Trabalho 02
Otimização com 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
Novembro, 2022

1 Introdução

Otimização com restrição é o processo que minimiza uma função objetivo em relação a algumas variáveis em presença de restrições aos valores dessas variáveis.

$$\begin{cases} \text{Minimizar } f(\vec{x}) \\ \text{Sujeito a: } h_k(\vec{x}) = 0, k = 1...m \\ \qquad\qquad\quad c_l(\vec{x}) \leq 0, l = 1...p \\ \qquad\qquad\quad x_i^L \leq x_i \leq x_i^U, i = 1...n \end{cases}$$

As equações h_k e c_l representam as restrições de igualdade e desigualdade, respectivamente. Já os valores x_i^L e x_i^U representam os limites laterais da variável x_i.

Para solução dos problemas desse trabalho, que consiste na minimização de algumas funções sob certas restrições, iremos utilizar os métodos indiretos, são eles o método de penalidade e o de barreira.

2 Objetivos

Os principais objetivos deste trabalho são:

- Implementar numericamente os algoritmos indiretos de otimização com restrições: penalidade e barreira.
- Avaliar a influência dos parâmetros dos algoritmos nas métricas de convergência.
- Aplicar os algoritmos implementados na solução do problema de OCR em dois casos: uma função polinomial de ordem 4 e uma função que representa um problema de engenharia (minimização do peso de uma treliça de duas barras e cujas restrições estão associadas a valores máximos de tensão nas barras das treliças que não podem ser maiores do que as tensões crítica de Euler e de escoamento do material)

As funções a serem minimizadas neste trabalho e as respectivas restrições são:

- **Problema 01:**

$$\begin{cases} \text{Minimizar: } f(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 2x_2)^2 \\ \text{Sujeito a: } x_1^2 - x_2 \leq 0 \end{cases}$$

$$\text{Ponto inicial: } \begin{cases} \text{Para o método da penalidade: } & x^0 = \{3, 2\} \\ \text{Para o método da barreira: } & x^0 = \{0, 1\} \end{cases}$$

- **Problema 02:**

$$\begin{cases} \text{Minimizar: } f(d, H) = 2\rho\pi td\sqrt{H^2 + B^2} \\ \text{Sujeito a: } \dfrac{P\sqrt{H^2 + B^2}}{\pi tdH} \leq \sigma_y \text{ e } \dfrac{P\sqrt{H^2 + B^2}}{\pi tdH} \leq \dfrac{\pi^2 E(d^2 + t^2)}{8(H^2 + B^2)} \end{cases}$$

$$\text{Onde: } \begin{cases} \text{d: diâmetro médio da seção transversal} \\ \text{t: espessura da seção transversal} \\ \text{E: módulo de elasticidade do material} \\ \rho\text{: peso específico do material} \end{cases}$$

$$\text{Ponto inicial: } \begin{cases} \text{Para o método da penalidade: } & x^0 = \{1, 15\} \\ \text{Para o método da barreira: } & x^0 = \{4, 25\} \end{cases}$$

3 Métodos Indiretos em OCR

Segundo [Menezes et al., 2012], as primeiras tentativas de se resolver o problema de otimização com restrições (OCR) foram feitas utilizando-se os métodos indiretos, nomeadamente, os métodos de penalidade e os de barreira. Esses métodos resolvem problemas de OCR por meio de uma sequência de soluções de problemas de OSR. Para que isso seja possível, as restriçõesoes dos problemas de OCR são incorporadas à função objetivo

criando-se as chamadas funções de penalidade (e de barreira) que são usadas nos problemas de OSR. A idéia da função de penalidade (e de barreira) é criar um alto custo pela violação das restrições o que força a solução a atender as restrições. Os métodos indiretos apresentam, em geral, dificuldades computacionais e por isso vêm sendo substituídos pelos métodos diretos. Eles têm, no entanto, o atrativo de serem métodos simples de se resolver problemas de OCR e apresentam uma importância histórica no desenvolvimento de métodos de programação matemática.

A pseudo-função objetivo é dada por ([Menezes, 2022]):

$$\phi(\vec{x}, r) = f(\vec{x}) + r \times p(\vec{x}) \tag{1}$$

Onde:
- r: escalar que define a magnitude da penalização
- $p(\vec{x})$: função penalidade
- $f(\vec{x})$: função objetivo original
- $\phi(\vec{x}, r)$: pseudo-função objetvo

Para evitar problemas numéricos de mal condicionamento, a penalidade r é introduzida de forma gradual, iniciando-se moderada e tendo seu valor incrementado a medida em que o processo de otimização se desenvolve. Dessa forma a solução de um problema de OCR se torna uma sequência de problemas de OSR ([Menezes, 2022]).

3.1 Método da Penalidade

O método de penalidade para restrições de desigualdade é também chamado de método exterior porque ele tem a característica de se aproximar da solução pela região não viável, ou seja, violando as restrições. Essa característica não é vantajosa porque, se o processo iterativo for interrompido por qualquer razão, como mal condicionamento numérico, a solução obtida não é uma solução viável do problema ([Menezes et al., 2012]).

$$\phi(\vec{x}, r_p) = f(\vec{x}) + \frac{1}{2} r_p \sum_{k=1}^{m} h_k^2(\vec{x}) + \frac{1}{2} r_p \sum_{l=1}^{p} \{max[0, c_l(\vec{x})]^2 \tag{2}$$

Podemos observar pela eq. 2 que caso a restrição seja atendida (ponto na região viável), $c_l \leq 0$, logo, a função ϕ não possui o termo da função de penalidade e enconrtrar o mínimo de ϕ equivale a encontrar o mínimo de f. Assim, o ponto em questão irá para a região não viável, por onde converge o algoritmo da penalidade.

O passo a passo do método da penalidade é descrito na figura 1

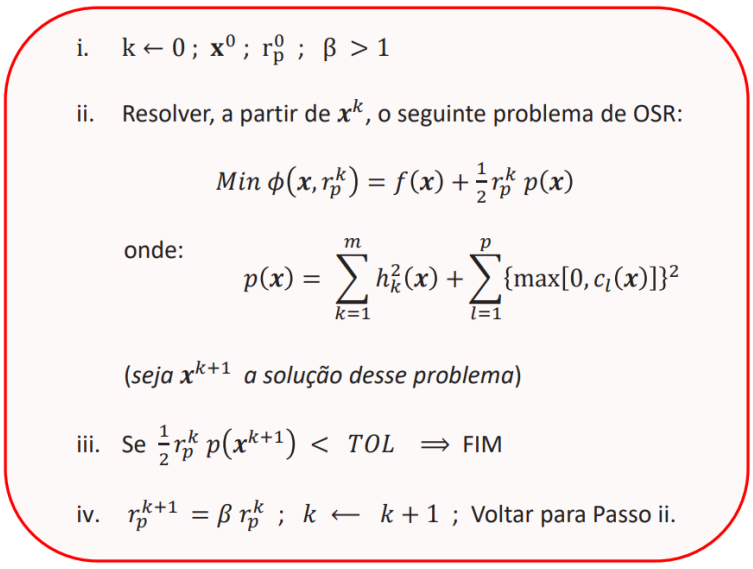


Figura 1: Principais passos do método da penalidade ([Menezes, 2022])

3.2 Método da Barreira

No método da barreira (ou método interior), a convergência se dá do interior da região das soluções viáveis para o contorno dela. Essa característica torna a solução em cada iteração do processo uma solução viável, o que é interessante. O método usa a denominação barreira porque a função de barreira se torna infinita no contorno da região viável ([Menezes et al., 2012]).

$$\phi(\vec{x}, r_p) = f(\vec{x}) + r_b \sum_{k=1}^{m} h_k^2(\vec{x}) + r_b \sum_{l=1}^{p} -\frac{1}{c_l(\vec{x})} \tag{3}$$

Como, no caso da barreira, a solução converge pela região viável, $c_l \leq 0$ e, portanto, existe um sinal negativo na eq. 3 para tornar a pseudo-função objetivo ϕ postiva e, consequentemente, permitir sua minimização, caso contrário esta tenderia para $-\infty$, já que c_l tende a se aproximar de zero à medida que as iterações avançam.

O passo a passo do método da barreira é descrito na figura 2

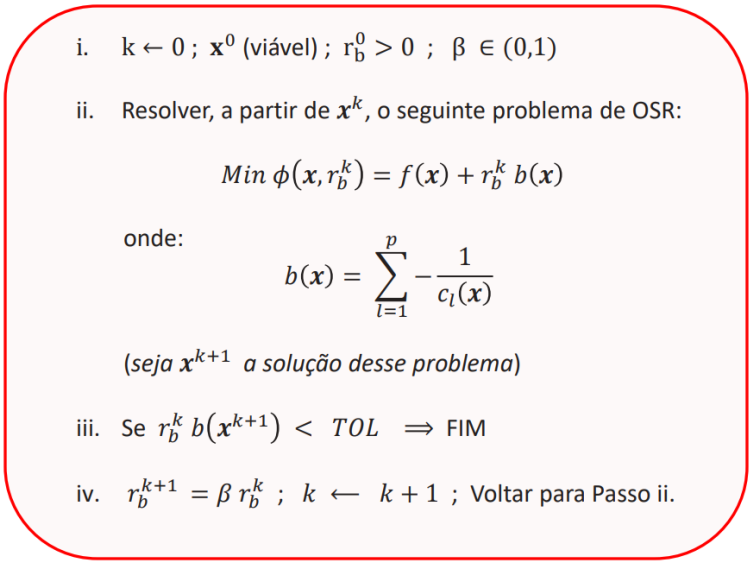


Figura 2: Principais passos do método da barreira ([Menezes, 2022])

4 Metodologia

O fluxo de trabalho para solução do problema de OCR está descrito na figura 3 a seguir é válido tanto para o método de penalidade quanto para o método de barreira.

O primeiro passo é a inicialização das funções f, suas restrições c_l, que no caso dos problemas propostos são apenas de desigualdade, além da escolha do ponto inicial x^0. Em seguida procede-se a escolha dos parâmetros dos algoritmos de OSR e OCR: Tolerâncias para verificação da convergência, $d\alpha$ (tamanho do passo da busca linear), β e r_p (ou r_b).

Em seguida cria-se um looping de $m = 1...6$ para varrer todos os métodos de OSR: Univariante, Powell, Steepest-Descent, Fletcher-Reeves, Newton-Raphson e BFGS.

Para cada um desses métodos cria-se o termo de penalização $p(\vec{x})$ ou $b(\vec{x})$ e a pseudo-função objetivo $\phi(\vec{x}, r)$, além das funções $\vec{\nabla}\phi(\vec{x}, r)$ e $\vec{H}(\phi(\vec{x}, r))$. Esses manipuladores de função são passados para o script OSR juntamente com o ponto x^0 para solução do problema de otimização sem restrição da função ϕ a partir de x^0.

Após a solução de x^k pelo método de OSR, plota-se as curvas de nível de ϕ e todos os pontos de solução x^0, x^1, ... x^{k-1}, x^k até o passo k atual, para evidenciar graficamente a trajetória da solução a cada passo.

Por fim, avalia-se a convergência do problema de OCR avaliando se o termo $\frac{1}{2} r_p p$ (ou $r_b b$) é suficientemente pequeno, e caso não haja convergência, proceder-se-á novo cálculo de x^{k+1} a partir de x^k, para isso atualiza-se

o valor de r_p (ou r_b) e procede-se novo cálculo de OSR a partir do novo x^k. O passo a passo é repetido par todos os métodos de busca direcional de OSR.

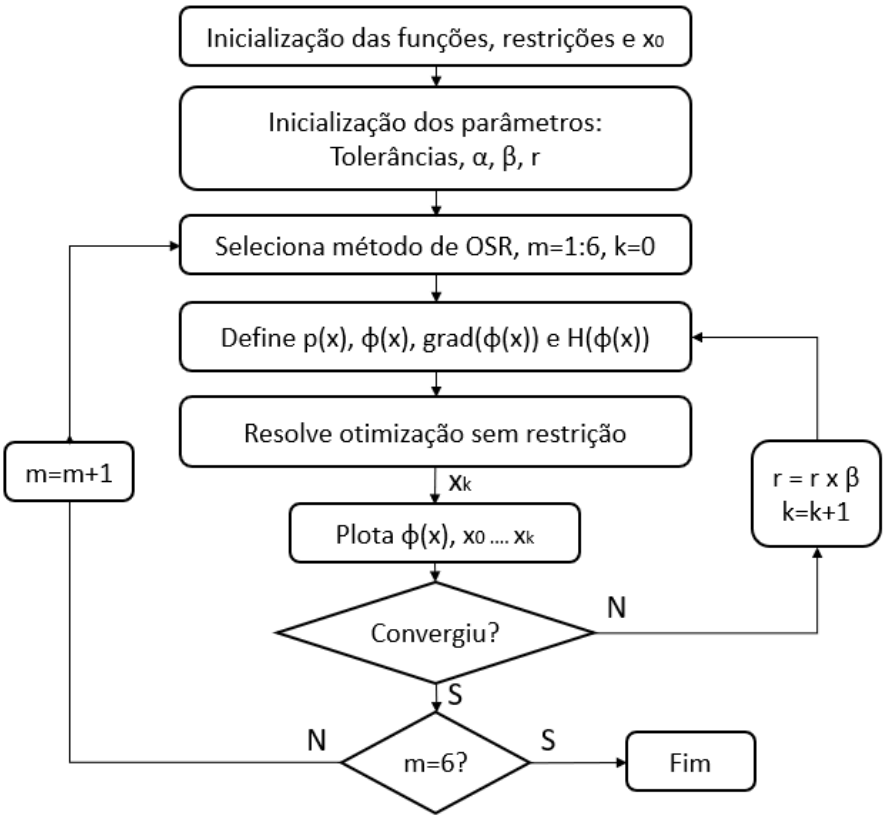


Figura 3: Passo a passo para solução do problema de OCR

Os principais trechos dos scripts *Matlab* utilizados serão listados na seção Anexos.

5 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.

O processo de busca linear implementado para solução do problema de OSR possui ineficiências e imperfeições, principalmente no caso das funções-objetivo serem mais complexas quando associadas aos termos de penalidade, como é o caso das pseudo-funções objetivo dos problemas de OCR, já que incorporam as funções de penalidade. Este fato levou a dificuldades na convergência dos algortimos, em geral.

Foram testados valores de tolerâncias menores, na faixa de 10^{-6}, e $d\alpha$ (passo da busca linear) maiores. Nesses casos, os algortimos até convergiam, porém em temos muito longos, o que prejudicou a possibilidade de testes e execução do trabalho. Dessa forma, o programa implementado para a solução dos problemas de OCR utilizou os valores da tabela 1.

Outro parâmetro relevante foi número máximo de iterações do algoritmo de OSR, para evitar que o algoritmo ficasse indefinidamente na busca pelo mínimo da função ϕ, o que aconteceu para alguns valores de r, principalmente no problema 2 (barreira)

Parâmetros:
- $d\alpha$: tamanho do passo do algoritmo do passo constante
- TOL(BL): tolerância associada ao algoritmo da Busca Linear no problema de OSR
- TOL(OCR): tolerância utilizada para verificar a convergência dos algortimos de OCR
- TOL(SA): tolerância associada ao algoritmo da Seção Áurea no problema de OSR
- β: multiplicados do termo de penalização r da pseudo-função objetivo do algoritmo de OCR
- iter_max: máximo número de iterações do algoritmo de OSR

Tabela 1: Melhores parâmetros para convergência dos algoritmos

Problema	Método	Algoritmo	$d\alpha$	TOL(BL)	TOL(OCR)	TOL(SA)	β
01	Penalidade	Univariante	0.002	10^{-4}	10^{-4}	10^{-6}	5
		Powell	0.002	10^{-4}	10^{-4}	10^{-5}	10
		Steepest Descent	0.002	10^{-4}	10^{-4}	10^{-9}	5
		Flecher-Reeves	0.001	10^{-4}	10^{-4}	10^{-7}	5
		Newton-Raphson	0.05	10^{-4}	10^{-4}	10^{-7}	20
		BFGS	0.04	10^{-5}	10^{-5}	10^{-8}	10
01	Barreira	Univariante	0.0002	10^{-4}	10^{-4}	10^{-8}	0.05
		Powell	0.0002	10^{-4}	10^{-4}	10^{-5}	0.2
		Steepest Descent	0.0002	10^{-4}	10^{-4}	10^{-7}	0.05
		Flecher-Reeves	0.0002	10^{-4}	10^{-4}	10^{-7}	0.05
		Newton-Raphson	0.002	10^{-6}	10^{-6}	10^{-7}	0.05
		BFGS	0.0002	10^{-4}	10^{-4}	10^{-5}	0.05
02	Penalidade	Univariante	0.005	10^{-6}	10^{-6}	10^{-7}	10
		Powell	0.005	10^{-6}	10^{-6}	10^{-5}	50
		Steepest Descent	0.0001	10^{-3}	10^{-4}	10^{-7}	10
		Flecher-Reeves	0.0001	10^{-3}	10^{-4}	10^{-7}	10
		Newton-Raphson	0.0002	10^{-5}	10^{-6}	10^{-7}	10
		BFGS	0.0002	10^{-3}	10^{-4}	10^{-7}	10
02	Barreira	Univariante	0.0001	10^{-4}	10^{-4}	10^{-9}	0.12
		Powell	0.0001	10^{-4}	10^{-4}	10^{-10}	0.09
		Steepest Descent	0.00002	10^{-4}	10^{-4}	10^{-9}	0.005
		Flecher-Reeves	0.00002	10^{-4}	10^{-4}	10^{-7}	0.008
		Newton-Raphson	0.001	10^{-6}	10^{-6}	10^{-6}	0.01
		BFGS	0.0002	10^{-5}	10^{-4}	10^{-8}	0.0006

A tabela 2 abaixo ilustra os resultados encontrados para a implementação dos métodos de penalidade e barreira para as funções e restrições dos problemas 01 e 02. Além dos pontos de mínimo encontrados, a tabela apresenta também o número de passos para a convergência e o tempo de execução. Estes dois últimos representam quantas vezes foi chamado o script de solução do problema de OSR e quanto tempo computacional foi consumido no total de iterações, respectivamente.

Tabela 2: Resultados

Problema	Método, x^0	Algoritmo	x^{min}	passos	Δt(ms)
01	Penalidade, $x^0 = \{3, 2\}$	Univariante	$\{0.9446, 0.8921\}$	8	55.3
		Powell	$\{0.9456, 0.8941\}$	6	198.4
		Steepest Descent	$\{0.9456, 0.8941\}$	8	20.8
		Flecher-Reeves	$\{0.9456, 0.8941\}$	8	5.9
		Newton-Raphson	$\{0.9456, 0.8941\}$	5	3.9
		BFGS	$\{0.9456, 0.8941\}$	7	2.4
01	Barreira, $x^0 = \{0, 1\}$	Univariante	$\{0.9467, 0.8969\}$	6	78.0
		Powell	$\{0.9453, 0.8944\}$	11	8615.5
		Steepest Descent	$\{0.9456, 0.8948\}$	6	797.9
		Flecher-Reeves	$\{0.9454, 0.8944\}$	6	297.1
		Newton-Raphson	$\{0.9456, 0.8941\}$	10	124.0
		BFGS	$\{0.9454, 0.8944\}$	6	239.1
02	Penalidade, $x^0 = \{1, 15\}$	Univariante	$\{1.8784, 20.2363\}$	15	144.3
		Powell	$\{1.8783, 20.2365\}$	8	247.3
		Steepest Descent	$\{1.8784, 20.2357\}$	14	5302.9
		Flecher-Reeves	$\{1.8783, 20.2350\}$	12	1920.4
		Newton-Raphson	$\{1.8783, 20.2365\}$	15	528.7
		BFGS	$\{1.8783, 20.2363\}$	13	6433.7
02	Barreira, $x^0 = \{4, 25\}$	Univariante	$\{1.8785, 20.2389\}$	15	13461.8
		Powell	$\{1.8785, 20.2396\}$	13	11053.0
		Steepest Descent	$\{1.8793, 20.8871\}$	5	9248.0
		Flecher-Reeves	$\{1.8855, 20.3466\}$	5	3708.0
		Newton-Raphson	$\{1.8784, 20.2367\}$	8	225.5
		BFGS	$\{1.8971, 20.5055\}$	3	1675.9

Conforme citado anteriormente, a utlização de métodos indiretos em OCR consiste na incorporação das restrições na função objetivo a ser minimizada, sendo os termos associados às restrições penalizados gradualmente.

Nas figuras que serão mostradas nas seções seguintes, para cada passo dos algoritmos de penalidade ou barreira, ilustra-se as curvas de nível da pseudo-função objetivo $\phi(x_1, x_2)$, onde se pode evidenciar a forma que esta vai tomando a cada passo do método, a partir da incorporação e penalização dos termos de penalidade.

Sobrepostos às curvas de nível de $\phi(x_1, x_2)$, para cada passo $k = 1....n$ ilustra-se o conjunto de pontos $\{x^0, x^1, ..., x^k\}$ (soluções do problema de OSR) até o passo atual.

As figuras a seguir ilustram as curvas da solução de OCR para a função a as restrições do problema 1. A restrição de desigualdade está representada pela linha vermelha em traço forte.

Na figura 4 podemos ver a região viável (acima da curva vermelha) e a convergência do algoritmo pelo algortimo de Powell de OSR pelo método da penalidade, assim como a deformação de $\phi(x_1, x_2)$ a partir do ponto $x^0 = \{3, 2\}$. Nota-se também a convergência do algoritmo pela região não viável (abaixo da curva vermelha).

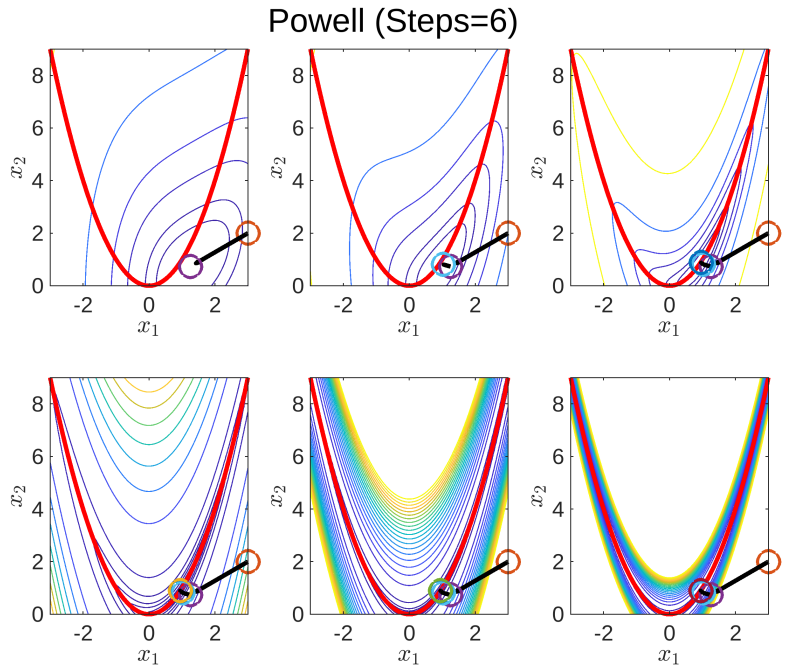


Figura 4: OCR do problema 01 pelo método da penalidade, a partir do ponto $x^0 = \{3, 2\}$ - Algoritmo de Powell

Para avaliar a robustez dos algoritmos, selecionamos um novo ponto $x^0 = \{0, 4\}$, desta vez dentro da região viável, uma vez que, diferente do método de barreira, não é uma exigência do algortimo que o ponto inicial seja um ponto viável. Nota-se que logo no primeiro passo, a solução de OSR de ϕ converge para mínimo da função f, já que, como x^0 é viável, esse não é computado em ϕ e esta fica igual a f. A partir daí, a sequencia do algoritmo segue pela região não-viável. Na figura 5 ilustra-se tal caso pelo método de Fletcher-Reeves.

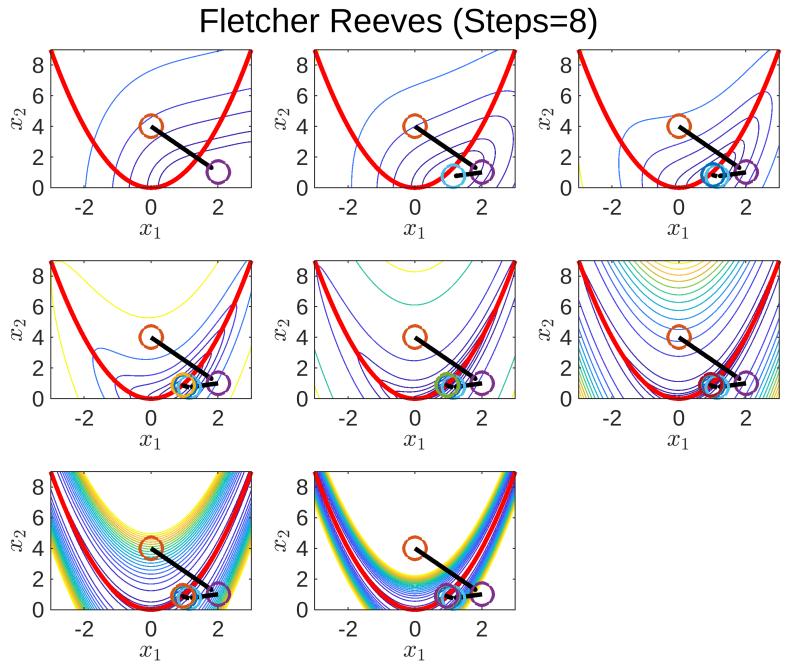


Figura 5: OCR do problema 01 pelo método da penalidade, a partir do ponto $x^0 = \{0, 4\}$ - Algoritmo de Flecher-Reeves

Para o mesmo problema 01, a figura 6 a seguir ilustra a convergênia, pelo método de OSR univariante, a partir do ponto $x^0 = \{0, 1\}$ e pelo método de barreira.

Desta vez é possível notar que a região associada à restrição (curva vermelha) vai se tornando uma barreira nas curvas de nível de ϕ. Aqui, é importante que a solução, em qualquer passo k, x^k permaneça viável, ou seja, $c_l(x^k) <= 0$, caso contrário, o ponto x^k "romperá"a barreira e a função irá convergir para um ponto na região não viável.

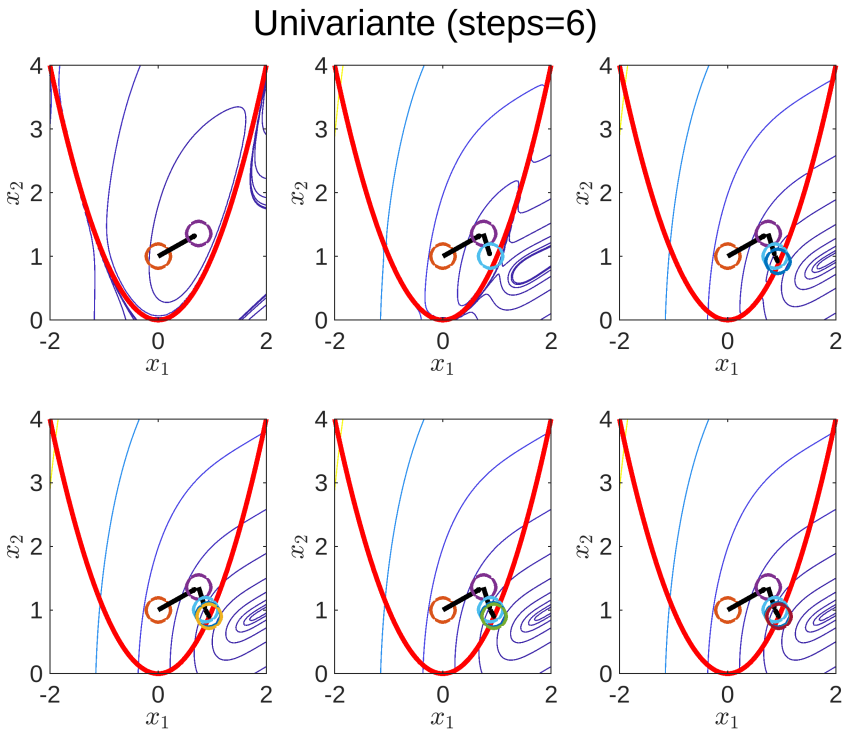
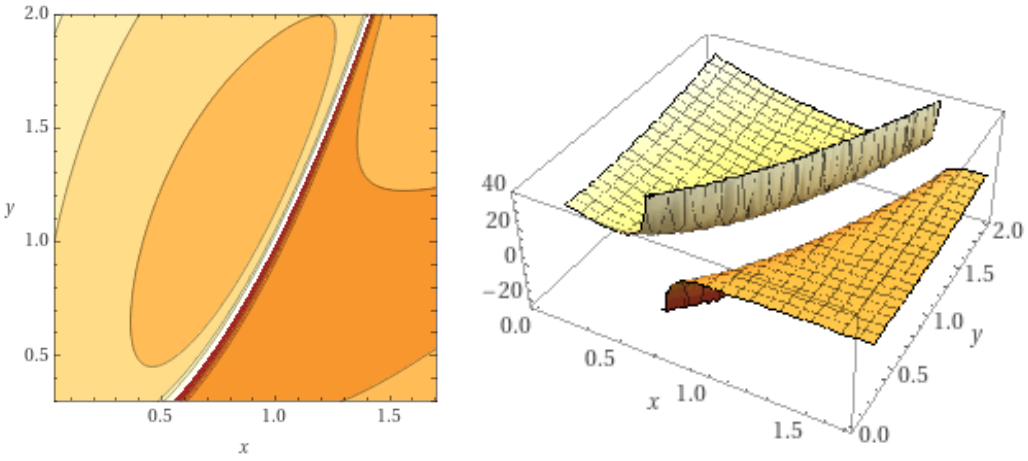


Figura 6: OCR do problema 01 pelo método da barreira, a partir do ponto $x^0 = \{0, 1\}$ - Algoritmo Univariante

Para melhor visualização da pseudo-função objetivo acima plotamos os gráficos de curvas de nível e 3D da função da eq. 4 nas figuras 7a e 7b, respectivamente, onde fica claro a região associada à restrição como sendo uma "descontinuidade"de ϕ e porque o método é chamado de método de barreira.

$$\phi(x, y, r_b = 1) = (x - 2)^4 + (x - 2y)^2 - \frac{rb}{x^2 - y} \tag{4}$$



(a) contono de ϕ **(b)** 3D plot de ϕ

Figura 7: Pseudo função objetivo do problema 01 (Barreira) com $rb = 1$ (equação 4), https://www.wolframalpha.com

Para avaliar a robustez do método, selecionamos um novo ponto $x^0 = \{2.5, 10\}$, ainda viável. Na figura 8 ilustra-se tal caso pelo método de Steepest-Descent.

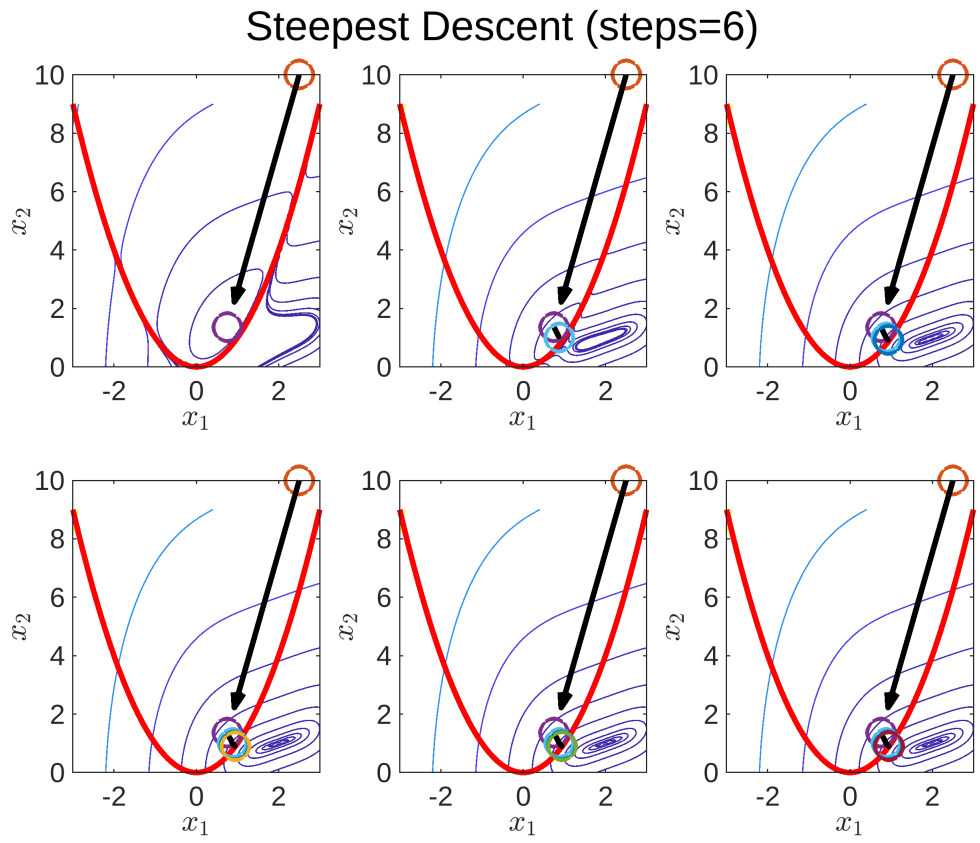


Figura 8: OCR do problema 01 pelo método da barreira, a partir do ponto $x^0 = \{2.5, 10\}$ - Algoritmo Steepest Descent

As figuras a seguir ilustram as curvas da solução de OCR para a função a as restrições do problema 2. As restrições de desigualdade estão representadas pelas linhas vermelha e preta em traço forte.

Nas figuras 9 e 10 a seguir ilustra-se as convergências do algoritmo de OCR pelo método de penalidade para dois pontos de partida distintos: $x^0 = \{1, 15\}$ e $x^0 = \{3, 10\}$, respecivamente. O primeiro utilizando, para OSR o método de direções de busca de Fletcher-Reeves e o segundo, Univariante.

Nota-se nesses dois gráficos a maior complexidade da função e o consequente maior número de passos necessários para convergência.

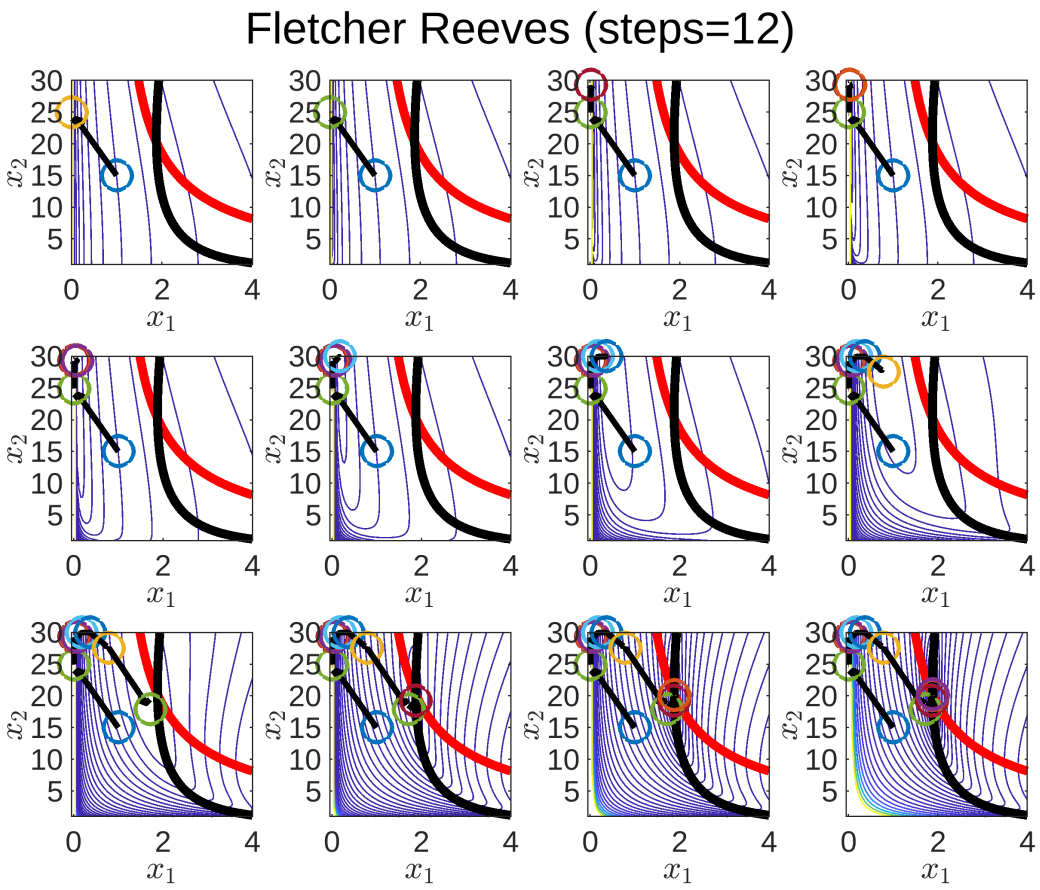


Figura 9: OCR do problema 02 pelo método da penalidade, a partir do ponto $x^0 = \{1, 15\}$ - Algoritmo de Flecher-Reeves

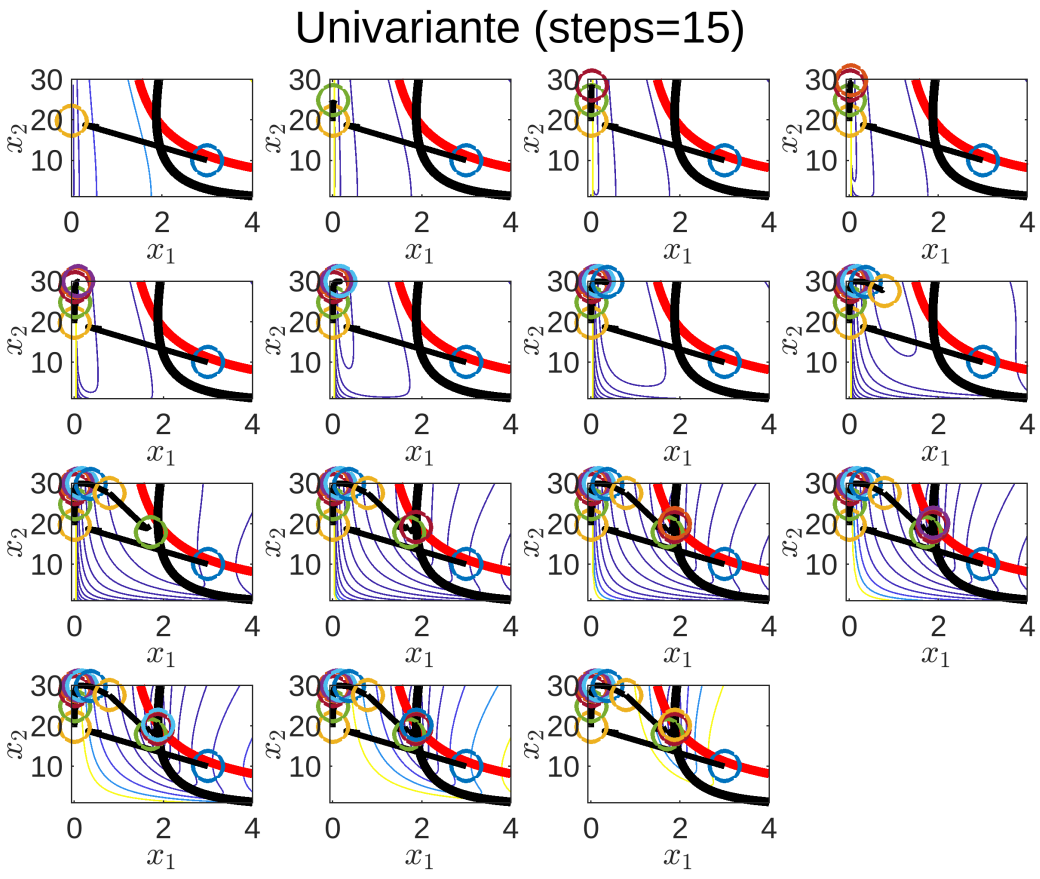


Figura 10: OCR do problema 02 pelo método da penalidade, a partir do ponto $x^0 = \{3, 10\}$ - Algoritmo Univariante

Nas figuras 11 e 12 a seguir ilustra-se as convergências do algoritmo de OCR pelo método de barreira para dois pontos de partida distintos: $x^0 = \{4, 25\}$ e $x^0 = \{10, 5\}$, respecivamente. O primeiro utilizando, para OSR o método de direções de busca de Newton-Raphson e o segundo, Fletcher-Reeves.

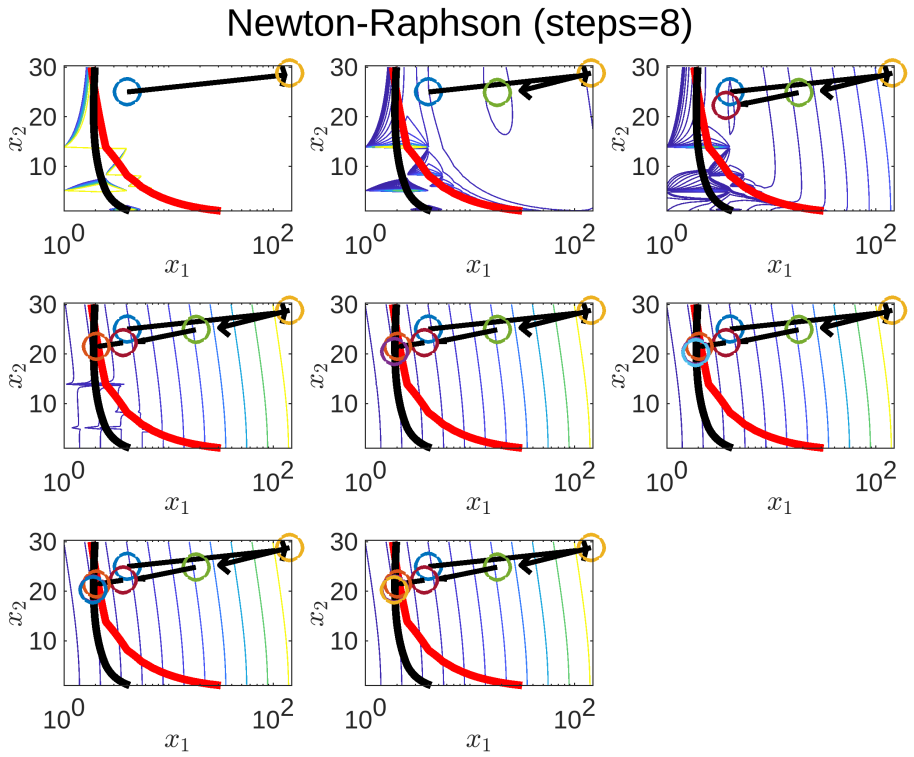


Figura 11: OCR do problema 02 pelo método da barreira, a partir do ponto $x^0 = \{4, 25\}$ - Algoritmo de Newton-Raphson

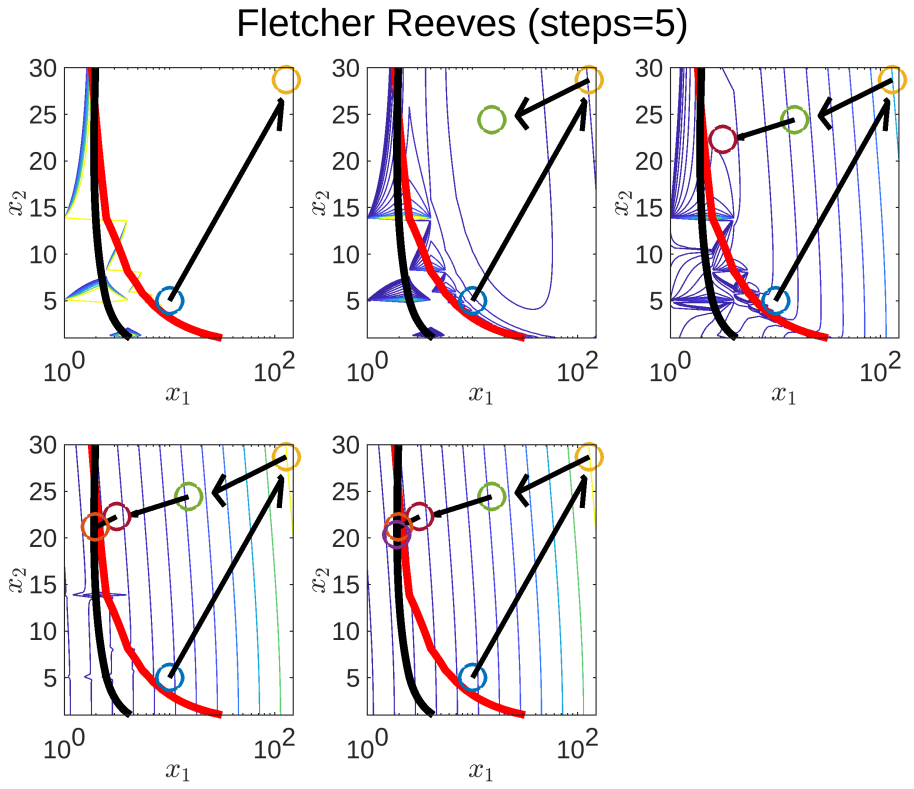


Figura 12: OCR do problema 02 pelo método da barreira, a partir do ponto $x^0 = \{10, 5\}$ - Algoritmo de Fletcher-Reeves

6 Conclusões

A realização deste trabalho permitiu implementar os métodos indiretos de solução de problemas de otimização com restrição (OCR) estudados na disciplina de otimização. Para todos os casos de aplicação, foram rodados os algortimos para outros pontos que não os pontos propostos no problema em si e cujo resultado mostrou que os algortimos de busca e minização são robustos.

Devido às ineficiências do método de busca linear e a maior complexidade da pseudo-função objetivo a partir da incorporação dos termos de penalidade, foi necessário ajustar os parâmetros dos algoritmos para se obter a convergência.

No problema 02, dadas as constantes físicas associadas ao problema, que impuseram às funções alguns termos com números grandes, foi fundamental a normalização das restrições para sua 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 parmêtros, o que reside nas incertezas numéricas associadas aos métodos de busca linear, principalmente.

Apesar da soluçao do problema de OSR para algumas configurações da função ϕ não ter convergido em alguns passos, a melhor solução aproximada até o número máximo de iterações foi suficiente para a convergência geral do problema de OCR.

7 Anexos

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

Anexo 1: Inicialização e dados do problema 01

```
% parametros dos algoritmos
iter_max = 500;
methods = ["Univariante","Powell","Steepest Descent","Fletcher Reeves","Newton-
    Raphson","BFGS"];

% dados do item 01
f1 = @(x) (x(1)-2)^4 + (x(1)-2*x(2))^2;
gf1 = @(x) [2*(2*(x(1)-2)^3 + x(1) - 2*x(2)); 8*x(2)-4*x(1)];
c1 = @(x) x(1)^2 - x(2);
```

Anexo 2: definição das funções e restrições do problema 02

```
% dados do item 02
% ro = 0.3;
% B = 30;
% P = 33e3;
% t = 0.1;
% E = 3e7;
% sy = 1e5;
% f2 = @(x) 2*ro*pi()*x(1)*t*sqrt(x(2)^2+B^2);
f2 = @(x) 0.1885*x(1)*sqrt(x(2)^2+900);
c2a = @(x) 1.0504*sqrt(x(2)^2+900)/(x(1)*x(2)) - 1;
c2b = @(x) 1.0504*sqrt(x(2)^2+900)/(x(1)*x(2)) - 3.7011e2*(x(1)^2+0.01)/(x(2)
    ^2+900);
```

Anexo 3: trecho de código do problema 01 (penalidade)

```
fprintf('**********␣PROBLEMA␣01␣-␣PENALIDADE␣**********\n');
x0 = [3;2];
%x0 = [0;4]; % ponto alternativo
alphas =    [0.002, 0.002, 0.002, 0.001, 0.05, 0.04];
bs =        [5, 10, 5, 5, 20, 10];
TOLS =      [1e-4, 1e-4, 1e-4, 1e-4, 1e-4, 1e-5];
TOLS2 =     [1e-6, 1e-5, 1e-9, 1e-7, 1e-7, 1e-8];
plot_model = 0; %escolher qual metodo sera plotado

if plot_model ~= 0
    figure
    i=3;
    j=3;
end
for m = [1:6]
    k=0;
    x=x0;
    x_values = x;
    rp=0.1;

    tstart = tic;
    fprintf('---%s---\n', methods(m));

    while k < iter_max
            if c1(x) < 0
            a = 0;
            else
            a = 1;
            end
            rp = bs(m)*rp;

            p1 = @(x) a * c1(x)^2;
            phi1 = @(x) f1(x) + 1/2 * rp * p1(x);
            gphi1 = @(x) [2*(a*rp*x(1)*(x(1)^2-x(2))+2*(x(1)-2)^3+x(1)-2*x(2)); -a
                *rp*x(1)^2+x(2)*(a*rp+8)-4*x(1)];
            hess1 = @(x) ...
            [[2*a*rp*(x(1)^2-x(2)) + 4*a*rp*x(1)^2 + 12*(x(1) - 2)^2 + 2, -2*a*rp*
                x(1) - 4];
            [-2*a*rp*x(1) - 4, a*rp + 8]];

            [x_] = osr_noprint(phi1, gphi1, hess1, x, m, iter_max, alphas(m), TOLS
                (m), TOLS2(m));
            k=k+1;

            if 1/2*p1(x)*rp < TOLS(m)*10 && p1(x) ~= 0
            conv=1;
            fprintf('x=[%.4f,%.4f],␣f(x)=%.4f␣(%d␣iter/%.1fms)\n', x(1), x(2), f1(
                x), k-1, toc(tstart)*1000);
            break;
            end
            x = x_(:,end);
            x_values = [x_values,x];

            intervals = [0 1 2 5 10 25 100:500:10000];
            if m == plot_model
            phi1_plot = @(x1,x2) (x1-2).^4 + (x1-2*x2).^2+1/2*a*rp*(x1.^2-x2).^2;
            c1_plot = @(x1,x2) x1.^2-x2;
            c2_plot = [];
            intervals = intervals*10;
            tit = strcat(methods(m), '␣(Steps=',num2str(k),')');
            plot_phi_c(-3, 3, 0, 9, phi1_plot, intervals, c1_plot, [], x_values,
                tit, k, i, j);
      end
    end
      if ~conv
        fprintf('\n␣nao␣deu');
    end
end
```

Anexo 4: trecho de código do problema 01 (barreira)

```
fprintf('**********␣PROBLEMA␣01␣-␣BARREIRA␣**********\n');
x0 = [0;1];
%x0 = [2.5;10]; % ponto alternativo
alphas =    [0.0002, 0.0002, 0.0002, 0.0002, 0.002, 0.0002];
bs =        [0.05,      0.2,    0.05, 0.05, 0.05, 0.05];
TOLS =      [2e-4,      6e-4,   2e-4, 2e-4, 1e-6, 2e-4];
TOLS2 =     [1e-8,      1e-5,   1e-7, 1e-7, 1e-7, 1e-5];
plot_model = 0;

if plot_model ~= 0
figure
i=2;
j=3;
end
for m = [1:6]
k=0;
x=x0;
x_values = x;
rb=100;

tstart = tic;
fprintf('---%s---\n', methods(m));

while k < iter_max
      rb = bs(m)*rb;
      b1 = @(x) - c1(x)^(-1);
      phib1 = @(x) f1(x) + rb * b1(x); % (x-2)^4 + (x-2y)^2 - a/(x^2-y)

      gphib1 = @(x) ... % (2 ((a x)/(x^2 - y)^2 + 2 (x - 2)^3 + x - 2 y), -a/(x^2
          - y)^2 - 4 x + 8 y)
            [2*((rb*x(1))/(x(1)^2-x(2))^2 + 2*(x(1)-2)^3 + x(1)-2*x(2)); ...
            -rb/(x(1)^2-x(2))^2 - 4*x(1)+8*x(2)];

      hessb1 = @(x) ...
            [[2 + 12*(-2 + x(1))^2 - (8*rb*x(1)^2)/(x(1)^2 - x(2))^3 + (2*rb)/(x
                (1)^2 - x(2))^2, -4 + (4*rb*x(1))/(x(1)^2 - x(2))^3]; ...
            [-4 + (4*rb*x(1))/(x(1)^2 - x(2))^3, 8 - (2*rb)/(x(1)^2 - x(2))^3]];

      [x_] = osr_noprint(phib1, gphib1, hessb1, x, m, iter_max, alphas(m), TOLS(m)
          , TOLS2(m));

      b1(x)*rb;
      if b1(x)*rb < TOLS(m)
            conv=1;
            c1(x);
            fprintf('x=[%.4f,%.4f],␣f(x)=%.4f␣(%d␣iter/%.1fms)\n', x(1), x(2), f1(
                x), k, toc(tstart)*1000);
            break;
      end
      x = x_(:,end);
      x_values = [x_values,x];

      k=k+1;

      intervals = 10.^[-2:.5:5];
      if m == plot_model
            phib1_plot = @(x1,x2) (x1-2).^4 + (x1-2*x2).^2 - rb./(x1.^2-x2);
            c_plot = @(x1,x2) x1.^2-x2;
            c2_plot = [];
            intervals = intervals*10;
            plot_phi_c(-3,3,0,9, phib1_plot, intervals, c1_plot, [], x_values, tit
                , k, i , j);
      end
end

if ~conv
      fprintf('\n␣nao␣deu');
end
```

Anexo 5: trecho de código do problema 02 (penalidade) (1/2)

```
x0 = [1;15];
%x0 = [3;3]; %ponto alternativo
alphas =    [0.005, 0.005, 0.0001,  0.0001, 0.0002, 0.0002];
bs =        [10,    50,      10,    10,     10,       10];
TOLS =      [1e-6,  1e-6,   2e-3,  1e-3,   1e-5,   5e-3];
TOLS_OCR =  [1e-6,  1e-6,   1e-4,  5e-4,   1e-6,   1e-4];
TOLS2 =     [1e-7,  1e-5,   1e-7,  1e-7,   1e-7,   1e-7];
plot_model = 0;

if plot_model ~= 0
figure
i=4;
j=4;
end
for m = [1:6]
k=0;
x=x0;
x_values = x;
rp=1e-8;

tstart = tic;
fprintf('---%s---\n', methods(m));

while k < 20
      if c2a(x) < 0
            a(1) = 0;
      else
            a(1) = 1;
      end
      if c2b(x) < 0
            a(2) = 0;
      else
            a(2) = 1;
      end

      rp = bs(m)*rp;
      p2 = @(x) a(1) * c2a(x)^2 + a(2) * c2b(x)^2;
      phi2 = @(x) f2(x) + 1/2 * rp * p2(x);

      g2f = @(x) ...
      [0.1885*sqrt(x(2)^2+900);
      (0.1885*x(1)*x(2))/sqrt(x(2)^2+900)];
      g2c1 = @(x) ...
      [(a(1)*rp*(1.0504*x(1)*sqrt(x(2)^2+900)*x(2)-1.10334*x(2)^2-993.006))/(x(1)
          ^3*x(2)^2);
      (a(1)*rp*((945.36*x(1)*x(2))/sqrt(x(2)^2+900)-993.006))/(x(1)^2*x(2)^3)];
      g2c2 = @(x) ...
      [a(2)*rp*(-(1.0504*sqrt(x(2)^2 + 900))/(x(1)^2*x(2)) - (740.22*x(1))/(x(2)^2
           + 900))*((1.0504*sqrt(x(2)^2 + 900))/(x(1)*x(2)) - (370.11*(x(1)^2 +
          0.01))/(x(2)^2 + 900));
      a(2)*rp*((740.22*x(1)^2*x(2))/(x(2)^2 + 900)^2 - 945.36/(x(1)*x(2)^2*sqrt(x
          (2)^2 + 900)) + (7.4022*x(2))/(x(2)^2 + 900)^2)*((1.0504*sqrt(x(2)^2 +
          900))/(x(1)*x(2)) - (370.11*(x(1)^2 + 0.01))/(x(2)^2 + 900))];

      gphi2 = @(x) g2f(x)+g2c1(x)+g2c2(x);

      hess2 = @(x) ...
            [[(1.10334*a(1)*rp*(900 + x(2)^2))/(x(1)^4*x(2)^2) + a(2)*rp
                *((-740.22*x(1))/(900 + x(2)^2) - (1.0504*sqrt(900 + x(2)^2))/(x
                (1)^2*x(2)))^2 + (2.1008*a(1)*rp*sqrt(900 + x(2)^2)*(-1 + (1.0504*
                sqrt(900 + x(2)^2))/(x(1)*x(2))))/(x(1)^3*x(2)) + a(2)*rp
                *(-740.22/(900 + x(2)^2) + (2.1008*sqrt(900 + x(2)^2))/(x(1)^3*x
                (2)))*((-370.11*(0.01 + x(1)^2))/(900 + x(2)^2) + (1.0504*sqrt(900
                 + x(2)^2))/(x(1)*x(2))), ...
```

Anexo 6: trecho de código do problema 02 (penalidade) (2/2)

```
                    (0.1885*x(2))/sqrt(900 + x(2)^2) - (1.0504*a(1)*rp*sqrt(900 + x(2)^2)
                        *(1.0504/(x(1)*sqrt(900 + x(2)^2)) - (1.0504*sqrt(900 + x(2)^2))/(
                        x(1)*x(2)^2)))/(x(1)^2*x(2)) + a(2)*rp*((740.22* (0.01 + x(1)^2)*x
                        (2))/(900 + x(2)^2)^2 + 1.0504/(x(1)*sqrt(900 + x(2)^2)) -
                        (1.0504*sqrt(900 + x(2)^2))/(x(1)*x(2)^2))*((-740.22*x(1))/(900 +
                        x(2)^2) - (1.0504*sqrt(900 + x(2)^2))/(x(1)^2*x(2))) - (1.0504*a
                        (1)*rp*(-1 + (1.0504*sqrt(900 + x(2)^2))/(x(1)*x(2))))/(x(1)^2*
                        sqrt(900 + x(2)^2)) + (1.0504*a(1)*rp*sqrt(900 + x(2)^2)*(-1 +
                        (1.0504*sqrt(900 + x(2)^2))/(x(1)*x(2))))/(x(1)^2*x(2)^2) + a(2)*
                        rp*((1480.44*x(1)*x(2))/(900 + x(2)^2)^2 - 1.0504/(x(1)^2*sqrt(900
                         + x(2)^2)) + (1.0504*sqrt(900 + x(2)^2))/(x(1)^2*x(2)^2))*
                        ((-370.11*(0.01 + x(1)^2))/(900 + x(2)^2) + (1.0504*sqrt(900 + x
                        (2)^2))/(x(1)*x(2)))]; ...
                    [(0.1885*x(2))/sqrt(900 + x(2)^2) - (1.0504*a(1)*rp*sqrt(900 + x(2)^2)
                        *(1.0504/(x(1)*sqrt(900 + x(2)^2)) - (1.0504*sqrt(900 + x(2)^2))/(
                        x(1)*x(2)^2)))/(x(1)^2*x(2)) + a(2)*rp*((740.22* (0.01 + x(1)^2)*x
                        (2))/(900 + x(2)^2)^2 + 1.0504/(x(1)*sqrt(900 + x(2)^2)) -
                        (1.0504*sqrt(900 + x(2)^2))/(x(1)*x(2)^2))*((-740.22*x(1))/(900 +
                        x(2)^2) - (1.0504*sqrt(900 + x(2)^2))/(x(1)^2*x(2))) - (1.0504*a
                        (1)*rp*(-1 + (1.0504*sqrt(900 + x(2)^2))/(x(1)*x(2))))/(x(1)^2*
                        sqrt(900 + x(2)^2)) + (1.0504*a(1)*rp*sqrt(900 + x(2)^2)*(-1 +
                        (1.0504*sqrt(900 + x(2)^2))/(x(1)*x(2))))/(x(1)^2*x(2)^2) + a(2)*
                        rp*((1480.44*x(1)*x(2))/(900 + x(2)^2)^2 - 1.0504/(x(1)^2*sqrt(900
                         + x(2)^2)) + (1.0504*sqrt(900 + x(2)^2))/(x(1)^2*x(2)^2))*
                        ((-370.11*(0.01 + x(1)^2))/(900 + x(2)^2) + (1.0504*sqrt(900 + x
                        (2)^2))/(x(1)*x(2))), ...
                    (-0.1885*x(1)*x(2)^2)/(900 + x(2)^2)^(3/2) + (0.1885*x(1))/sqrt(900 +
                        x(2)^2) + a(1)*rp*(1.0504/(x(1)*sqrt(900 + x(2)^2)) - (1.0504*sqrt
                        (900 + x(2)^2))/(x(1)*x(2)^2))^2 + a(2)*rp*((740.22*(0.01 + x(1)
                        ^2)*x(2))/(900 + x(2)^2)^2 + 1.0504/(x(1)*sqrt(900 + x(2)^2)) -
                        (1.0504*sqrt(900 + x(2)^2))/(x(1)*x(2)^2))^2 + a(1)*rp*((-1.0504*x
                        (2))/(x(1)*(900 + x(2)^2)^(3/2)) - 1.0504/(x(1)*x(2)*sqrt(900 + x
                        (2)^2)) + (2.1008*sqrt(900 + x(2)^2))/(x(1)*x(2)^3))*(-1 +
                        (1.0504*sqrt(900 + x(2)^2))/(x(1)*x(2))) + a(2)*rp
                        *((-2960.88*(0.01 + x(1)^2)*x(2)^2)/(900 + x(2)^2)^3 +
                        (740.22*(0.01 + x(1)^2))/(900 + x(2)^2)^2 - (1.0504*x(2))/(x(1)
                        *(900 + x(2)^2)^(3/2)) - 1.0504/(x(1)*x(2)*sqrt(900 + x(2)^2)) +
                        (2.1008*sqrt(900 + x(2)^2))/(x(1)*x(2)^3))*((-370.11*(0.01 + x(1)
                        ^2))/(900 + x(2)^2) + (1.0504*sqrt(900 + x(2)^2))/(x(1)*x(2)))]];

            [x_] = osr_noprint(phi2, gphi2, hess2, x, m, 400, alphas(m), TOLS(m), TOLS2(
                m));
            x = x_(:,end);
            x_values = [x_values,x];
            k=k+1;

            if 1/2*p2(x)*rp < TOLS_OCR(m)
                conv=1;
                fprintf('x=[%.4f,%.4f], f(x)=%.4f (%d iter/%.1fms)\n', x(1), x(2), f2(
                    x), k-1, toc(tstart)*1000);
                break;
            end

            intervals=10.^[-3:.1:6];
            if m == plot_model
                phi2_plot = @(x1,x2) ...
                0.1885.*x1.*sqrt(x2.^2+900) + ...
                1/2*rp*a(1)*(1.0504.*sqrt(x2.^2+900)./(x1.*x2) - 1).^2 + ...
                1/2*rp*a(2)*(1.0504.*sqrt(x2.^2+900)./(x1.*x2) - 3.7011e2.*(x1
                    .^2+0.01)./(x2.^2+900)).^2;
                c2a_plot = @(x1, x2) 1.0504.*sqrt(x2.^2+900)./(x1.*x2) - 1;
                c2b_plot = @(x1, x2) 1.0504.*sqrt(x2.^2+900)./(x1.*x2) - 3.7011e2.*(x1
                    .^2+0.01)./(x2.^2+900);
                tit = strcat(methods(m), ' (steps=',num2str(k),')');
                plot_phi_c(0, 4, 1, 30, phi2_plot, intervals, c2a_plot, c2b_plot,
                    x_values, tit, k, i, j);
            end
        end
        if ~conv
            fprintf('\n nao deu');
        end
```

Anexo 7: trecho de código do problema 02 (barreira) (1/2)

```
x0 = [4;25];
% x0 = [10;5];
% verifica se o ponto alternativo) esta na regiao viavel
% c2a(x0)
% c2b(x0)
alphas =    [0.0001,  0.0001,  0.00002, 0.00002, 0.001,   0.0002];
bs =        [  0.12,    0.09,   0.005,   0.008,   0.01,   0.0006];
TOLS =      [  1e-4,    1e-4,    5e-4,   1e-4,    1e-6,    1e-5];
TOLS_OCR =  [  2e-4,    2e-4,    2e-4,   3e-4,    1e-6,    1e-4];
TOLS2 =     [  1e-9,    1e-10,   1e-9,   1e-7,    1e-6,    1e-8];
plot_model = 0;

if plot_model ~= 0
    figure
    i=3;
    j=2;
end
for m = [1:6]
    k=0;
    x=x0;
    x_values = x;
    rb=1e7;

    tstart = tic;
    fprintf('---%s---\n', methods(m));

    while k < iter_max
      rb = bs(m)*rb;
      b2 = @(x) -c2a(x)^-1 -c2b(x)^-1;
      phib2 = @(x) f2(x) + rb * b2(x);

      g2f = @(x) ...
      [0.1885*sqrt(x(2)^2+900);
      (0.1885*x(1)*x(2))/sqrt(x(2)^2+900)];

      g2c1 = @(x) ...
      [-(0.952018*rb*x(2)*sqrt(x(2)^2 + 900))/(0.952018*x(1)*x(2) - sqrt(x(2)^2 +
          900))^2;
      -(856.816*rb*x(1))/(sqrt(x(2)^2 + 900)*(0.952018*x(1)*x(2) - sqrt(x(2)^2 +
          900))^2)];

      g2c2 = @(x) ...
      [-(0.952018*rb*x(2)*(x(2)^2 + 900)*(704.703*x(1)^3*x(2) + (x(2)^2 + 900)
          ^(3/2)))/(352.351*x(1)^3*x(2) + 3.52351*x(1)*x(2) - (x(2)^2 + 900)^(3/2)
          )^2; ...
      (670.89*rb*x(1)*((x(1)^3 + 0.01*x(1))*sqrt(x(2)^2 + 900)*x(2)^3 - 1.27713*x
          (2)^4 - 2298.84*x(2)^2 - 1.03448*10^6))/(sqrt(x(2)^2 + 900)*(352.351*x
          (1)^3*x(2) + 3.52351*x(1)*x(2) - (x(2)^2 + 900)^(3/2))^2)];

      gphib2 = @(x) g2f(x)+g2c1(x)+g2c2(x);

      hessf = @(x) ...
      [[0, ...
            (0.1885*x(2))/sqrt(900 + x(2)^2)]; [(0.1885* x(2))/sqrt(900 + x(2)^2),
                 ...
            (-0.1885* x(1)* x(2)^2)/(900 + x(2)^2)^(3/2) + (0.1885* x(1))/sqrt(900
                 + x(2)^2)]];

      hessc1 = @(x) ...
            [[(-2.20668* rb* (900 + x(2)^2))/(x(1)^4* x(2)^2* (-1 + (1.0504* sqrt
                (900 + x(2)^2))/(x(1)* x(2)))^3) + (2.1008 *rb *sqrt(900 + x(2)^2)
                )/(x(1)^3* x(2)* (-1 + (1.0504* sqrt(900 + x(2)^2))/(x(1)* x(2)))
                ^2), ...
            (2.1008 *rb* sqrt(900 + x(2)^2)* (1.0504/(x(1)* sqrt(900 + x(2)^2)) -
                (1.0504* sqrt(900 + x(2)^2))/(x(1)* x(2)^2)))/(x(1)^2* x(2)* (-1 +
                 (1.0504* sqrt(900 + x(2)^2))/(x(1)* x(2)))^3) - (1.0504 *rb)/(x
                (1)^2* sqrt(900 + x(2)^2)* (-1 + (1.0504* sqrt(900 + x(2)^2))/(x
                (1)* x(2)))^2) + (1.0504 *rb* sqrt(900 + x(2)^2))/(x(1)^2* x(2)^2
                *(-1 + (1.0504 *sqrt(900 + x(2)^2))/(x(1)* x(2)))^2)];
```

Anexo 8: trecho de código do problema 02 (barreira) (2/2)

```
                    [(2.1008 *rb* sqrt(900 + x(2)^2)* (1.0504/(x(1)* sqrt(900 + x(2)^2))
                        - (1.0504 *sqrt(900 + x(2)^2))/(x(1)* x(2)^2)))/(x(1)^2* x(2)*
                        (-1 + (1.0504 *sqrt(900 + x(2)^2))/(x(1)* x(2)))^3) + (rb*
                        (-1.0504/(x(1)^2 *sqrt(900 + x(2)^2)) + (1.0504* sqrt(900 + x(2)
                        ^2))/(x(1)^2* x(2)^2)))/(-1 + (1.0504 *sqrt(900 + x(2)^2))/(x(1)*
                         x(2)))^2, ...
                     (-2* rb* (1.0504/(x(1)* sqrt(900 + x(2)^2)) - (1.0504 *sqrt(900 + x
                        (2)^2))/(x(1)* x(2)^2))^2)/(-1 + (1.0504* sqrt(900 + x(2)^2))/(x
                        (1)* x(2)))^3 + (rb* ((-1.0504 *x(2))/(x(1)* (900 + x(2)^2)
                        ^(3/2)) - 1.0504/(x(1) *x(2)* sqrt(900 + x(2)^2)) + (2.1008*
                        sqrt(900 + x(2)^2))/(x(1)* x(2)^3)))/(-1 + (1.0504 *sqrt(900 + x
                        (2)^2))/(x(1)* x(2)))^2]];
                hessc2 = @(x) ...
                    [[(-2*rb* ((-740.22* x(1))/(900 + x(2)^2) - (1.0504 *sqrt(900 + x(2)
                        ^2))/(x(1)^2* x(2)))^2)/((-370.11* (0.01 + x(1)^2))/(900 + x(2)^2)
                         + (1.0504 *sqrt(900 + x(2)^2))/(x(1)* x(2)))^3 + (rb
                        *(-740.22/(900 + x(2)^2) + (2.1008 *sqrt(900 + x(2)^2))/(x(1)^3* x
                        (2))))/((-370.11* (0.01 + x(1)^2))/(900 + x(2)^2) + (1.0504* sqrt
                        (900 + x(2)^2))/(x(1)*x(2)))^2, ...
                     (-2* rb* ((740.22* (0.01 + x(1)^2)* x(2))/(900 + x(2)^2)^2 + 1.0504/(
                        x(1) *sqrt(900 + x(2)^2)) - (1.0504 *sqrt(900 + x(2)^2))/(x(1)* x
                        (2)^2)) *((-740.22* x(1))/(900 + x(2)^2) - (1.0504 *sqrt(900 + x
                        (2)^2))/(x(1)^2* x(2))))/((-370.11* (0.01 + x(1)^2))/(900 + x(2)
                        ^2) + (1.0504* sqrt(900 + x(2)^2))/(x(1)* x(2)))^3 + (rb*
                        ((1480.44* x(1)* x(2))/(900 + x(2)^2)^2 - 1.0504/(x(1)^2* sqrt
                        (900 + x(2)^2)) + (1.0504* sqrt(900 + x(2)^2))/(x(1)^2* x(2)^2)))
                        /((-370.11* (0.01 + x(1)^2))/(900 + x(2)^2) + (1.0504* sqrt(900 +
                         x(2)^2))/(x(1)* x(2)))^2];
                      [(-2* rb* ((740.22* (0.01 + x(1)^2)* x(2))/(900 + x(2)^2)^2 +
                        1.0504/(x(1) *sqrt(900 + x(2)^2)) - (1.0504 *sqrt(900 + x(2)^2))
                        /(x(1)* x(2)^2)) *((-740.22* x(1))/(900 + x(2)^2) - (1.0504 *
                        sqrt(900 + x(2)^2))/(x(1)^2* x(2))))/((-370.11* (0.01 + x(1)^2))
                        /(900 + x(2)^2) + (1.0504* sqrt(900 + x(2)^2))/(x(1)* x(2)))^3 +
                         (rb* ((1480.44* x(1)* x(2))/(900 + x(2)^2)^2 - 1.0504/(x(1)^2*
                        sqrt(900 + x(2)^2)) + (1.0504* sqrt(900 + x(2)^2))/(x(1)^2* x(2)
                        ^2)))/((-370.11* (0.01 + x(1)^2))/(900 + x(2)^2) + (1.0504* sqrt
                        (900 + x(2)^2))/(x(1)* x(2)))^2, ...
                      (-2*rb* ((740.22* (0.01 + x(1)^2)*x(2))/(900 + x(2)^2)^2 + 1.0504/(
                        x(1)* sqrt(900 + x(2)^2)) - (1.0504 *sqrt(900 + x(2)^2))/(x(1)*
                         x(2)^2))^2)/((-370.11* (0.01 + x(1)^2))/(900 + x(2)^2) +
                        (1.0504* sqrt(900 + x(2)^2))/(x(1)* x(2)))^3 + (rb* ((-2960.88*
                         (0.01 + x(1)^2)* x(2)^2)/(900 + x(2)^2)^3 + (740.22* (0.01 + x
                        (1)^2))/(900 + x(2)^2)^2 - (1.0504* x(2))/(x(1)* (900 + x(2)^2)
                        ^(3/2)) - 1.0504/(x(1)* x(2)* sqrt(900 + x(2)^2)) + (2.1008*
                        sqrt(900 + x(2)^2))/(x(1)* x(2)^3)))/((-370.11* (0.01 + x(1)^2)
                        )/(900 + x(2)^2) + (1.0504* sqrt(900 + x(2)^2))/(x(1)* x(2)))
                        ^2]];
                hess2= @(x) hessf(x)+hessc1(x)+hessc2(x);
                [x_] = osr_noprint(phib2, gphib2, hess2, x, m, 300, alphas(m), TOLS(m),
                    TOLS2(m));
                if b2(x)*rb/2 < TOLS_OCR(m)
                    conv=1;
                    fprintf('x=[%.4f,%.4f], f(x)=%.4f (%d iter/%.1fms)\n', x(1), x(2), f2(
                        x), k, toc(tstart)*1000);
                    break;
                end
                x = x_(:,end);
                x_values = [x_values,x];
                k=k+1;
                intervals = 10.^[1:.1:6];
                if m == plot_model
                    phi2_plot = @(x1,x2) ...
                          0.1885*x1.*sqrt(x2.^2+900) + ...
                          -rb./(1.0504*sqrt(x2.^2+900)./(x1.*x2) - 1) ...
                          -rb./(1.0504*sqrt(x2.^2+900)./(x1.*x2) - 370.11.*(x1.^2+0.01)./(
                              x2.^2+900));
                    c2a_plot = @(x1, x2) 1.0504.*sqrt(x2.^2+900)./(x1.*x2) - 1;
                    c2b_plot = @(x1, x2) 1.0504.*sqrt(x2.^2+900)./(x1.*x2) - 3.7011e2.*(x1
                        .^2+0.01)./(x2.^2+900);
                    tit = strcat(methods(m), ' (steps=',num2str(k),')');
                    plot_phi_c(1, 150, 1, 30, phi2_plot, intervals, c2a_plot, c2b_plot,
                        x_values, tit, k, i, j);
                end
            end
        end
```

Anexo 9: resultado da execução do script

```
>>
************* PROBLEMA 01 - PENALIDADE *********************
---Univariante---
x=[0.9448,0.8923], f(x)=1.9447 (7 iter/70.3ms)
---Powell---
x=[0.9456,0.8941], f(x)=1.9461 (6 iter/198.4ms)
---Steepest Descent---
x=[0.9456,0.8941], f(x)=1.9459 (8 iter/20.8ms)
---Fletcher Reeves---
x=[0.9456,0.8941], f(x)=1.9459 (8 iter/5.9ms)
---Newton-Raphson---
x=[0.9456,0.8941], f(x)=1.9461 (5 iter/3.9ms)
---BFGS---
x=[0.9456,0.8941], f(x)=1.9462 (7 iter/2.4ms)

************* PROBLEMA 01 - BARREIRA *********************
---Univariante---
x=[0.9467,0.8969], f(x)=1.9485 (6 iter/78.0ms)
---Powell---
x=[0.9453,0.8944], f(x)=1.9488 (11 iter/8615.5ms)
---Steepest Descent---
x=[0.9456,0.8948], f(x)=1.9485 (6 iter/797.9ms)
---Fletcher Reeves---
x=[0.9454,0.8944], f(x)=1.9485 (6 iter/297.1ms)
---Newton-Raphson---
x=[0.9456,0.8941], f(x)=1.9462 (10 iter/124.0ms)
---BFGS---
x=[0.9454,0.8944], f(x)=1.9485 (6 iter/239.1ms)

************* PROBLEMA 02 - PENALIDADE *********************
---Univariante---
x=[1.8784,20.2363], f(x)=12.8128 (15 iter/144.3ms)
---Powell---
x=[1.8783,20.2365], f(x)=12.8127 (8 iter/247.3ms)
---Steepest Descent---
x=[1.8784,20.2357], f(x)=12.8129 (14 iter/5302.9ms)
---Fletcher Reeves---
x=[1.8783,20.2350], f(x)=12.8124 (12 iter/1920.4ms)
---Newton-Raphson---
x=[1.8783,20.2365], f(x)=12.8127 (15 iter/528.7ms)
---BFGS---
x=[1.8783,20.2363], f(x)=12.8127 (13 iter/6433.7ms)

************* PROBLEMA 02 - BARREIRA *********************
---Univariante---
x=[1.8785,20.2389], f(x)=12.8143 (15 iter/13461.8ms)
---Powell---
x=[1.8785,20.2396], f(x)=12.8147 (13 iter/11053.0ms)
---Steepest Descent---
x=[1.8793,20.8871], f(x)=12.9492 (5 iter/9248.0ms)
---Fletcher Reeves---
x=[1.8855,20.3466], f(x)=12.8837 (5 iter/3708.0ms)
---Newton-Raphson---
x=[1.8784,20.2367], f(x)=12.8129 (8 iter/225.5ms)
---BFGS---
x=[1.8971,20.5055], f(x)=12.9950 (3 iter/1675.9ms)
>>
```

Referências

[Menezes, 2022] Menezes, I. F. M. (2022). Otimização: Algoritmos e aplicações na engenharia mecânica - notas de aula.

[Menezes et al., 2012] Menezes, I. F. M., Luiz, E. V., and Pereira, A. (2012). Programação matemática, teoria, algoritmos e aplicações.