UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

EEL 5102-47 – Métodos Numéricos de Otimização I

Trabalho Computacional utilizando os métodos de Gradiente, Quase-Newton e Newton

Aluno: João José Medeiros de Figueiredo
Matrícula: 202302284

Professor: Erlon Cristian Finardi, Phd.

Florianópolis, abril de 2024.

1. Introdução

Este trabalho visa a implementação dos métodos de Gradiente, Quase-Newton e Newton para encontrar um ponto estacionário da função $f(x,y) = x^4 - 2x^2y + x^2 + y^2 - 2x + 5$, utilizando como ponto inicial: $[1\ 4]^T$, tolerância de 10^{-8} e busca linear inexata que implementa as condições fortes de Wolfe com interpolação quadrática.

Para a implementação escolhi a linguagem de programação Python, os códigos estarão fornecidos nos anexos finais.

2. Desenvolvimento

Primeiramente vamos analisar algumas características da função em estudo, como sua hessiana, e a convexidade da matriz.

2.1 Hessiana

A Hessiana é uma matriz quadrada de derivadas segundas de uma função escalar, para a nossa função em estudo a hessiana é montada seguindo os seguintes passos:

- Derivando a função:$f(x,y) = x^4 - 2x^2y + x^2 + y^2 - 2x + 5$ em relação a x temos: $\frac{\partial f}{\partial x} = 4x^3 - 4xy + 2x - 2$, derivando novamente obtemos: $\frac{\partial^2 f}{\partial x^2} = 12x^2 - 4y + 2$.
- Derivando em relação a y, temos: $\frac{\partial f}{\partial y} = -2x^2 + 2y$, derivando novamente obtemos: $\frac{\partial^2 f}{\partial y^2} = 2$.
- Se derivarmos: $\frac{\partial f}{\partial x} = 4x^3 - 4xy + 2x - 2$ em relação a y, ou se derivarmos $\frac{\partial f}{\partial y} = -2x^2 + 2y$ em relação a x, obtemos: $\frac{\partial^2 f}{\partial xy} = -4x$.

Realizando a montagem da matriz, ficamos com:

$$\begin{bmatrix} 12x^2 - 4y + 2 & -4x \\ -4x & 2 \end{bmatrix}$$

2.2 Convexidade

A convexidade da função formalmente é definida como:

$$f(\alpha x + (1+\alpha)y \leq \alpha f(x) + (1-\alpha)f(y), \forall\, x, y\ \in C$$

Sendo $C\ um\ conjunto\ convexo\ de\ R^n$, e um conjunto convexo é definido como:

$$\alpha x + (1-\alpha)y \in C, \forall x, y \in C, \forall \alpha \in (0,1)$$

Em termos práticos, podemos testar um ponto aleatório na matriz hessiana e verificar seus autovalores, se todos os autovalores forem maiores ou iguais a zero, ou seja a matriz não é Indefinida nem Definida Negativa, temos fortes indícios de que a função é convexa.

Por exemplo, escolhendo o ponto (2,2) para nossa matriz hessiana, ficamos com:

$$\begin{bmatrix} 42 & -8 \\ -8 & 2 \end{bmatrix}$$

Sendo I a matriz Identidade:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Resolvendo $\det(A - \lambda I)$ temos:

$$det \begin{bmatrix} 42-\lambda & -8 \\ -8 & 2-\lambda \end{bmatrix}$$

Analisando o determinante:

$$(42-\lambda)(2-\lambda) - 64 = 0$$

$$\lambda^2 - 44\lambda + 20 = 0$$

E resolvendo:

$$\lambda = \frac{44 \pm \sqrt{1936 - 80}}{2}$$

Obtemos:

$$\lambda = \frac{44 \pm 43.08}{2}$$

$$\lambda 1 = 0.46$$

$$\lambda 2 = 43.54$$

Portanto, a função pode ser convexa, baseando-se na análise de um ponto arbitrário.

2.3 Intervalos desejados para α_k

Vamos exemplificar as condições de decréscimo suficiente (ou condições de Armijo), condições de curvatura, condições de Wolfe (que nada mais são do que a soma das condições de Armijo com as condições de curvatura), e por fim analisar as condições fortes de Wolfe (que são as condições de Armijo somadas ao módulo das condições de curvatura).

2.3.1 Condições de decréscimo suficiente (Armijo)

$$\boldsymbol{f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k}$$

Onde:

A_k é o tamanho do passo

p_k é a direção de descida

$\nabla f(x_k)^T$ é o gradiente da função objetivo em x_k

c_1 é uma constante pequena (por exemplo, 0,0001) que varia de (0,1). Esta condição garante que o passo realmente reduza o valor da função.

2.3.2 Condições de curvatura

$$\boldsymbol{\nabla f(x_k + \alpha_k p_k)^T d_k \geq c_2 \nabla f(x_k)^T p_k}$$

onde

$\boldsymbol{c_2}$ é uma constante pode variar de (c_1,1)

2.3.3 Condições de Wolfe

As condições de Wolfe nada mais são do que a soma das condições de decréscimo suficiente e as condições de curvatura

$$\boldsymbol{f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k + \nabla f(x_k + \alpha_k p_k)^T d_k \geq c_2 \nabla f(x_k)^T p_k}$$

2.3.4 Condições fortes de Wolfe

As condições fortes de Wolfe nada mais são do que a soma das condições de decréscimo suficiente com o módulo das condições de curvatura

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k + |\nabla f(x_k + \alpha_k p_k)^T p_k| \geq c_2 |\nabla f(x_k)^T p_k|$$

Esta versão mais forte garante que o gradiente no novo ponto não só não esteja muito alinhado com a direção de descida (como nas condições de Wolfe regulares) mas também que o componente do gradiente na direção de descida tenha diminuído suficientemente em magnitude. Isso ajuda a prevenir passos excessivamente longos que podem ocorrer em áreas de curvatura baixa da função objetivo.

2.3.5 Verificação das condições fortes de Wolfe na função de estudo

Para a exemplificação e aplicação prática dos conceitos citados, vamos fazer a verificação das condições fortes de Wolfe na nossa função de estudo $f(x,y) = x^4 - 2x^2y + x^2 + y^2 - 2x + 5$ utilizando como ponto inicial $x_0 = [1\ 4]^T$ como direção inicial $p_0 = [1\ -1]^T$, com o valor tradicional de α_k para a primeira iteração $\alpha_0 = 1$ com $c_1 = 0.0001$ e $c_2 = 0.9$.

Verificando a condição de Armijo:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k$$

Temos:

$$f(x_0, y_0) = f(1,4) = 13$$

$$x_0 + \alpha_0 p_0 = \begin{bmatrix}1\\4\end{bmatrix} + 1\begin{bmatrix}1\\-1\end{bmatrix} = \begin{bmatrix}2\\3\end{bmatrix}$$

$$f(x_0 + \alpha_0 p_0) = f(2,3) = 6$$

$$\nabla f(x_k, y_k) = \begin{bmatrix}4x^3 - 4xy + 2x - 2\\ -2x^2 + 2y\end{bmatrix}$$

$$\nabla f(x_0, y_0) = \begin{bmatrix}-12\\6\end{bmatrix}$$

$$\nabla f(x_0, y_0)^T p_0 = [-12 \quad 6]\begin{bmatrix}1\\-1\end{bmatrix} = -18$$

Decréscimo suficiente:

$$6 \leq 13 + 0.0001 * 1(-18)$$

$$6 \leq 12.9982$$

Portanto o critério de decréscimo suficiente (Armijo) é satisfeito

Verificando a condição de Curvatura:

$$\nabla f(x_k + \alpha_k p_k)^T d_k \geq c_2 \nabla f(x_k)^T p_k$$

Temos:

$$f(x_0, y_0) = f(1,4) = 13$$

$$x_0 + \alpha_0 p_0 = \begin{bmatrix}1\\4\end{bmatrix} + 1\begin{bmatrix}1\\-1\end{bmatrix} = \begin{bmatrix}2\\3\end{bmatrix}$$

$$\nabla f(x_k, y_k) = \begin{bmatrix}4x^3 - 4xy + 2x - 2\\ -2x^2 + 2y\end{bmatrix}$$

$$\nabla f(x_0, y_0)^T = [-12 \quad 6]$$

$$\nabla f(x_k + \alpha_k p_k)^T = [12 \quad 2]$$

$$\nabla f(x_k + \alpha_k p_k)^T d_k = [12 \quad 2].\begin{bmatrix}1\\-1\end{bmatrix} = 10$$

$$c_2 \nabla f(x_k)^T p_k = 0.9[-12 \quad 6].\begin{bmatrix}1\\-1\end{bmatrix} = -16.2$$

$$10 \geq -16.2$$

A condição de curvatura é satisfeita. Porém se formos analisar as condição de curvatura presente nas condições fortes de Wolfe, ficaríamos com:

$$10 \geq 16.2$$

E portanto a condição não seria satisfeita.

Se não fosse satisfeito deveríamos realizar a interpolação quadrática:

$$\alpha_1 = -\frac{\varphi'(0)\alpha_0^2}{2[\varphi(\alpha_0) - \varphi(0) - \varphi'(0)\alpha_0]}$$

Sendo:

$$\varphi(\alpha) = f(x_k + \alpha_k p_k)$$
$$\varphi'(0) = \nabla f(x_k)^T p_k$$
$$\varphi(0) = f(x_k)$$

E verificar se esse novo valor de α_1 satisfaz as condições fortes de Wolfe. Geralmente é utilizado um processo iterativo chamado de BackTracking Line Search para achar valores de α_k que satisfaçam as condições fortes de Wolfe.

2.4 Método Gradiente

O método foi implementado em python, e o código fonte está disponível no github: *https://github.com/joaojosemfigueiredo/EEL-5102-47-Metodos-Numericos-de-Otimizacao-I-*

Utilizando como parâmetros iniciais:

- $ponto\ inicial$: $[1\ 4]^T$
- $tolerância\ de\ 10^{-8}$
- $max_{iter} = 1000$
- $c_1 = 0.1$
- $c_2 = 0.9$
- $alpha_{init} = 1$

O método atingiu convergência em 317 iterações. Abaixo, na tabela 1, podemos conferir as 10 primeiras iterações:

k	(x_1)^k	(x_2)^k	(p_1)^k	(p_2)^k	norma do gradiente	alpha_utilizado	f(x_k)	Tempo Decorrido (s)
0	1	4	12	-6	13,416	0,062	4,879	0,012
1	1,75	3,625	2,438	-1,125	2,685	0,062	4,818	0,023
2	1,902	3,555	-2,293	0,128	2,297	0,031	4,733	0,037
3	1,831	3,559	-0,143	-0,415	0,439	1	4,56	0,039
4	1,688	3,144	0,625	-0,593	0,861	0,062	4,544	0,049
5	1,727	3,107	-0,583	-0,252	0,636	0,062	4,531	0,059
6	1,69	3,091	0,207	-0,47	0,513	0,125	4,52	0,067
7	1,716	3,033	-0,828	-0,176	0,847	0,062	4,505	0,078
8	1,664	3,022	0,349	-0,504	0,613	0,062	4,492	0,089
9	1,686	2,99	-0,378	-0,295	0,479	0,125	4,48	0,099
10	1,639	2,953	0,476	-0,535	0,716	0,062	4,465	0,11

Tabela 1 – Método do Gradiente (10 primeiras iterações)

Abaixo na tabela 2 podemos ver as 10 últimas iterações:

k	(x_1)^k	(x_2)^k	(p_1)^k	(p_2)^k	norma do gradiente	alpha_utilizado	f(x_k)	Tempo Decorrido (s)
306	1	1	0	0	0	0,484	4	2,93
307	1	1	0	0	0	0,125	4	2,941
308	1	1	0	0	0	0,125	4	2,951
309	1	1	0	0	0	0,016	4	2,967
310	1	1	0	0	0	0,859	4	2,984
311	1	1	0	0	0	0,062	4	2,997
312	1	1	0	0	0	0,234	4	3,014
313	1	1	0	0	0	0,016	4	3,032
314	1	1	0	0	0	0,125	4	3,043
315	1	1	0	0	0	0,016	4	3,06
316	1	1	0	0	0	0,125	4	3,071

Tabela 2 – Método do Gradiente (10 últimas iterações)

2.5 Método Newton

O método foi implementado em python, e o código fonte está disponível no github: *https://github.com/joaojosemfigueiredo/EEL-5102-47-Metodos-Numericos-de-Otimizacao-I-*

Utilizando como parâmetros iniciais:

- $ponto\ inicial$: $[1\ 4]^T$
- $tolerância\ de\ 10^{-8}$
- $max_{iter} = 1000$
- $c_1 = 0.1$
- $c_2 = 0.9$
- $alpha_{init} = 1$

O método atingiu convergência em 8 iterações. Como podemos conferir na tabela 3.

Iteração	x_1	x_2	p_1	p_2	Norma do Gradiente	Alpha	f_x_k	Tempo Decorrido (s)
0	1,75	3,625	12	-6	13,416	0,062	4,879	0,021
1	1,902	3,555	2,438	-1,125	2,685	0,062	4,818	0,036
2	1,503	2,066	-0,8	-2,978	2,297	0,5	4,289	0,048
3	1,139	1,166	-0,363	-0,899	2,193	1	4,037	0,059
4	1,029	1,047	-0,11	-0,12	0,918	1	4,001	0,065
5	1,001	1,001	-0,028	-0,046	0,111	1	4	0,076
6	1	1	-0,001	-0,001	0,005	1	4	0,083
7	1	1	0	0	0	1	4	0,094

Tabela 3 – Método de Newton

2.6 Método Quase-Newton BFGS

O método foi implementado em python, e o código fonte está disponível no github: *https://github.com/joaojosemfigueiredo/EEL-5102-47-Metodos-Numericos-de-Otimizacao-I-*

Utilizando como parâmetros iniciais:

- $ponto\ inicial$: $[1\ 4]^T$
- $tolerância\ de\ 10^{-8}$
- $max_{iter} = 1000$
- $c_1 = 0.1$
- $c_2 = 0.9$
- $alpha_{init} = 1$

O método atingiu convergência em 11 iterações. Como podemos conferir na tabela 4.

Iteração	x_1	x_2	p_1	p_2	Norma do Gradiente	Alpha	f(x_k)	Tempo Decorrido (s)
0	1,75	3,625	12	-6	13,416	0,062	4,879	0,012
1	1,868	3,625	0,235	0	2,685	0,5	4,771	0,018
2	1,739	3,247	-0,129	-0,378	0,762	1	4,596	0,022
3	1,322	1,809	-0,833	-2,877	0,453	0,5	4,108	0,025
4	1,096	1,056	-0,454	-1,505	0,35	0,5	4,03	0,032
5	1,103	1,139	0,007	0,083	0,872	1	4,016	0,037
6	1,048	1,092	-0,055	-0,047	0,565	1	4,002	0,041
7	1,008	1,016	-0,04	-0,075	0,123	1	4	0,045
8	1	1	-0,008	-0,017	0,017	1	4	0,048
9	1	1	0	0	0,001	1	4	0,048
10	1	1	0	0	0	1	4	0,055

Tabela 4 – Método Quase-Newton BFGS

4. Classificação dos pontos

Pontos críticos reais encontrados pelos 3 algoritmos x_k = (1, 1)

Tendo a hessiana

$$\begin{bmatrix} 12x^2 - 4y + 2 & -4x \\ -4x & 2 \end{bmatrix}$$

Aplicando no ponto (1,1):

$$\begin{bmatrix} 10 & -4 \\ -4 & 2 \end{bmatrix}$$

Os autovalores encontrados são: (11.65685425, 0.34314575)

Como os autovalores são todos positivos a Matriz no ponto encontrado é Definida Positiva (DP).A análise de convexidade já foi realizada na seção 2.2, e concluímos que a função é convexa. Como a função é convexa e definida positiva, o ponto encontrado é de mínimo global.

5. Conclusão

Este trabalho teve como objetivo entender e implementar 3 diferentes métodos de otimização sem restrições, sendo eles o método do Gradiente, o método de Newton e o método de Quase-Newton BFGS. Os métodos foram implementados utilizando as condições fortes de Wolfe, e a interpolação quadrática para que pudesse achar o melhor passo (α_k) na direção de busca (p_k).

O método de Newton puro se mostrou o mais eficaz e convergiu em apenas 8 iterações, ele foi implementado de forma que em cada iteração testava se a hessiana no ponto em questão era Definida Positiva, se sim ele utilizava a direção de newton dada por p_k = -np.linalg.inv(hess) @ grad, se não ele utiliza a direção utilizada no método do gradiente, que é dado por p_k = -grad.

Apesar da convergência rápida o método de newton pode encontrar problemas de convergência se não conseguir resolver a inversa da hessiana. Portanto o método que traz a melhor solução é o de Quase-Newton BFGS, pois apesar de ter atingido a convergência em 11 iterações, seu tratamento utilizando a modificação da matriz hessiana durante o processo de solução do sistema linear força que a mesma seja "suficientemente" definida positiva e é mais eficaz para uma gama maior de funções se comparado ao método de Newton puro.

O método que obteve o pior desempenho foi o método do gradiente, atingindo a convergência em 317 iterações, em aproximadamente 3 segundos. Isso é devido a sua simplicidade na busca de uma direção de busca, que é definida apenas como o negativo do gradiente no ponto atual.