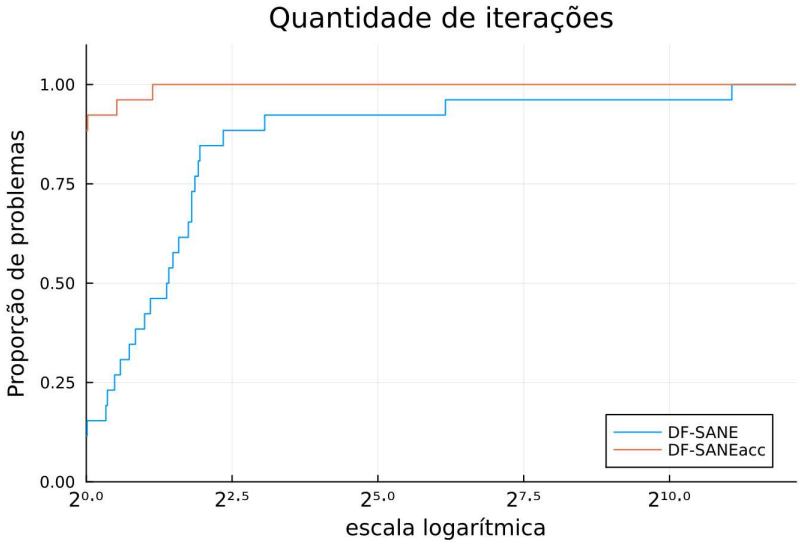

Figura 6 – Perfil de desempenho para problemas resolvidos por ambos algoritmos, comparação da quantidade de iterações.

Fonte: Autor, 2023.

5 Considerações Finais

Neste trabalho foi implementado e testado o uso da estratégia de aceleração em um método sem derivadas para a resolução de sistemas não lineares. Foi selecionado o pacote BB, por ser uma implementação reconhecida e bastante utilizada da linguagem R para resolver essa classe de problemas, e foi inserida nessa implementação do DF-SANE a estratégia de aceleração.

Com os experimentos numéricos, pode-se afirmar que aplicar a estratégia de aceleração mostrou ser uma abordagem adequada que aumentou tanto a robustez, com um menor número de avaliações de função, quanto a eficiência, em termos de quantidade de iterações. Em especial, apresentou bons resultados quando comparado com a mesma versão do algoritmo sem a aceleração.

5.1 Trabalhos futuros

Do modo em que o objetivo deste trabalho foi implementar e testar uma estratégia de aceleração para o DF-SANE, ainda há possibilidades a serem exploradas, sendo listadas a seguir:

- Os testes podem ser realizados alterando a dimensão de cada problema, desse modo, é possível realizar uma comparação mais detalhada para problemas de uma mesma classe;
- Os testes também podem ser refeitos customizando os parâmetros de entrada do algoritmo. Os experimentos apresentados neste trabalho foram realizados apenas com os valores padrão, entretanto, conforme mostra os experimentos feitos por Varadhan e Gilbert (2009), alterar os valores iniciais permite que alguns problemas alcancem o critério de convergência;
- O problema HYDCAR20 apresentou um tempo de execução superior a 12 horas e por isso foi interrompido antes de apresentar resultados. Mas, como o problema não apresentou erros durante a execução, é possível que seja um comportamento específico desse teste. Neste trabalho essa questão não foi investigada;
- Conforme a análise de resultados mostra, o tempo de CPU pode ser aprimorado ao explorar estratégias de paralelização da decomposição QR.

Diante disso, o método DF-SANE acelerado manifestou robustez nos quesitos analisados e, à vista de suas propriedades, é um campo fértil para avanços.

Referências

BARZILAI, J.; BORWEIN, J. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, v. 8, n. 1, p. 141–148, 1988. Citado na página 17.

BIRGIN, E.; MARTíNEZ, J. M.; RAYDAN, M. Algorithm 813: Spg - software for convex-constrained optimization. *ACM Trans. Math. Softw.*, v. 27, p. 340–349, 09 2001. Citado na página 12.

BIRGIN, E. G. et al. *Accelerated derivative-free spectral residual method for nonlinear systems of equations*. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2104.13447>. Citado 2 vezes nas páginas 20 e 34.

BIRGIN, E. G.; MARTíNEZ, J. M. Secant acceleration of sequential residual methods for solving large-scale nonlinear systems of equations. *SIAM Journal on Numerical Analysis*, v. 60, n. 6, p. 3145–3180, 2022. Citado 10 vezes nas páginas 11, 12, 13, 15, 16, 17, 18, 19, 21 e 23.

CRUZ, W. A spectral algorithm for large-scale systems of nonlinear monotone equations. *Numerical Algorithms*, v. 76, 12 2017. Citado na página 11.

CRUZ, W. L.; MARTÍNEZ, J. M.; RAYDAN, M. Spectral residual method without gradient information for solving large-scale nonlinear systems of equations. *Math. Comput.*, v. 75, p. 1429–1448, 2006. Citado 4 vezes nas páginas 11, 16, 17 e 18.

CRUZ, W. L.; RAYDAN, M. Nonmonotone spectral methods for large-scale nonlinear systems. *Optimization Methods and Software*, v. 18, n. 5, p. 583–599, 2003. Disponível em: <https://doi.org/10.1080/10556780310001610493>. Citado 2 vezes nas páginas 16 e 17.

DENNIS, J. E.; SCHNABEL, R. B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1996. Disponível em: <https://epubs.siam.org/doi/abs/10.1137/1.9781611971200>. Citado 3 vezes nas páginas 11, 14 e 15.

DOLAN, E. D.; MORÉ, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming*, v. 91, n. 2, p. 201–213, 2002. Citado na página 34.

GILL, P. E.; MURRAY, W.; WRIGHT, M. H. *Practical Optimization*. London: Academic Press Inc, 1981. Citado 3 vezes nas páginas 11, 13 e 15.

GOULD, N. I. M.; ORBAN, D.; TOINT, P. L. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, v. 60, n. 3, p. 545–557, 2014. Citado na página 23.

GRIPPO, L.; LAMPARIELLO, F.; LUCIDI, S. A nonmonotone line search technique for newton's method. *SIAM Journal on Numerical Analysis*, v. 23, p. 707–716, 1986. Citado na página 16.

LUENGO, F.; RAYDAN, M. Gradient method with dynamical retards for large-scale optimization problems. *Electronic Transactions on Numerical Analysis*, v. 16, p. 186–193, 2003. Citado na página 19.

MELI, E. et al. *Solving nonlinear systems of equations via spectral residual methods: stepsize selection and applications*. arXiv, 2020. Disponível em: <https://arxiv.org/abs/2005.05851>. Citado na página 11.

MORÉ, J. J.; GARBOW, B. S.; HILLSTROM, K. E. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software (TOMS)*, v. 7, n. 1, p. 17–41, 1981. Citado na página 23.

NELDER, J.; MEAD, R. A simplex method for function minimization. *Computer Journal*, v. 7, p. 308, 1965. Citado na página 19.

NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. 2. ed. New York: Springer, 2006. Citado 3 vezes nas páginas 11, 14 e 23.

RAYDAN, M. *Marcos Raydan's Home Page*. 2009. <http://kuainasi.ciens.ucv.ve/ccct/mraydan/mraydan_pub.html>. Último acesso em 25 de Jan. de 2023. Citado na página 19.

VARADHAN, R.; GILBERT, P. Bb: An r package for solving a large system of nonlinear equations and for optimizing a high-dimensional nonlinear objective function. *Journal of Statistical Software*, v. 32, n. 4, p. 1–26, 2009. Disponível em: <http://www.jstatsoft.org/v32/i04/>. Citado 8 vezes nas páginas 11, 12, 13, 17, 19, 24, 27 e 39.

VARADHAN, R.; ROLAND, C. Simple and globally convergent methods for accelerating the convergence of any em algorithm. *Scandinavian Journal of Statistics*, v. 35, n. 2, p. 335–353, 2008. Citado na página 17.