UNIVERSIDADE FEDERAL DO CEARÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM E MÉTODOS QUANTITATIVOS

OTIMIZAÇÃO NÃO-LINEAR

PROJETO COMPUTACIONAL II

ISMAEL FERNANDES BRITO

FORTALEZA - CE
2022

1 Fundamentos Teóricos dos Métodos

1.1 Método da Máxima Descida

O **método do gradiente** (ou **método do máxima descida**) é um método numérico usado em otimização. Para encontrar um mínimo (local) de uma função usa-se um esquema iterativo, onde em cada passo se toma a direção (negativa) do gradiente, que corresponde à direção de descida máximo. Pode ser encarado como o método seguido por um curso da água, na sua descida pela força da gravidade.

O método da Máxima Descida, também conhecido como *Steepest Descent Method*, é considerado um método básico em otimização irrestrita. Neste método, a direção de busca $\mathbf{d}^k$ é dada pela direção contrária à direção do gradiente da função objetivo no ponto $\mathbf{x}^k$, ou seja, a direção de busca é a direção do anti-gradiente da função expressa por

$$\mathbf{d}^k = -\nabla f(\mathbf{x}^k).$$

Dessa forma, o método da máxima descida utiliza o procedimento iterativo $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$ com direção de busca e tamanho do passo α^k que pode ser determinado por alguma estratégia de busca linear. O Algoritmo 1, a seguir, ilustra um possível algoritmo para o método da máxima descida.

Algoritmo 1: Algoritmo para o Método da Máxima Descida

Dados $f(\mathbf{x})$ contínua e diferenciável com $\mathbf{x} \in \mathbb{R}^n$ e um ponto inicial $\mathbf{x}^0$
1: **Faça** k = 0
2: **Enquanto** $\nabla f(\mathbf{x}^k) \neq 0$, **faça**
3: $\quad \mathbf{d}^k = -\nabla f(\mathbf{x}^k)$
4: $\quad$ Obtenha $\alpha^k > 0$ tal que $f(\mathbf{x}^k + \alpha^k \mathbf{d}^k) < f(\mathbf{x}^k)$
5: $\quad \mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$
6: $\quad$ k = k + 1
7: **Retorne** $\mathbf{x}^k$

1.2 Método de Newton

Agora, vamos considerar o problema de minimização irrestrita

$$\min f(x) \text{e } x \in \mathbb{R}^n$$

em que $f : \mathbb{R}^n \to \mathbb{R}$ é uma função de classe C^2. Os pontos estacionários desse problema são caracterizados pela equação $\nabla f(x) = 0$. Vamos aplicar a relação $F(\overline{x}) + J_F(x - \overline{x}) = 0$ para $F : \mathbb{R}^n \to \mathbb{R}^n$ dada por

$$F(x) = \nabla f(x).$$

Seja $x_k \in \mathbb{R}^n$ uma aproximação de um ponto estacionário $\overline{x}$ do problema. A aproximação seguinte x_{k+1} é computada como solução do sistema de equações lineares

$$\nabla f(x_k) + \nabla^2 f(x_k)(x - x_k) = 0$$

em relação a $x \in \mathbb{R}^n$. Supondo que $\nabla^2 f(x_k)$ seja não-singular para todo $k \in \mathbb{N}$,obtemos o seguinte esquema iterativo:

$$x_{k+1} = x_k - \alpha_k (\nabla^2 f(x_k))^{-1} \nabla f(x_k), k = 0, 1, \cdots .$$

Assim, pode-se formalizar o algoritmo do método de Newton para minimizar uma função com tamanho de passo variável ($\alpha_k > 0$) da seguinte forma

Algoritmo 2: Algoritmo do método de Newton

Dado: $x_0 \in \mathbb{R}^n$
k = 0
REPITA enquanto $\nabla f(\mathbf{x}_k) \neq 0$
    Defina $d_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$
    Determine o tamanho do passo $\alpha_k > 0$
    Faça $x_{k+1} = x_k + \alpha_k d_k$
    k = k + 1

1.3 Método Quasi-Newton

O custo computacional elevado do método de Newton se deve à avaliação e uso da matriz Hessiana de f. A ideia por trás do método Quasi-Newton é realizar uma aproximação iterativa da inversa da matriz Hessiana.

O método Quasi-Newton utilizam o procedimento iterativo $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$ com direção de busca dada por

$$\mathbf{d}^k = -\mathbf{B}_k \nabla f(\mathbf{x}^k),$$

em que $\mathbf{B}_k$ é uma matriz simétrica definida positiva. Note que, se $\mathbf{B}_k = I_n$ (I_n é a matriz identidade de ordem n por n), a direção de busca $\mathbf{d}^k$ do método Quasi-Newton se transforma na direção do método da máxima descida e, se $\mathbf{B}_k = [\nabla^2 f(\mathbf{x}^k)]^{-1}$, a direção de busca se transforma na direção do método de Newton.

O método Quasi-Newton diferem entre si na forma como as atualizações da matriz $\mathbf{B}_k$ são realizadas. Essa matriz deve se aproximar da inversa da matriz Hessiana de f a cada iteração k. Logo, dados

$$\mathbf{r}^k = \mathbf{x}^{k+1} - \mathbf{x}^k,$$

e

$$\mathbf{s}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k),$$

a matriz $\mathbf{B}_k$ pode ser atualizada pela expressão

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\mathbf{r}^k (\mathbf{r}^k)^T}{(\mathbf{r}^k)^T \mathbf{s}^k} - \frac{\mathbf{B}_k \mathbf{s}^k (\mathbf{B}_k \mathbf{s}^k)^T}{(\mathbf{s}^k)^T \mathbf{B}_k \mathbf{s}^k} + c(\mathbf{s}^k)^T \mathbf{B}_k \mathbf{s}^k \mathbf{v}^k (\mathbf{v}^k)^T,$$

com

$$\mathbf{v}^k = \frac{\mathbf{r}^k}{(\mathbf{r}^k)^T \mathbf{s}^k} - \frac{\mathbf{B}_k \mathbf{s}^k}{(\mathbf{s}^k)^T \mathbf{B}_k \mathbf{s}^k}$$

Para $c = 0$ na expressão de cima, tem-se a fórmula de Davidson-Fletcher-Powell (DFP).

Dessa forma, o procedimento iterativo com a direção de busca requer, além do ponto inicial $\mathbf{x}^k$, uma matriz $\mathbf{B}_0$ simétrica positiva. Em geral, toma-se $\mathbf{B}_0 = I_n$. Para completar o esquema iterativo, o tamanho do passo α^k deve ser obtido por uma técnica de busca linear exata ou inexata.

Algoritmo 3: Método de Davidon - Fletcher - Powell (DFP)

Escolha $\epsilon > 0$, x^0 e $H_0 = H_0^T$. Calcule $g^0 = \nabla f(x^0)$ e faça $k = 0$
while $||g^0|| \geqslant \epsilon$
  $d^k = -H_k g^0$
  $\alpha_k = \arg\min_{\alpha \geqslant 0} f(x^k + \alpha d^k)$
  $x^{k+1} = x^k + \alpha_k d^k$
  if $k < n - 1$ **then**
    $g^{k+1} = \nabla f(x^{k+1})$
    $q^k = g^{k+1} - g^k, p^k = \alpha_k d^k$
    $H_{k+1} = H_k - \dfrac{p^k (p^k)^T}{(p^k)^T q^k} - \dfrac{H_k q^k (q^k)^T H_k}{(q^k)^T H_k q^k}$
    $k = k + 1$
    else
    $x^0 = x^n, g^0 = \nabla f(x^0), k = 0$
  end
end

1.4 Método do Gradiente Conjugado

O método do gradiente conjugado foi desenvolvido com o objetivo de acelerar a taxa de convergência do método da máxima descida e, ao mesmo tempo, evitar o alto custo computacional do método de Newton. Tal como o método de Newton, o método do gradiente conjugado foi desenvolvido para obter soluções exatas em um número finito de iterações quando a função objetivo é quadrática, com matriz **A** simétrica definida positiva.

Inicialmente, considere o método das direções conjugadas para a minimização de uma função quadrática, com matriz **A** simétrica definida. Este método constrói um conjunto de n direções $\{\mathbf{d}^0, \cdots, \mathbf{d}^{n-1}\}$**A**-ortogonais, ou seja, direções conjugadas com respeito à matriz **A** em que

$$(\mathbf{d}^i)^T \mathbf{A} \mathbf{d}^j = 0, \forall i \neq j.$$

Para qualquer matriz **A** de ordem nxn simétrica positiva definida, qualquer conjunto de direções **A**-ortogonais é linearmente independente. O método do gradiente conjugado, por sua vez, é o método das direções conjugadas em que as direções de busca correspondem a uma versão conjugada dos sucessivos gradientes obtidos ao longo do procedimento iterativo. Logo, considere o procedimento iterativo $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$ e um ponto inicial $\mathbf{x}^0 \in \mathbb{R}^n$. As direções de busca do método do gradiente conjugado são obtidas por

$$\mathbf{d}^0 = -\nabla f(\mathbf{x}^0),$$

e

$$\mathbf{d}^k = -\nabla f(\mathbf{x}^k) + \delta^{k-1} \mathbf{d}^{k-1}, k > 0,$$

com

$$\delta^{k-1} = \frac{\mathbf{g}^{k^T} \mathbf{A} \mathbf{d}^{k-1}}{\mathbf{d}^{k-1^T} \mathbf{A} \mathbf{d}^{k-1}}$$

onde $\mathbf{g}^k = \nabla f(\mathbf{x}^k)$.

Note que, a primeira direção de busca é a direção de máxima descida. Em todas as demais iterações, a direção de busca será a soma da direção contrária à direção do vetor gradiente no ponto atual com a combinação linear das direções de busca anteriores. O Algoritmo 2 do método do gradiente conjugado para funções não-lineares, a seguir, ilustra um possível algoritmo para o método do gradiente conjugado.

Algoritmo 4: Método de Fletcher-Reeves

Escolha $\epsilon > 0$ e x^0. Calcule $g^0 = \nabla f(x^0)$
while $||g^0|| \geqslant \epsilon$
  $d^0 = -g^0$
  for $k = 0 : n-1$
    $\alpha_k = \arg\min_{\alpha \geqslant 0} f(x^k + \alpha d^k)$
    $x^{k+1} = x^k + \alpha_k d^k, g^{k+1} = \nabla f(x^{k+1})$
    if $k < n-1$ **then**
      $\beta_k = \dfrac{(g^{k+1})^T g^{k+1}}{(g^k)^T g^k}$
      $d^{k+1} = -g^{k+1} + \beta_k d^k$
    end
  end
  $x^0 = x^n, g^0 = \nabla f(x^0)$
end
$x^* = x^0$

1.5 Método de Polak-Ribière (PR)

As derivações das equações anteriores foram feitas supondo que estamos tratando de problemas quadráticos, o que nem sempre é verdade. Para que possamos adaptar as equações anteriores a problemas não-quadráticos, a matriz $\mathbb{A}$ deve ser aproximada pela matriz hessiana calculado no ponto x_i. A aplicação destes algoritmos a problemas não quadráticos envolve um procedimento de busca unidimensional do passo de ajuste (taxa de aprendizagem) e a aproximação do parâmetro δ utilizando informações de primeira ordem (gradientes).

Uma destas aproximações é dada pelo método de Polak-Ribière. O algoritmo para este método torna-se então:

Algoritmo 5: Método de Polak-Ribière

1. Atribua um valor inicial $x^0 \in \mathbb{R}$ para o vetor de parâmetros e um valor arbitrariamente pequeno para a constante $\epsilon > 0$.
2. Calcule $\nabla f(x^0)$, faça $d^0 = -\nabla f(x^0)$
3. Enquanto a condição de parada não for satisfeita, faça:
   3.1. Utilize um procedimento de busca unidimensional para encontrar um α_k que seja solução ótima do problema $\min f(x^k + \alpha_k d^k)$ e faça $x^{k+1} = x^k + \alpha_k d^k$ e $\alpha_k \in (0, 1]$
   3.2. Calcule $g^k = -\nabla f(x^k)$
   3.3. Se (k < n -1), faça $d^{k+1} = g^k + \beta_k d^k$, onde $\beta_k = \dfrac{(g^{k+1})^T (g^{k+1} - g^k)}{(g^k)^T g^k}$
   3.4. Senão, faça: $d^k = g^k$
   3.5. Faça k = k + 1

1.6 Região de Confiança

Considere o seguinte problema de programação diferenciável não linear sem restrições:

$$(P) = \begin{cases} \min f(x) \\ x \in \mathbb{R}^n \end{cases}$$

onde $f : \mathbb{R}^n \longmapsto \mathbb{R}$ é de classe $C^2(\mathbb{R}^n)$. Observação: Como f não é necessariamente convexa, a matriz $\nabla^2 f(x)$ pode não ser definida positiva, apesar de ser simétrica. Neste caso, o método de Newton ou suas variantes (direções conjugadas, quasi Newton, etc) não servem.

O objetivo principal do método de região de confiança é resolver o problema (P). É um método iterativo que gera uma sequência $\{x_k\}_{k \in \mathbb{N}}$ convergindo para um minimizador local de f.

Em cada iteração é construído um modelo quadrático de f e minimiza-se este modelo em uma região, daí o nome *região de confiança*. Dependendo do resultado obtido com esta minimização, algumas decisões são tomadas:

- aceitar ou não o minimizador como um novo ponto (uma nova iterada);
- aumentar, reduzir, ou deixar inalterada a região de confiança.

Antes de iniciar o algoritmo, algumas informações precisam ser fornecidas:

- a função $f : \mathbb{R}^n \to \mathbb{R}$;
- ponto inicial x_0;
- o raio da região de confiança Δ_0;

Em cada iteração x_k, construímos um modelo quadrático $m_k(p)$, esperando que este modelo seja uma boa representação da variação de f na vizinhança de x_k, e resolvemos o problema (P).

O método de região de confiança será uma generalização da busca de Armijo, consistindo da construção de um modelo quadrático e uma região R, chamada de região de confiança, e nessa região calcular o novo iterando.

Algoritmo 6: Método de Região de Confiança

Entrada: função $f : \mathbb{R}^n \to \mathbb{R}$, raio da bola Δ_0, raio máximo $\overline{\Delta}, \epsilon, \eta \in [0, \frac{1}{4})$, ponto inicial x_0
Saída: minimizador x^*

$curv \longleftarrow \Delta f(x_k)^T B \Delta f(x_k)$
se $curv \leqslant 0$ **então**
  $pd \longleftarrow -\Delta \dfrac{\nabla f(x_k)}{||\nabla f(x_k)||}$
senão
  $p^C \longleftarrow -\dfrac{\nabla f(x_k)^T \nabla f(x_k)}{curv} \nabla f(x_k)$
  se $||p^C|| \geqslant \Delta$ **então**
    $pd \longleftarrow -\Delta \dfrac{\nabla f(x_k)}{||\nabla f(x_k)||}$
  senão
    se $B \not\succ 0$ **então**
      Informar que B não é definida positiva.
      $pd \longleftarrow p^C$
    senão
      $p^N \longleftarrow -B^{-1} \Delta f(x_k)$
      se $||p^N|| \leqslant \Delta$ **então**
        $pd \longleftarrow p^N$
      senão
        $\lambda \longleftarrow \dfrac{-2p^{C^T}(p^N - p^C) + \sqrt{(2p^{C^T}(p^N - p^C))^2 - 4||p^N - p^C||^2(||p^C||^2 - \Delta^2)}}{2||p^N - p^C||^2}$
        $pd \longleftarrow p^C + \lambda(p^N - p^C)$
      fim
    fim
  fim
fim

Algoritmo 7: Método Dogleg

Entrada: matriz $B(nxn)$ positiva definida, vetor $\nabla f(x_k)(nx1)$, raio da bola Δ
Saída: passo do dogleg pd $(nx1)$

$k \longleftarrow 0$
Calcular $\nabla f(x_k)$ e $B_k = \nabla^2 f(x_k)$
enquanto $||\nabla f(x_k)|| > \epsilon$ **faça**
    $p_k \longleftarrow$ solução de minimizar $m_k(p) = p^T \nabla f(x_k) + \frac{1}{2} p^T B_k p$ sujeito a $||p|| \leqslant \Delta_k$
    $\rho_k \longleftarrow \frac{ared}{pred} = \frac{f(x_k) - f(x_k + p_k)}{-m_k(p_k)}$
    se $\rho_p < \frac{1}{4}$ **então**
        $\Delta_{k+1} \longleftarrow \frac{1}{4}||p_k||$
    senão
        se $\rho_k > \frac{3}{4}$ e $||p_k|| = \Delta_k$ **então**
            $\Delta_{k+1} \longleftarrow \min(2\Delta_k, \overline{\Delta})$
        senão
            $\Delta_{k+1} \longleftarrow \Delta_k$
        fim
    fim
    se $\rho_k > \eta$ **então**
        $x_{k+1} \longleftarrow x_k + p_k$
        Calcular $\Delta f(x_{k+1})$ e $B_{k+1} = \Delta^2 f(x_{k+1})$
    senão
        $x_{k+1} \longleftarrow x_k$
    fim
    $k \longleftarrow k + 1$
fim
$x^* \longleftarrow x_k$

2 Manual de utilização dos métodos no Octave

O GNU Octave é uma linguagem de alto nível, destinada principalmente a cálculos numéricos. Ele fornece uma interface de linha de comando conveniente para resolver problemas lineares e não lineares numericamente usando uma linguagem que é principalmente compatível com o Matlab.

A motivação original para escrever o Octave era fornecer um software para acompanhar um livro de graduação em engenharia química (Análise de Reatores Químicos e Designs Fundamentais, em inglês - Chemical Reactor Analysis and Design Fundamentals, publicado em 2002 e escrito por Jim Rawlings e John Ekerdt). No entanto, Octave evoluiu, com o tempo, tornando-se uma ferramenta útil para a área de computação numérica e de construção de gráficos, as quais são usadas para uma grande variedade de tarefas (EATON, 2020).

O Octave, além disso, é uma plataforma livre desenvolvida por uma comunidade de usuários, ou seja, é distribuída sob termos que garantem certas liberdades para seus usuários, como a liberdade de executar, copiar, distribuir, estudar, alterar e melhorar a plataforma (EATON, 2020).

As funções, bem como o vetor gradiente e também a matriz hessiana no pontos iniciais foram armazenadas em arquivo texto padrão do Matlab/Octave com o nome de "funcoes.m"que deve ser carregado na memória antes de fazer a chamada dos métodos computacionais. Neste arquivo contém todas as 5 funções pedidas que foram criadas como f1, f2, f7, f8 e f9 juntamente com as seus respectivos vetores gradientes criadas como df1, df2, df7, df8 e df9 que serão utilizadas nos métodos e também os vetores de entrada $x^{(1)}$ foram criados como f1x1, f2x1, f7x1, f8x1 e f9x1 e para o $x^{(2)}$ foram criados como f1x2, f2x2, f7x2, f8x2 e f9x2 que serão utilizados nos métodos.

Outra entrada utilizada será a informação do intervalo de incerteza (0,1) que será utilizada no método da razão áurea como vetor linha da seguinte forma [0 1] que será calculado para encontrar o tamanho do passo α_k a ser utilizado nos métodos. Em todos os métodos tem um parâmetro $nmax$

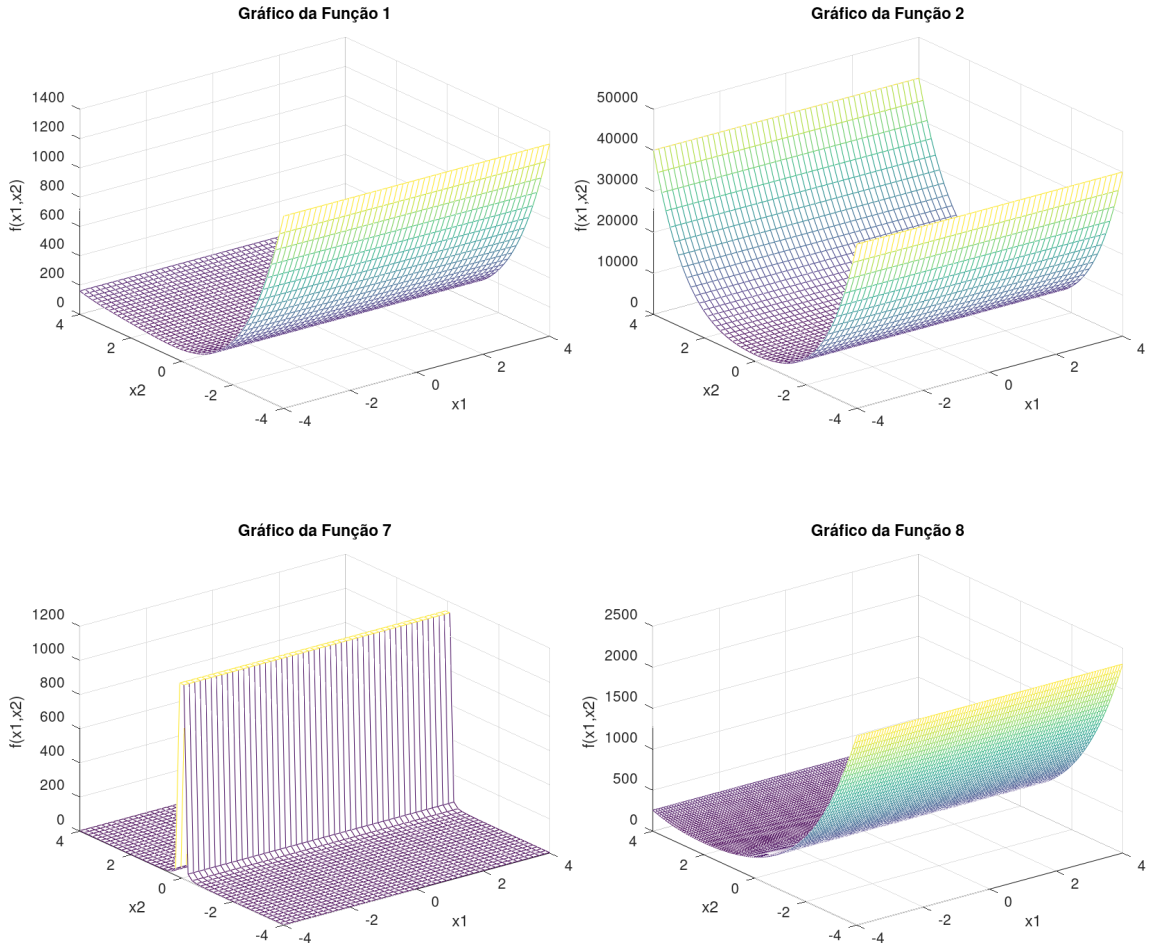
que significa o número máximo de iterações e o parâmetro *tol* que significa o valor da precisão utilizado pelos métodos.

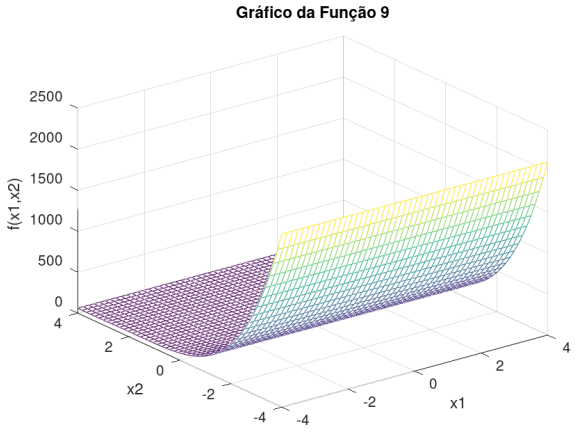
Em geral a sintaxe para executar os métodos comupacionais utiliza-se o shell do Octave da seguinte forma: "nome-metodo(parametros)". A saída será o valor do número de iterações k até o método convergir ou ser parado pelo número máximo de iterações $nmax$, e de xmin para o valor de x_{min}, o valor de fval para $f(x_{min})$. Já para o método da razão aurea foi feito uma chamada dentro de cada método que utiliza esse parâmetro e a saída será alphak para o valor de α_k.

Por fim, todos os métodos foram implementados como funções e salvas com extensão ".m"que é o padrão do Octave. Os arquivos dos mesmos irão junto com este trabalho. Também será produzido um vídeo explicativo de como pode ser feito as execuções dos métodos.

3 Resultados dos Experimentos

3.1 Gráficos das Funções





3.2 Funções Utilizadas nos Testes

* $f_1(\mathbf{x}) = (x_1 - 2)^2 + (x_1 - 2x_2)^2$

  (i) $\mathbf{x}^1 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}$

  (ii) $\mathbf{x}^2 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$

* $f_2(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

  (i) $\mathbf{x}^1 = \begin{pmatrix} -1,9 \\ 2 \end{pmatrix}$

  (ii) $\mathbf{x}^2 = \begin{pmatrix} 1,2 \\ 1 \end{pmatrix}$

* $f_7(\mathbf{x}) = 0,1\left(12 + x_1^2 + \frac{1 + x_2^2}{x_1^2} + \frac{x_1^2 x_2^2 + 100}{x_1^4 x_2^4}\right)$

  (i) $\mathbf{x}^1 = \begin{pmatrix} 0,5 \\ 0,5 \end{pmatrix}$

  (ii) $\mathbf{x}^2 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$

* $f_8(\mathbf{x}) = (x_1^2 + x_2^2 + x_1 x_2)^2 + sen^2(x_1) + cos^2(x_2)$

  (i) $\mathbf{x}^1 = \begin{pmatrix} 3 \\ 0,1 \end{pmatrix}$

  (ii) $\mathbf{x}^2 = \begin{pmatrix} 2 \\ -2 \end{pmatrix}$

* $f_9(\mathbf{x}) = 1,41x_1^4 - 12,76x_1^3 + 39,91x_1^2 - 51,93x_1 + 24,37 + (x_2 - 3,9)^2$

  (i) $\mathbf{x}^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

  (ii) $\mathbf{x}^2 = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$

Em todos os métodos implementados foi utilizado os seguintes parâmetros:

* tol $= 10^{-04}$ (precisão)

* nmax $= 1000$ (número máximo de iterações)

Tabela 1: Dados utilizados nos testes

Função	**Vetor gradiente**	**Pontos iniciais**	**Matrizes hessianas**
f1	df1	f1x1 e f1x2	H1 e H2
f2	df2	f2x1 e f2x2	H3 e H4
f7	df7	f7x1 e f7x2	H5 e H6
f8	df8	f8x1 e f8x2	H7 e H8
f9	df9	f9x1 e f9x2	H9 e H10

3.3 Método da Máxima Descida

Tabela 2: Resultados dos testes utilizando o como ponto inicial $\mathbf{x}^1$

Função	**Nº de iterações**	**xmin**	**fval**
1	591	[2.0653 1.0327]	1.8196e-05
2	1000	[-1.1461 1.3214]	4.6119
7	1000	[0.5000 0.5000]	2563.3
8	145	[0.1557 -0.6948]	0.7732
9	63	[3.4827 3.8996]	-3.9872

Tabela 3: Resultados dos testes utilizando o como ponto inicial $\mathbf{x}^2$

Função	**Nº de iterações**	**xmin**	**fval**
1	1000	[1.9067 0.9532]	7.6031e-05
2	1000	[1.0247 1.0501]	6.1060e-04
7	1000	[2.3933 2.9205]	1.9454
8	12	[-0.1552 0.6944]	0.7732
9	238	[3.4827 3.9005]	-3.9872

3.4 Método de Newton

Tabela 4: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^1$

Função	**Nº de iterações**	**xmin**	**fval**
1	1000	[1.6920 0.8460]	8.9998e-03
2	1000	[-1.8787 3.4509]	8.9029
7	1000	[0.4290 0.5892]	2453.9
8	1000	[0.5876 0.2610]	1.5619
9	559	[1.3476 3.9000]	0.2896

Tabela 5: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^2$

Função	**Nº de iterações**	**xmin**	**fval**
1	1000	[1.1743 0.5872]	0.4648
2	1000	[1.1757 1.3821]	0.030864
7	4	[NaN NaN]	NaN
8	300	[0.1629 -0.7013]	0.7733
9	1000	[3.5301 3.9000]	-3.9657

3.5 Método Quasi-Newton

Tabela 6: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^1$

Função	Nº de iterações	xmin	fval
1	328	[1.9973 0.9987]	5.1919e-11
2	1000	[-1.7398 2.9492]	8.1099
7	1000	[0.5231 0.5191]	1841.3
8	490	[-0.1545 0.6947]	0.7732
9	132	[1.3578 3.8996]	0.2892

Tabela 7: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^2$

Função	Nº de iterações	xmin	fval
1	1000	[1.9931 0.9965]	2.3060e-09
2	862	[1.0005 1.0009]	2.2472e-07
7	1000	[-21.827 -28.870]	49.018
8	60	[0.1559 -0.6954]	0.7732
9	538	[3.4837 3.9002]	-3.9872

3.6 Método do Gradiente Conjugado

3.6.1 Método de Fletcher-Reeves

Tabela 8: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^1$

Função	Nº de iterações	xmin	fval
1	231	[2.0582 1.0292]	1.1538e-05
2	1000	[1.6213 2.6307]	0.3864
7	22	[NaN NaN]	NaN
8	12	[0.1557 -0.6948]	0.7732
9	36	[3.4827 3.8996]	-3.9872

Tabela 9: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^2$

Função	Nº de iterações	xmin	fval
1	234	[1.9418 0.9709]	1.1481e-05
2	1000	[1.1796 1.3925]	0.032364
7	1000	[1.7436 2.5720]	1.7842
8	9	[-0.1552 0.6944]	0.7732
9	30	[3.4827 3.9003]	-3.9872

3.6.2 Método de Polak-Ribière

Tabela 10: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^1$

Função	Nº de iterações	xmin	fval
1	231	[2.0581 1.0292]	1.1438e-05
2	1000	[1.5796 2.4962]	0.3361
7	3	[NaN NaN]	NaN
8	10	[0.1556 -0.6946]	0.7732
9	37	[3.4827 3.8996]	-3.9872

Tabela 11: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^2$

Função	Nº de iterações	xmin	fval
1	232	[1.9417 0.9709]	1.1559e-05
2	491	[1.0011 1.0021]	1.1306e-06
7	1000	[-1.8210 3.0410]	1.8545
8	6	[-0.1553 0.6945]	0.7732
9	33	[3.4827 3.9003]	-3.9872

3.7 Método da Região de Confiança

Para este método foi utilizado os seguintes parâmetros:

* $\Delta_0 = 1$
* $\eta = \frac{1}{8}$

Tabela 12: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^1$

Função	Nº de iterações	xmin	fval
1	1000	[1.9229 0.9614]	3.5421e-05
2	1000	[-5.2120e-02 4.0153e-03]	1.1071
7	1000	[1.0065 1.0193]	10.624
8	239	[-0.1552 0.6946]	0.7732
9	134	[1.3586 3.9000]	0.2891

Tabela 13: Resultados dos testes utilizando como ponto inicial $\mathbf{x}^2$

Função	Nº de iterações	xmin	fval
1	1000	[1.8843 0.9421]	1.7948e-04
2	666	[1.0003 1.0005]	7.4737e-08
7	1000	[3 3]	2.2139
8	75	[0.1557 -0.6948]	0.7732
9	55	[3.4827 3.9000]	-3.9872

Referências

[1] https://pt.wikibooks.org/wiki/Otimiza%C3%A7%C3%A3o/M%C3%A9todos_de_regi%C3%A3o_de_confian%C3%A7a

[2] John W. Eaton, David Bateman, Søren Hauberg, Rik Wehbring (2020). GNU Octave version 6.1.0 manual: a high-level interactive language for numerical computations. https://www.gnu.org/software/octave/doc/v6.1.0/

[3] https://www.ime.unicamp.br/~sandra/MS629/handouts/livro28jul.pdf

[4] https://www.ime.unicamp.br/~friedlan/livro.pdf

[5] Nonlinear Programming - M. S. Bazaraa & C. M. Shetty - John Wiley & Sons, 1979.