

# Trabalho 01

## Otimização sem Restrições

MEC 2403 - Otimização e Algoritmos para Engenharia Mecânica

**Felipe da Costa Pereira**  
felipecostapereira@gmail.com

Professor: Ivan Menezes



Departamento de Engenharia Mecânica  
PUC-RJ Pontifícia Universidade Católica do Rio de Janeiro  
outubro, 2022

# 1 Introdução

Otimização sem restrição (OSR) consiste em encontrar o mínimo de uma função  $f(\vec{x})$  onde não há restrição em relação ao domínio das variáveis  $\vec{x}$ . A fim de se encontrar o mínimo da função  $f(\vec{x})$  a partir de um ponto de partida ( $\vec{x}_0$ ), os métodos apresentados nesse trabalho consistem na repetição das etapas seguintes até que um critério de parada seja atingido:

1. Selecionar uma direção  $\vec{d}$  a partir do ponto  $\vec{x}_0$
2. Encontrar o mínimo da função  $f$  nessa direção, chegando a um novo ponto  $\vec{x}_1 = \vec{x}_0 + \alpha\vec{d}$ , onde  $\alpha$  é um número real (busca linear).
3. Tomar  $\vec{x}_1$  como o novo ponto de partida  $\vec{x}_0$
4. Repetir os passos de 1 a 3 até que uma condição de parada seja atingida: mínimo encontrado ( $|\vec{\nabla}f| = 0$ ), ou máximo número de iterações atingido.

A etapa descrita no item 2 consiste na minimização de uma função de uma única variável ( $\alpha$ ). Dessa forma o problema de minimização da função  $f(\vec{x})$  se torna um problema de sucessivas determinações de direções de busca e suas respectivas buscas lineares nessas direções.

## 2 Objetivos

Os principais objetivos deste trabalho são:

- Implementar numericamente os algoritmos de otimização: Univariate, Powell, Steepest Descent, Fletcher-Reeves, Newton-Raphson e BFGS.
- Avaliar a influência dos parâmetros dos algoritmos nas métricas de convergência e comparar essas últimas com os valores esperados da teoria.
- Aplicar os algoritmos implementados na solução do problema de OSR em três casos: uma função quadrática, uma não quadrática e uma terceira função que representa um problema de engenharia (minimização da energia de um sistema massa-mola visando encontrar seu ponto de equilíbrio estático)

As funções a serem minimizadas neste trabalho são:

Problema 01a):

$$f(x_1, x_2) = x_1^2 - 3x_1x_2 + 4x_2^2 + x_1 - x_2 \quad (1a)$$

a partir dos pontos iniciais  $x^0 = \{2, 2\}^t$  e  $x^0 = \{-1, -3\}^t$

Problema 01b):

$$f(x_1, x_2) = (1 + a - bx_1 - bx_2)^2 + (b + x_1 + ax_2 - bx_1x_2)^2 \quad (1b)$$

com  $a = 10$  e  $b = 1$  a partir dos pontos iniciais  $x^0 = \{10, 2\}^t$  e  $x^0 = \{-2, -3\}^t$

Problema 02:

$$\Pi(x_1, x_2) = 450(\sqrt{(30 + x_1)^2 + x_2^2} - 30)^2 + 300(\sqrt{(30 - x_1)^2 + x_2^2} - 30)^2 - 360x_2 \quad (2)$$

a partir do ponto inicial  $x^0 = \{0.01, -0.10\}^t$

## 3 Algoritmos de Busca Linear

Os algoritmos de minimização são executados em duas etapas conforme citado anteriormente: a primeira etapa consiste em determinar uma direção de busca  $\vec{d}$  e em seguida promove-se a minimização da função  $f$  nessa direção, o que significa encontrar o valor de  $\alpha = \alpha_k$  que minimiza a função  $f(x = x_0 + \alpha\vec{d})$  ao longo da direção  $\vec{d}$ . A partir do valor de  $\alpha_k$  encontra-se um novo ponto de partida  $f(x_{k+1} = x_k + \alpha_k\vec{d}_k)$ . Uma nova direção  $\vec{d}_{k+1}$  é então determinada e o processo se repete até que uma condição de parada seja atingida.

Neste trabalho, a busca linear é realizada em duas etapas: Passo constante e refinamento do cálculo de  $\alpha$  através do método da seção áurea.

Um maior detalhamento dos métodos de busca aqui apresentados, assim como outros comumente utilizados na solução de problemas de otimização pode ser encontrado em [Menezes et al., 2012].

### 3.1 Passo Constante

A primeira etapa da busca linear consiste numa busca inexata que visa encontrar um intervalo de valores do passo  $\alpha$ ,  $[\alpha_L, \alpha_H]$  que represente uma redução suficientemente grande da função  $f$ . Essa implementação numérica incrementa o passo de um valor  $d\alpha$  até que a função pare de diminuir.

### 3.2 Bisseção e Seção Áurea

Após a etapa do passo constante, os métodos da Bisseção ou Seção Áurea são aplicados para encontrar o valor de  $\alpha$ , entre os valores determinados no intervalo da etapa 1. Em ambos os casos, o intervalo de ocorrência do valor mínimo de  $f$  é sucessivamente reduzido até que seja muito pequeno e considera-se, dado esse critério numérico, solucionado o problema da minimização de  $f$  nessa direção, encontrando o passo  $\alpha_k$  correspondente a esse mínimo.

O método da bisseção divide sucessivamente o intervalo descartando a parte superior ou inferior do mesmo avaliando o valor da função  $f$  na vizinhança esquerda e direita de um  $\alpha_M$  médio do intervalo. Já o método da seção áurea utiliza a razão áurea para descarte dos intervalos onde  $f$  aumenta. Este último método realiza mais passos, porém possui a vantagem de avaliar menos vezes a função  $f$  uma vez que os valores avaliados no passo anterior podem ser reutilizados no passo seguinte do algoritmo, o que pode ser interessante em problemas onde a avaliação da função  $f$  é computacionalmente cara.

## 4 Algoritmos de Direção

Conforme descrito anteriormente, os métodos de direção determinam direções de busca do mínimo de  $f$  a partir de um ponto  $\vec{x}_k$ . A partir da busca linear encontra-se o valor do passo nessa direção que minimiza  $f$ ,  $(\alpha_k)$ , os algoritmos de direção calculam, então, a próxima direção onde se deve minimizar  $f$  e a partir do novo ponto  $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{d}$ .

Os algoritmos de determinação das direções de busca utilizados nesse trabalho são brevemente descritos a seguir.

Um maior detalhamento dos algoritmos de direção aqui apresentados, assim como outros comumente utilizados na solução de problemas de otimização pode ser encontrado em [Menezes et al., 2012].

### 4.1 Univariante

O método univariante é o mais simples, onde as direções de busca são as direções canônicas,  $\vec{d}_k = \vec{e}_k$ . Esse procedimento é equivalente a modificar uma variável de cada vez no processo iterativo, ou seja, apenas a variável na posição  $k$  do vetor de variáveis  $\vec{x}$  é modificada na iteração  $k$  ([Menezes et al., 2012]).

### 4.2 Powell

O método de Powell utiliza em seu algoritmo direções denominadas "movimento padrão". O método de Powell gera direções Q-conjugadas, o que representa uma aceleração em relação ao método univariante. Um maior detalhamento sobre o método de Powell e a demonstração de que o método de Powell converge para o mínimo de uma função quadrática de  $n$  variáveis em um número finito de passos dado por  $(n + 1)^2$  está descrito em [Menezes et al., 2012].

### 4.3 Steepest Descent

O método Steepest Descent é um método onde as direções são dadas pela direção oposta ao gradiente da função  $f$ . Ou seja,  $\vec{d}_k = -\vec{\nabla} f(\vec{x}_k)$ .

### 4.4 Fletcher-Reeves

Este método consiste em uma adaptação do método dos Gradientes Conjugados que o torna capaz de ser usado para minimização de uma função qualquer. Para tanto, duas modificações precisam ser realizadas, a avaliação da matriz Q (ou Hessiana para função não quadrática) é substituída por uma busca linear e o parâmetro  $\beta$  modificado ([Menezes et al., 2012]).

## 4.5 Newton-Raphson

O princípio deste método é minimizar uma função  $f$  através de uma aproximação local por uma função quadrática. A direção de minimização de  $f$  é obtida a partir da derivada da expansão de  $f$  numa série de Taylor de ordem 2 em relação a  $\vec{d}_k$ , de onde se obtém:  $\vec{d}_k = -H^{-1}\vec{\nabla}f(\vec{x}_k)$ , onde  $H$  é a matriz Hessiana da função  $f$  ([Menezes et al., 2012]).

## 4.6 BFGS

O método Broyden-Fletcher-Goldfarb-Shanno (BFGS) é um método para resolver um problema de otimização não linear sem restrições. trata-se de uma solução frequentemente usada quando se deseja um algoritmo com direções de descida.

A ideia principal deste método é evitar a construção explícita da matriz Hessiana e, em vez disso, construir uma aproximação da inversa da segunda derivada da função a ser minimizada, analisando os diferentes gradientes sucessivos. Esta aproximação das derivadas da função leva a um método quase-Newton (uma variante do método de Newton) para encontrar a direção de busca. A matriz Hessiana não precisa ser recalculada a cada iteração do algoritmo. No entanto, o método assume que a função pode ser aproximada localmente por uma expansão quadrática limitada em torno do ótimo ([Broyden et al., ]).

## 5 Metodologia

Para atingir os objetivos do trabalho, foram programados scripts em linguagem Matlab, organizados conforme esquematizado na figura 1.

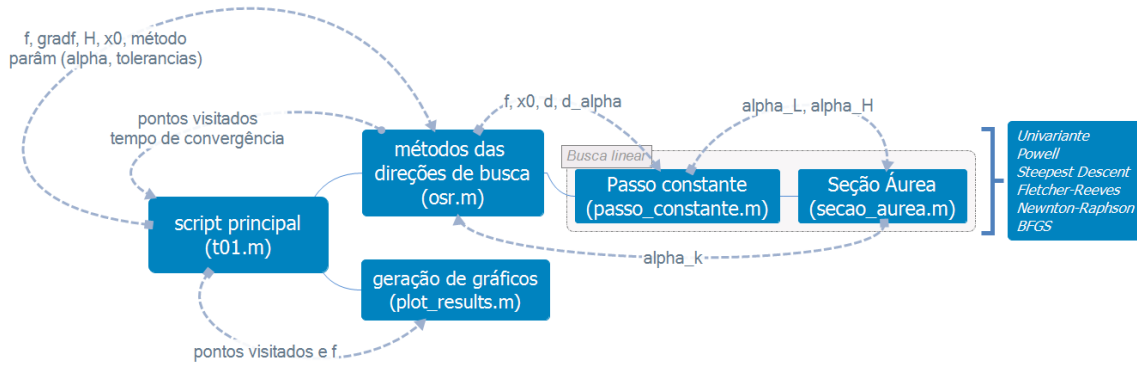


Figura 1: Fluxo dos scripts

Um script principal *t01.m* escolhe os parâmetros do algoritmo:  $a = d\alpha$  (passo da busca linear), máximo número de iterações (*iter\_max*) e tolerâncias para avaliar a convergência da função e da seção áurea: *TOL* e *TOL2*, respectivamente. Além disso esse script cria as funções, seus gradientes, matriz Hessiana e pontos iniciais, a serem passados como parâmetros para o script que implementa os algoritmos, conforme ilustrado no anexo 1.

Em seguida, o script *t01.m* chama o script *osr.m* (anexo 2) para cada método e plota as curvas de nível da função  $f$  e todos os pontos visitados durante a busca do algoritmo através do script *plot\_result.m*.

O script *osr.m* implementa de fato os algoritmos a partir dos parâmetros recebidos, retornando todos os valores de  $\vec{x}$  visitados e o tempo de execução, conforme o anexo 3. O último dos pontos visitados durante a execução do algoritmo será o ponto da convergência  $x_{min}$ .

Outros dois scripts invocados na solução dos problemas propostos são *passo\_constante.m* e *secao\_aurea.m*, que realizam a etapa de busca linear para cada direção de busca. Esses códigos encontram-se nos anexos 4 e 5, respectivamente. O script *plot\_result.m* é listado no anexo 6.

A saída da execução do script principal *t01.m* está listada no anexo 7.

## 6 Resultados

Visando a convergência de todos os métodos, foram testados alguns valores dos parâmetros dos algoritmos. A melhor combinação dos parâmetros, para as funções dos problemas 1 e 2 em termos de convergência dos algoritmos está listada na tabela 1.

**Tabela 1:** Melhores parâmetros para convergência dos algoritmos

|                | $d\alpha$ | TOL(gradiente)     | TOL2(busca linear) | max.iter |
|----------------|-----------|--------------------|--------------------|----------|
| Função 1a e 1b | 0.002     | $10^{-4}$          | $10^{-7}$          | 100      |
| Função 2       | 0.005     | $5 \times 10^{-4}$ | $10^{-7}$          | 100      |

As tabelas 2, 3 apresentam os principais resultados dos algoritmos implementados para as funções dos itens 1a e 1b. Além dos pontos de mínimo encontrado, as tabelas apresentam também o número de passos para a convergência e o tempo de execução.

**Tabela 2:** Resultados para a função 1a

| método           | $x^0$          | $x^{min}$                | passos | $\Delta t(\text{ms})$ |
|------------------|----------------|--------------------------|--------|-----------------------|
| Univariante      | $\{2, 2\}^t$   | $\{-0.7142, -0.1428\}^t$ | 34     | 4.8                   |
| Univariante      | $\{-1, -3\}^t$ | $\{-0.7144, -0.1429\}^t$ | 36     | 6.9                   |
| Powell           | $\{2, 2\}^t$   | $\{-0.7143, -0.1429\}^t$ | 6      | 24.8                  |
| Powell           | $\{-1, -3\}^t$ | $\{-0.7143, -0.1429\}^t$ | 12     | 7.2                   |
| Steepest Descent | $\{2, 2\}^t$   | $\{-0.7142, -0.1428\}^t$ | 25     | 6.5                   |
| Steepest Descent | $\{-1, -3\}^t$ | $\{-0.7143, -0.1429\}^t$ | 7      | 3.0                   |
| Fletcher-Reeves  | $\{2, 2\}^t$   | $\{-0.7143, -0.1429\}^t$ | 2      | 2.5                   |
| Fletcher-Reeves  | $\{-1, -3\}^t$ | $\{-0.7143, -0.1429\}^t$ | 2      | 1.7                   |
| Newton-Raphson   | $\{2, 2\}^t$   | $\{-0.7143, -0.1429\}^t$ | 1      | 2.5                   |
| Newton-Raphson   | $\{-1, -3\}^t$ | $\{-0.7143, -0.1429\}^t$ | 1      | 1.1                   |
| BFGS             | $\{2, 2\}^t$   | $\{-0.7143, -0.1429\}^t$ | 2      | 2.2                   |
| BFGS             | $\{-1, -3\}^t$ | $\{-0.7143, -0.1429\}^t$ | 2      | 1.5                   |

No caso da função do item 1a, o método de Powell levou mais passos do que o esperado em teoria para uma função quadrática, já que nesse caso é esperado que o mesmo atinja a convergência em até 9 passos.

**Tabela 3:** Resultados para a função 1b

| método           | $x^0$          | $x^{min}$                  | passos | $\Delta t(\text{ms})$ |
|------------------|----------------|----------------------------|--------|-----------------------|
| Univariante      | $\{10, 2\}^t$  | $\{13.0000, 4.0000\}^t$    | 45     | 9.9                   |
| Univariante      | $\{-2, -3\}^t$ | $\{7.0000, -2.0000\}^t$    | 45     | 10.8                  |
| Powell           | $\{10, 2\}^t$  | $\{13.0000, 4.0000\}^t$    | 24     | 11.6                  |
| Powell           | $\{-2, -3\}^t$ | $\{7.0000, -2.0000\}^t$    | 18     | 8.0                   |
| Steepest Descent | $\{10, 2\}^t$  | $\{13.0000, 4.0000\}^t$    | 46     | 2.3                   |
| Steepest Descent | $\{-2, -3\}^t$ | $\{7.0000, -2.0000\}^t$    | 59     | 2.6                   |
| Fletcher-Reeves  | $\{10, 2\}^t$  | $\{13.0000, 4.0000\}^t$    | 41     | 1.6                   |
| Fletcher-Reeves  | $\{-2, -3\}^t$ | $\{7.0000, -2.0000\}^t$    | 16     | 0.9                   |
| Newton-Raphson   | $\{10, 2\}^t$  | $\{10.0000, 1.0000\}^t(*)$ | 1      | 0.9                   |
| Newton-Raphson   | $\{-2, -3\}^t$ | $\{12.9999, 4.0001\}^t$    | 6      | 4.1                   |
| BFGS             | $\{10, 2\}^t$  | $\{13.0000, 4.0000\}^t$    | 7      | 4.1                   |
| BFGS             | $\{-2, -3\}^t$ | $\{7.0000, -2.0000\}^t$    | 6      | 5.7                   |

No caso da função do item 1b nota-se que o método de Newton-Raphson convergiu para um valor diferente dos demais métodos (\*). A primeira direção de descida foi  $\vec{d}_0 = [0, -1]^t$ . A busca linear, assim, resultou no ponto  $x^1 = [10, 1]^t$ .

Nesse ponto, temos a norma do gradiente de  $f$ :

$$|\vec{\nabla} f_{1b}([10, 1])| = 0$$

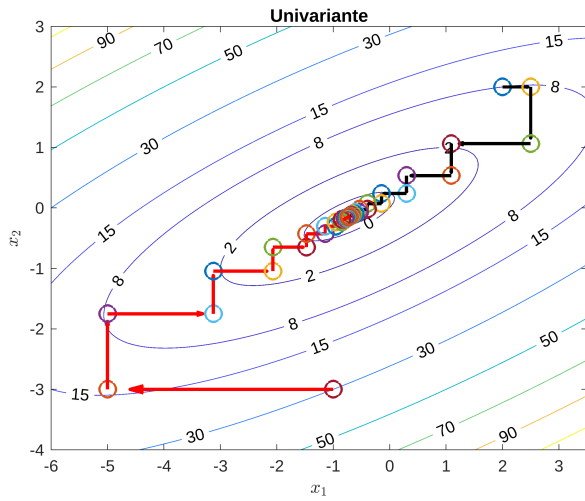
e a matriz hessiana de  $f$ :

$$H_{1b}([10, 1]) = \begin{bmatrix} 2 & -20 \\ -20 & 2 \end{bmatrix}$$

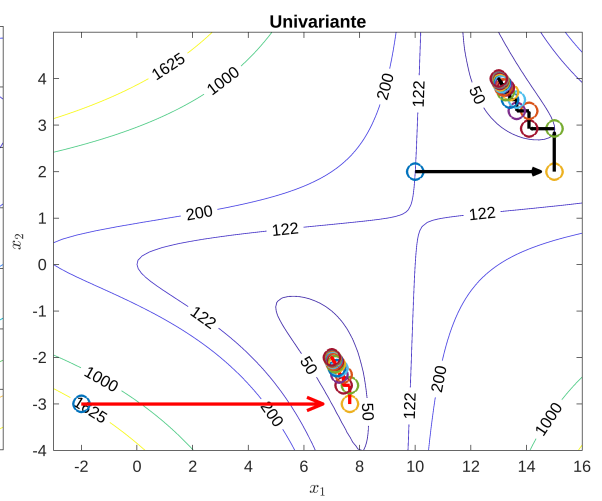
Sendo os autovalores de  $H_{1b}$  iguais a  $\lambda_1 = -18$  e  $\lambda_2 = 22$  positivo e negativo, esse ponto é um ponto crítico mas não um ponto de mínimo da função  $f$  mas sim um ponto de sela.

As curvas de nível de  $f$ , assim como o ponto crítico encontrado pode ser visualizado também graficamente na figura 6b.

As figuras de 2 a 7 a seguir ilustram as curvas de nível de  $f$  para os casos das funções dos itens 1a e 1b, bem como a trajetória dos pontos  $\vec{x}_k$  associadas ao algoritmo de minimização.

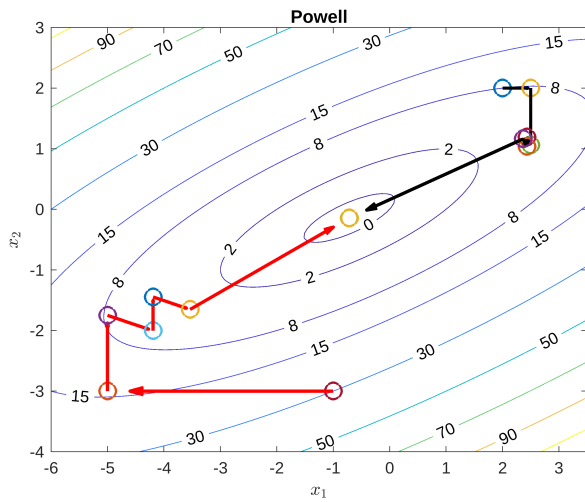


(a) Curvas de nível de  $f_{1a}$  e pontos  $x_k$

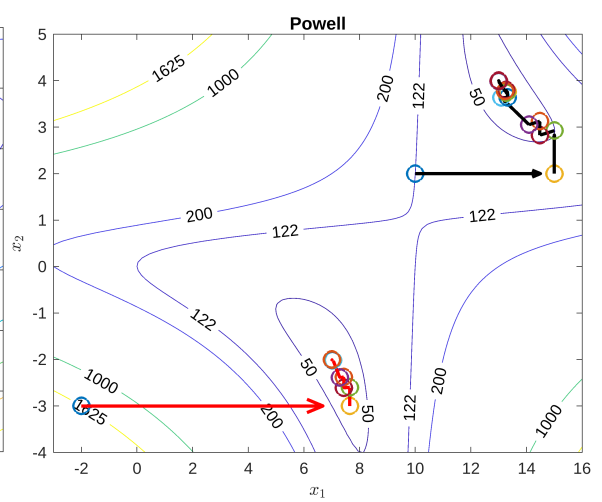


(b) Curvas de nível de  $f_{1b}$  e pontos  $x_k$

**Figura 2:** Resultados do método univariante, para as duas funções e pontos iniciais

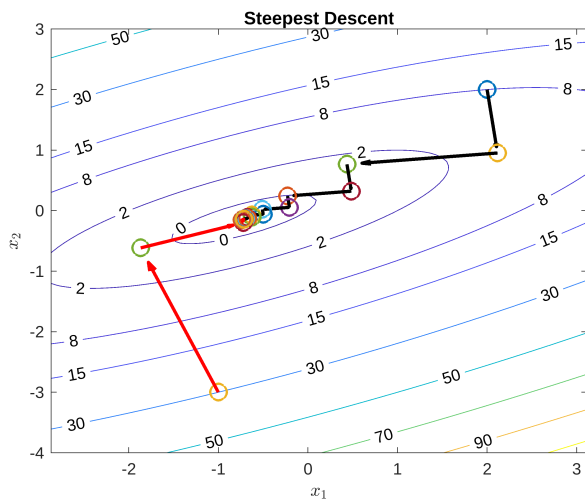


(a) Curvas de nível de  $f_{1a}$  e pontos  $x_k$

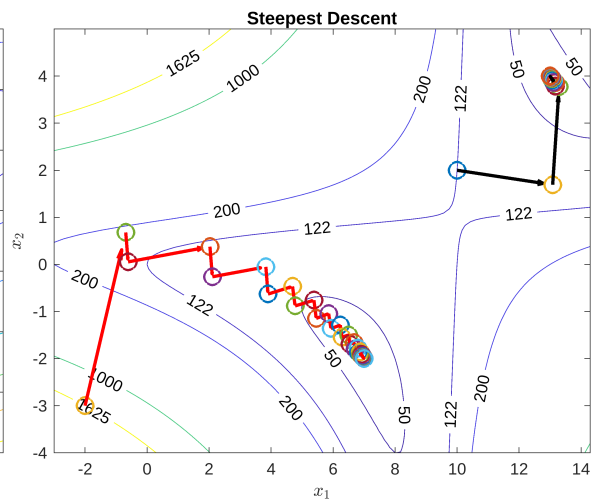


(b) Curvas de nível de  $f_{1b}$  e pontos  $x_k$

**Figura 3:** Resultados do método de Powell, para as duas funções e pontos iniciais

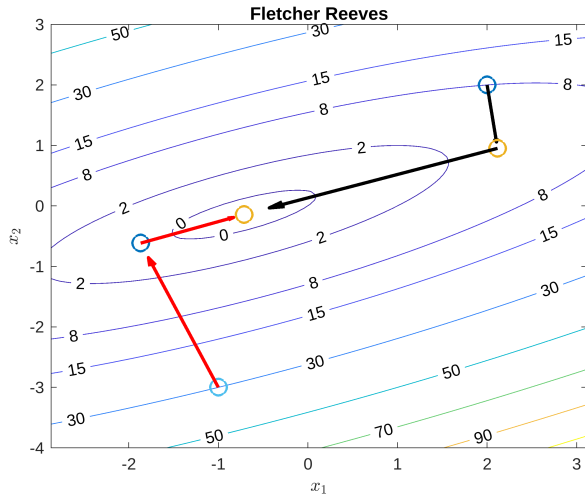


(a) Curvas de nível de  $f_{1a}$  e pontos  $x_k$

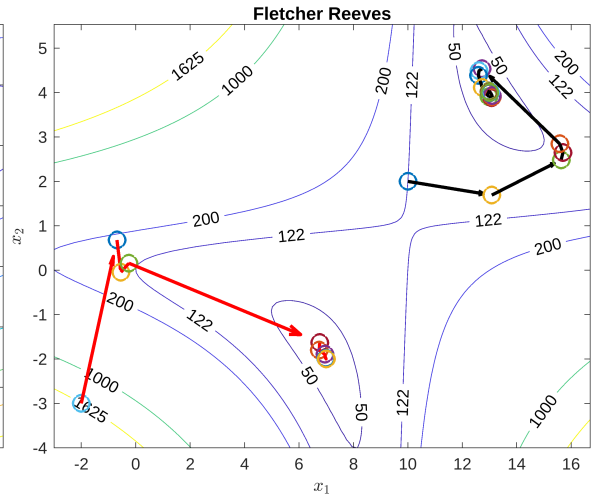


(b) Curvas de nível de  $f_{1b}$  e pontos  $x_k$

**Figura 4:** Resultados do método Steepest Descent, para as duas funções e pontos iniciais

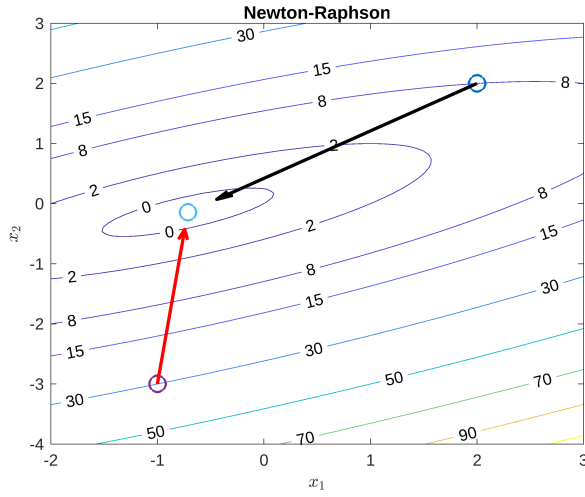


(a) Curvas de nível de  $f_{1a}$  e pontos  $x_k$

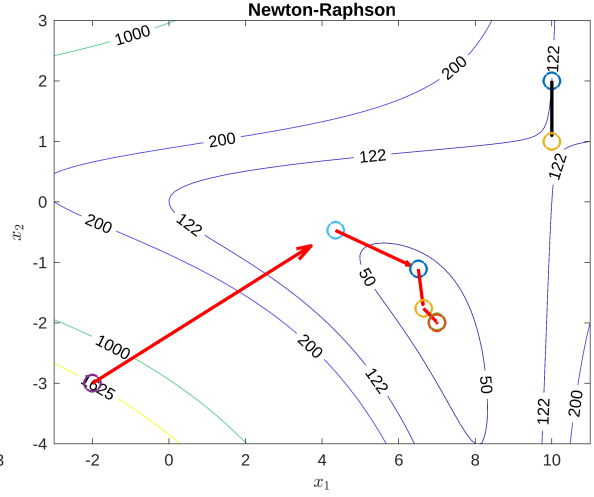


(b) Curvas de nível de  $f_{1b}$  e pontos  $x_k$

**Figura 5:** Resultados do método Fletcher-Reeves, para as duas funções e pontos iniciais

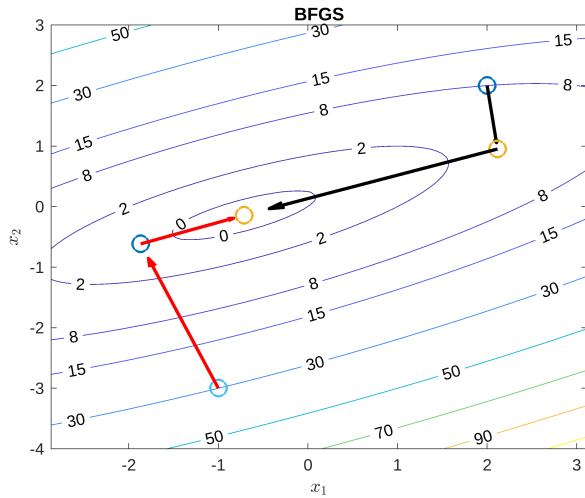


(a) Curvas de nível de  $f_{1a}$  e pontos  $x_k$

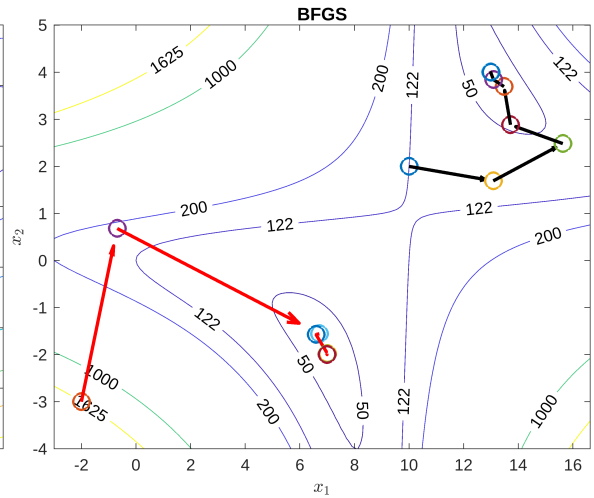


(b) Curvas de nível de  $f_{1b}$  e pontos  $x_k$

**Figura 6:** Resultados do método Newton-Raphson, para as duas funções e pontos iniciais



(a) Curvas de nível de  $f_{1a}$  e pontos  $x_k$



(b) Curvas de nível de  $f_{1b}$  e pontos  $x_k$

**Figura 7:** Resultados do método BFGS, para as duas funções e pontos iniciais

A tabela 4 apresenta os principais resultados dos algoritmos implementados para a função do item 2. Além

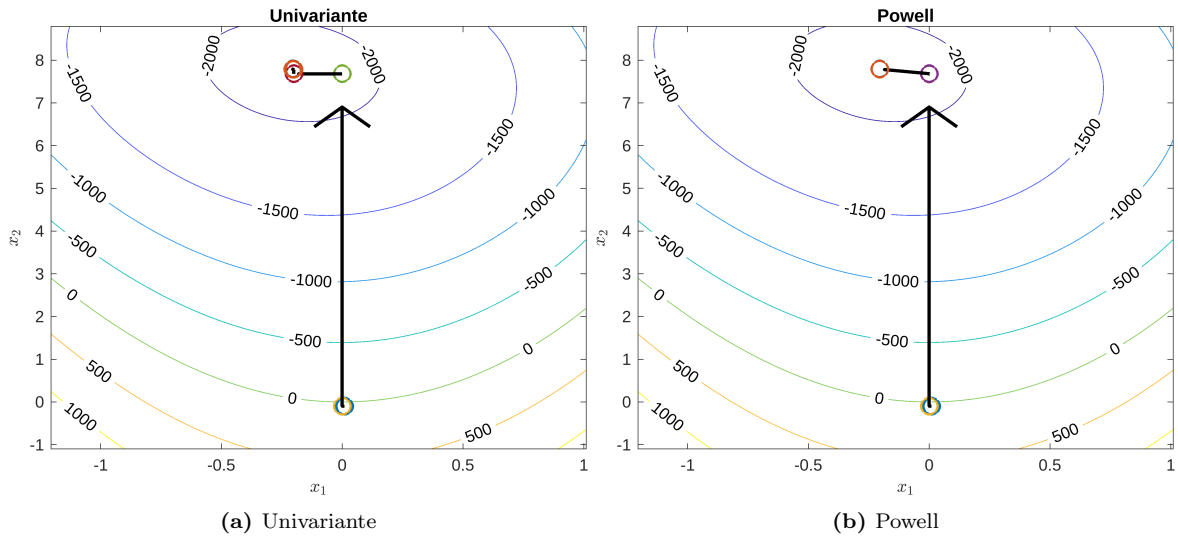
dos pontos de mínimo encontrados, as tabelas apresentam também o número de passos para a convergência e o tempo de execução.

**Tabela 4:** Resultados para a função 2

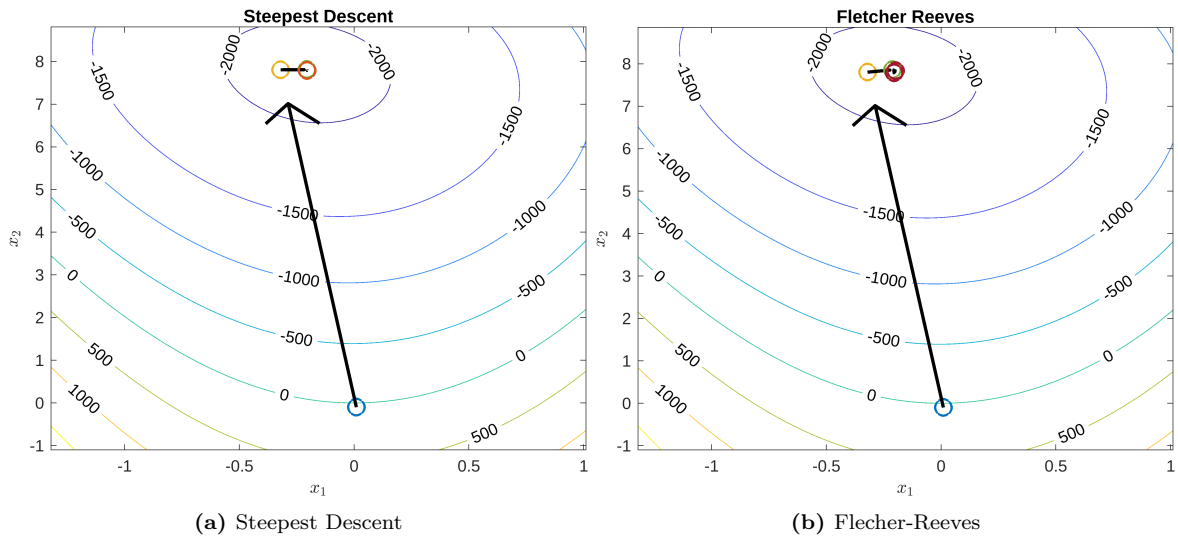
| método           | $x^0$               | $x^{min}$             | passos | $\Delta t(\text{ms})$ |
|------------------|---------------------|-----------------------|--------|-----------------------|
| Univariante      | $\{0.01, -0.10\}^t$ | $\{-0.205, 7.789\}^t$ | 11     | 2.1                   |
| Powell           | $\{0.01, -0.10\}^t$ | $\{-0.205, 7.789\}^t$ | 12     | 6340.8(**)            |
| Steepest Descent | $\{0.01, -0.10\}^t$ | $\{-0.205, 7.789\}^t$ | 6      | 0.3                   |
| Fletcher-Reeves  | $\{0.01, -0.10\}^t$ | $\{-0.205, 7.789\}^t$ | 10     | 0.4                   |
| Newton-Raphson   | $\{0.01, -0.10\}^t$ | $\{-0.205, 7.789\}^t$ | 3      | 4.4                   |
| BFGS             | $\{0.01, -0.10\}^t$ | $\{-0.205, 7.789\}^t$ | 3      | 0.3                   |

No caso da função do item 2 nota-se que o elevado tempo de execução do método de Powell (\*\*) em relação aos demais.

A figuras 8, 9 e 10 a seguir ilustram, para os seis métodos, as curvas de nível de  $f$  para os casos da função do item 2, bem como a trajetória dos pontos  $x_k$  associadas ao algoritmo de minimização.

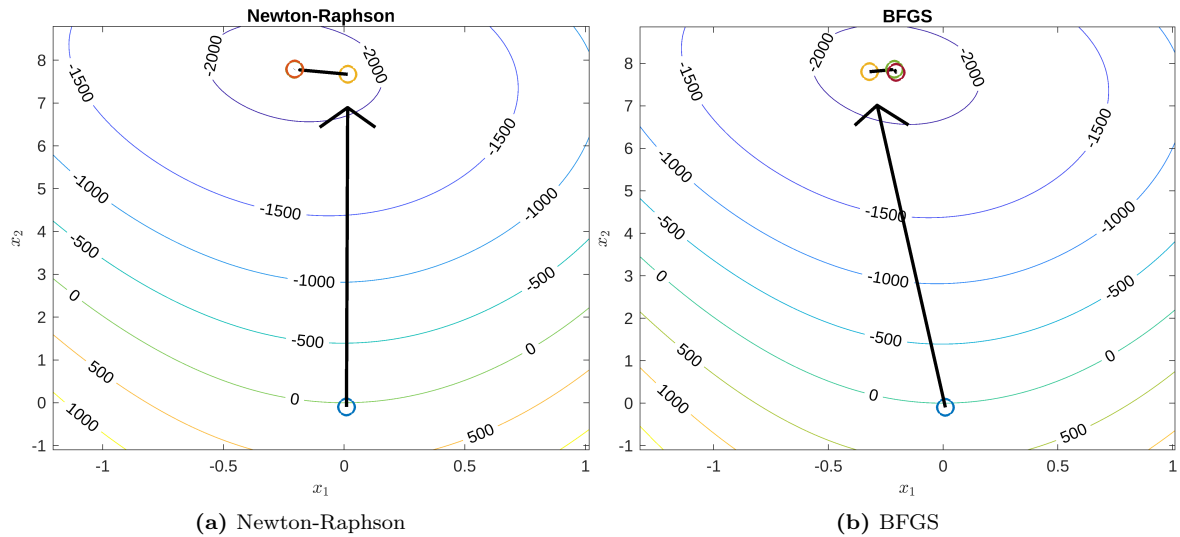


**Figura 8:** Curvas de nível de  $f_2$  e pontos  $x_k$ , métodos de ordem zero



**Figura 9:** Curvas de nível de  $f_2$  e pontos  $x_k$ , métodos de ordem um





**Figura 10:** Curvas de nível de  $f_2$  e pontos  $x_k$ , métodos de Newton-Raphson e BFGS

## 7 Conclusões

A realização deste trabalho permitiu implementar os métodos estudados na disciplina de otimização. Para todos os casos de aplicação, foram rodados os algoritmos para outros pontos que não os pontos propostos no problema em si e cujo resultado mostrou que os algoritmos de busca e minimização são robustos, mesmo nos casos de funções não quadráticas.

Em alguns poucos casos o algoritmo não obteve o resultado esperado em relação ao número de passos até a convergência. Em geral, notou-se que a convergência, número de passos e o tempo de execução dos algoritmos é muito sensível a escolha de seus parâmetros, o que reside nas incertezas numéricas associadas aos métodos de busca linear, principalmente.

Em um dos métodos (Newton-Raphson) foi encontrado um ponto de sela e não um ponto de mínimo, mas este é um comportamento esperado do algoritmo já que o mesmo se propõe apenas a identificar os pontos críticos, candidatos a mínimo de  $f$ , sem entrar na avaliação da hessiana.

## 8 Anexos

A seguir estão ilustrados alguns dos códigos ou trechos de códigos decritos na seção metodologia.

### Anexo 1: script t01.m setando parâmetros e criando as funções

```
% dados do item 01a, f, grad f, hess f e x0
fa = @(x) x(1)^2-3*x(1)*x(2)+4*x(2)^2+x(1)-x(2);
gfa = @(x) [2*x(1)-3*x(2)+1 ; -3*x(1)+8*x(2)-1];
Ha = @(x) [2 -3; -3 8];
x01 = [2;2];
x02 = [-1;-3];
% parametros dos algoritmos
iter_max = 100;
a = 0.002; % passo
TOL = 1e-4; % parada do gradiente
TOL2 = 1e-7; % busca linear
methods = ["Univariante","Powell","Steepest Descent","Fletcher Reeves","Newton -
Raphson","BFGS"];
```

**Anexo 2:** script t01.m chamando o script osr.m para a função do item 1a para cada um dos 6 métodos estudados

```
fprintf('\n*****ITEM_01A*****\n');
for method = 1:6
    fprintf('---%s---\n', methods(method));

    fprintf('x0=[%2d,%2d]:\n',x01(1), x01(2));
    [x_1,t] = osr (fa, gfa, Ha, x01, method, iter_max, a, TOL, TOL2);
    fprintf(' (%.1fms),xmin=[%0.4f,%0.4f],f=%0.4f\n', t*1000, x_1(1,end), x_1(2,
        end), fa(x_1(:,end)));

    fprintf('x0=[%2d,%2d]:\n',x02(1), x02(2));
    [x_2,t] = osr (fa, gfa, Ha, x02, method, iter_max, a, TOL, TOL2);
    fprintf(' (%.1fms),xmin=[%0.4f,%0.4f],f=%0.4f\n', t*1000, x_2(1,end), x_2(2,
        end), fa(x_2(:,end)));

    plot_result(min([x_1(1,:), x_2(1,:)]) - dx, max([x_1(1,:), x_2(1,:)]) + dx, min([x_1
        (2,:), x_2(2,:)]) - dx, max([x_1(2,:), x_2(2,:)]) + dx, x_1, x_2, methods(
        method), 1)
    exportgraphics(gcf, strcat('./figures/img01A_m0', num2str(method), '.png'), '
        Resolution', 500)
end
```

### Anexo 3: script osr.m implementando o método de Powell

```
function [x_,time_elap] = osr (f, gf, H, x0, method, iter_max, a, TOL, TOL2)
% 1. Univariante
% 2. Powell
% 3. Steepest Descent
% 4. Fletcher-Reeves
% 5. Newton-Raphson
% 6. BFGS

k=0;
conv=0; %flag convergencia
tstart = tic;
switch method
case 2
    % 2. Powell
    x_ = x0;
    x = x0;
    while k < iter_max
        j = 1;
        n = 2;
        y = [[1;0],[0;1]];
        while j <= n
            [alpha_L, alpha_H] = passo_constante(f, x, y(:,1), a);
            alpha_k = secas_aurea(f, x, y(:,1), TOL2, alpha_L, alpha_H);
            k=k+1;
            x = x + alpha_k*y(:,1);
            x_ = [x_,x];
            [alpha_L, alpha_H] = passo_constante(f, x, y(:,2), a);
            alpha_k = secas_aurea(f, x, y(:,2), TOL2, alpha_L, alpha_H);
            k=k+1;
            x = x + alpha_k*y(:,2);
            x_ = [x_,x];
            d = x-x0;
            [alpha_L, alpha_H] = passo_constante(f, x, d, a);
            alpha_k = secas_aurea(f, x, d, TOL2, alpha_L, alpha_H);
            k=k+1;
            x0 = x + alpha_k*d;
            x=x0;
            x_ = [x_,x];

            y(:,1) = y(:,2);
            y(:,2) = d;

            j = j+1;
        end
        if norm(gf(x)) < TOL
            fprintf('%d steps!', k);
            conv=1;
            break;
        end
    end
end
if conv == 0
    fprintf('Nao convergiu apos %d steps', k);
end
```

#### Anexo 4: script passo\_constante.m

```
function [alpha_L, alpha_H] = passo_constante(f, x0, d, a)
    alpha = 0;
    f_min = Inf;
    f_val = f(x0);
    alphas = [];
    f1 = f(x0 - a*d);
    f2 = f(x0 + a*d);
    if f1 < f2
        a=-a; % desce a esq (d-)
    end
    while f_val <= f_min
        x = x0 + alpha * d;
        f_val = f(x);
        if f_val < f_min
            f_min = f_val;
        end
        alphas = [alphas; alpha];
        alpha = alpha + a;
    end
    alpha_L = alphas(end-1);
    alpha_H = alphas(end);
    if a < 0
        alpha_H = alphas(end-1);
        alpha_L = alphas(end);
    end
end
```

#### Anexo 5: script secao\_aurea.m

```
function alpha_k = secao_aurea (f, x0, d, TOL, alpha_L, alpha_H)
    ra = (sqrt(5)-1)/2;
    b = norm(alpha_L-alpha_H);
    alpha_E = alpha_L + (1-ra)*b;
    alpha_D = alpha_L + ra*b;
    f1 = f(x0 + alpha_E * d);
    f2 = f(x0 + alpha_D * d);
    while b > TOL
        if f1 > f2
            alpha_L = alpha_E;
            alpha_E = alpha_D;
            b = norm(alpha_L-alpha_H);
            alpha_D = alpha_L + ra*b;
            % avaliar menos vezes a funcao f
            f1 = f2;
            f2 = f(x0 + alpha_D * d);
        else
            alpha_H = alpha_D;
            alpha_D = alpha_E;
            b = norm(alpha_L-alpha_H);
            alpha_E = alpha_L + (1-ra)*b;
            % avaliar menos vezes a funcao f
            f2 = f1;
            f1 = f(x0 + alpha_E * d);
        end
    end
    alpha_k = (alpha_L+alpha_H)/2;
end
```

# Anexo 6: script plot\_result.m

```
function [] = plot_result(xmin, xmax, ymin, ymax, x_, x2_, graph_title, func)
    n = length(x_);
    n2 = length(x2_);
    figure
    x1 = linspace(xmin, xmax, 100);
    x2 = linspace(ymin, ymax, 100);
    [x1,x2] = meshgrid(x1,x2);
    if func == 1
        fplot = x1.^2 - 3*x1.*x2 + 4*x2.^2 + x1 - x2;
        contour(x1, x2, fplot, [0 2 8 15 30:20:120], 'ShowText','on');
    elseif func == 2
        fplot = (11-x1-x2).^2 + (1+x1+10*x2-x1.*x2).^2;
        contour(x1, x2, fplot, [50 122 200 1000 1625 5000 10000], 'ShowText','on')
    else
        fplot = 450*(sqrt((30+x1).^2+x2.^2)-30).^2+300*(sqrt((30-x1).^2+x2.^2)-30)
            .^2-360*x2;
        contour(x1, x2, fplot, 'ShowText','on')
    end
    title(graph_title)
    xlabel('$x_{1}$', 'Interpreter', 'latex')
    ylabel('$x_{2}$', 'Interpreter', 'latex')
    hold on
    for k = 1:n
        plot(x_(1,k), x_(2,k),'o', 'LineWidth', 2, 'MarkerSize', 10)
        if k<n
            desenha_flecha(x_(:,k)', x_(:,k+1)', 'k');
        end
    end
    if ~ isempty(x2_)
        for k = 1:n2
            plot(x2_(1,k), x2_(2,k),'o', 'LineWidth', 2, 'MarkerSize', 10)
            if k<n2
                desenha_flecha(x2_(:,k)', x2_(:,k+1)', 'r');
            end
        end
    end
end
```

## Anexo 7: resultado da execução do script

```
***** ITEM 01A *****
---Univariante---
x0=[ 2, 2]: 34 steps!(4.8ms), xmin=[-0.7142,-0.1428], f=-0.2857
x0=[-1,-3]: 36 steps!(6.9ms), xmin=[-0.7144,-0.1429], f=-0.2857
---Powell---
x0=[ 2, 2]: 6 steps!(24.8ms), xmin=[-0.7143,-0.1429], f=-0.2857
x0=[-1,-3]: 12 steps!(7.2ms), xmin=[-0.7143,-0.1429], f=-0.2857
---Steepest Descent---
x0=[ 2, 2]: 25 steps!(6.5ms), xmin=[-0.7142,-0.1428], f=-0.2857
x0=[-1,-3]: 7 steps!(3.0ms), xmin=[-0.7143,-0.1429], f=-0.2857
---Fletcher Reeves---
x0=[ 2, 2]: 2 steps!(2.5ms), xmin=[-0.7143,-0.1429], f=-0.2857
x0=[-1,-3]: 2 steps!(1.7ms), xmin=[-0.7143,-0.1429], f=-0.2857
---Newton-Raphson---
x0=[ 2, 2]: 1 steps!(2.5ms), xmin=[-0.7143,-0.1429], f=-0.2857
x0=[-1,-3]: 1 steps!(1.1ms), xmin=[-0.7143,-0.1429], f=-0.2857
---BFGS---
x0=[ 2, 2]: 2 steps!(2.2ms), xmin=[-0.7143,-0.1429], f=-0.2857
x0=[-1,-3]: 2 steps!(1.5ms), xmin=[-0.7143,-0.1429], f=-0.2857

***** ITEM 01B *****
---Univariante---
x0=[10, 2]: 45 steps!(9.9ms), xmin=[13.0000,4.0000], f=40.0
x0=[-2,-3]: 45 steps!(10.8ms), xmin=[7.0000,-2.0000], f=40.0
---Powell---
x0=[10, 2]: 24 steps!(11.6ms), xmin=[13.0000,4.0000], f=40.0
x0=[-2,-3]: 18 steps!(8.0ms), xmin=[7.0000,-2.0000], f=40.0
---Steepest Descent---
x0=[10, 2]: 46 steps!(2.3ms), xmin=[13.0000,4.0000], f=40.0
x0=[-2,-3]: 59 steps!(2.6ms), xmin=[7.0000,-2.0000], f=40.0
---Fletcher Reeves---
x0=[10, 2]: 41 steps!(1.6ms), xmin=[13.0000,4.0000], f=40.0
x0=[-2,-3]: 16 steps!(0.9ms), xmin=[7.0000,-2.0000], f=40.0
---Newton-Raphson---
x0=[10, 2]: 1 steps!(0.9ms), xmin=[10.0000,1.0000], f=121.0
x0=[-2,-3]: 6 steps!(4.1ms), xmin=[7.0000,-2.0000], f=40.0
---BFGS---
x0=[10, 2]: 7 steps!(4.1ms), xmin=[12.9999,4.0001], f=40.0
x0=[-2,-3]: 6 steps!(5.7ms), xmin=[7.0000,-2.0000], f=40.0

***** ITEM 02 *****
---Univariante---
x0=[0.01,-0.10]: 11 steps!(2.1ms), xmin=[-0.205,7.789], f=-2091.7
---Powell---
x0=[0.01,-0.10]: 12 steps!(6340.8ms), xmin=[-0.205,7.789], f=-2091.7
---Steepest Descent---
x0=[0.01,-0.10]: 6 steps!(0.3ms), xmin=[-0.205,7.789], f=-2091.7
---Fletcher Reeves---
x0=[0.01,-0.10]: 10 steps!(0.4ms), xmin=[-0.205,7.789], f=-2091.7
---Newton-Raphson---
x0=[0.01,-0.10]: 3 steps!(4.4ms), xmin=[-0.205,7.789], f=-2091.7
---BFGS---
x0=[0.01,-0.10]: 3 steps!(0.3ms), xmin=[-0.205,7.789], f=-2091.7
```

## Referências

- [Broyden et al.,] Broyden, C. G., Fletcher, R., Goldfarb, D., and Shanno, D. F. Método de broyden-fletcher-goldfarb-shanno. [https://pt.frwiki.wiki/wiki/M%C3%A9thode\\_de\\_Broyden-Fletcher-Goldfarb-Shanno#R%C3%A9f%C3%A9rences](https://pt.frwiki.wiki/wiki/M%C3%A9thode_de_Broyden-Fletcher-Goldfarb-Shanno#R%C3%A9f%C3%A9rences). Acessado em: 10/10/2022.
- [Menezes et al., 2012] Menezes, I. F. M., Luiz, E. V., and Pereira, A. (2012). Programação matemática, teoria, algoritmos e aplicações.