Trabalho 01

MEC 2403 - Otimização e Algoritmos para Engenharia Mecânica

Pedro Henrique Cardoso Paulo
pedrorjpaulo.phcp@gmail.com

Professor: Ivan Menezes

Departamento de Engenharia Mecânica
PUC-RJ Pontifícia Universidade Católica do Rio de Janeiro
maio de 2023

Trabalho 01

MEC 2403 - Otimização e Algoritmos para Engenharia Mecânica

Pedro Henrique Cardoso Paulo

maio de 2023

1 Introdução

1.1 Objetivos

Esse é o entregável da Trabalho 01 da disciplina MEC 2403 - Otimização e Algoritmos para Engenharia Mecânica. Esse trabalho tem como objetivos:

1. Aplicar os principais métodos de otimização sem restrição (OSR) implementados na Lista 02
2. Comparar os valores obtidos e número de passos para funções quadráticas e não quadráticas com o previsto pela literatura para cada método
3. Aplicar os otimizadores em problemas complexos e testar sua escalabilidade

Para atingir esse objetivo, este trabalho consistirá na realização de dois exercícios. O primeiro consistirá na aplicação dos métodos de otimização a duas funções distintas, sendo a primeira quadrática e a segunda não. O segundo exercício consistirá na solução de um problema de inspiração física, capaz de ser solucionado pelo método dos elementos finitos.

1.2 Links úteis

Nesta seção são listados alguns links e referências úteis para se entender o trabalho desempenhado.

1. Apostila de programação matemática da disciplina
2. GitHub usado para essa disciplina
3. Notebook com o código para as figuras desse relatório
4. Notebook com o código para derivação simbólica das funções mais complexas
5. Pasta com os códigos a serem aproveitados em todas as listas

2 Materiais e métodos

Nesta seção serão descritos os principais métodos de otimização aplicados neste estudo, tal como detalhes de sua implementação em código Python.

2.1 Otimização sem restrição (OSR)

Otimização sem restrição consiste na busca de um mínimo de uma função $f(\mathbf{x})$ a partir de um processo iterativo partindo de um ponto inicial $\mathbf{x_0}$. Tal como o nome sugere, nesta classe de problemas não há restrição quanto ao domínio da função f, sendo o mínimo final obtido uma função do ponto inicial e dos algoritmos aplicados para a solução do problema.

Via de regra, todo algoritmo de otimização sem restrição consiste em um *loop* dos seguintes passos, repetidos até que dada condição de parada seja atingida:

1. A partir de dada posição $\mathbf{x_i}$, determinar uma direção de busca $\mathbf{d_i}$ na qual o mínimo será procurado
2. Executar bsca linear para determinar o passo α_i que leva ao ponto de mínimo na direção $\mathbf{d_i}$

3. Atualizar o valor de posição $\mathbf{x_{i+1}} = \mathbf{x_i} + \alpha_i \mathbf{d_i}$

A condição de parada típica para esses problemas é a condição necessária de primeira ordem, i.e. $|\nabla f| < tol$. Esta será a condição aplicada em todos os exercícios deste trabalho e, embora de fácil implementação nos casos em que o gradiente é conhecido, apresenta como desvantagens não garantir o atingimento de um mínimo global (problema comum a todos os algoritmos de minimização) e de não garantir o atingimento de um mínimo (qualquer ponto crítico como máximos e pontos de sela atenderiam à condição).

2.2 Métodos de determinação de direção

Nesta seção serão descritos os principais métodos de determinação de direção implementados neste estudo. Ressalta-se que no escopo deste trabalho, todos eles serão usados em todos os itens.

2.2.1 Univariante

É o método mais simples de seleção de direção que consiste em selecionar alternadamente as direções da base canônica como como a direção de busca linear. tem a seu favor a simplicidade e a independência do conhecimento das derivadas da função, mas apresenta como contras sua tendência a baixa eficiência e o fato de não haver garantias de que a direção proposta tem sentido alinhado com um mínimo, exigindo checagem para valores de α negativos.

2.2.2 Powell

Método de otimização que, como o univariante, independe do conhecimento prévio de derivadas da função a ser otimizada. Sua maior característica é o uso de direções Q-conjugadas para uma tentativa de ganho de performance quando comparado ao univariante. Uma descrição simples do processo o método é apresentada a seguir:

1. Começa-se com um conjunto de direções candidatas composto pelas canônicas
2. É realizado um *loop* nas direções candidatas
3. Uma nova direção $\mathbf{d}$ é determinada pela diferença entre o ponto ao fim e ao início do *loop*
4. Uma otimização é realizada na direção $\mathbf{d}$. Essa direção é Q-conjugada.
5. Caso ainda hajam direções canônicas na lista de candidatas, a primeira direção da lista é descartada e $\mathbf{d}$ é adicionada à lista. Retorna-se ao passo 2
6. Caso contrário, retorna-se ao item 1

A grande vantagem do método de Powell é que a aplicação de direções Q-conjugadas garante a convergência para funções quadráticas no primeiro passo após termos apenas candidatas Q-conjugadas. Assim, para funções quadráticas, espera-se que o método convirja em $(n+1)^2$ passos, onde n é a dimensão do domínio da função.

2.2.3 Steepest Descent

Método que consiste em definir $\mathbf{d_i} = -\nabla f(\mathbf{x_i})$. A motivação para essa formulação é seguir a direção que localmente leva ao mais rápido decréscimo do valor da função, sendo esta função o oposto do gradiente.

2.2.4 Fletcher-Reeves

Consiste na minimização da função f por sua aproximação pro uma função quadrática. Esse método trabalha com direções Q-conjugadas após o primeiro passo por meio da atualização das direções por meio da expressão $\mathbf{d_{i+1}} = -\mathbf{g_{i+1}} + \beta_i \mathbf{d_i}$, com $\mathbf{g_{i+1}}$ sendo o gradiente da função f. Para o método de Fletcher-Reeves, o valor de β_i é calculado pela equação 1.

$$\beta_i = \frac{\mathbf{g_{i+1}^T g_{i+1}}}{\mathbf{g_i^T g_i}} \tag{1}$$

Por trabalhar com a direção conjugada a partir do primeiro passo, no caso de uma função quadrática, espera-se que o método convirja em $(n+1)$ passos, onde n é a dimensão do domínio da função.

2.2.5 Newton-Raphson

Método de segunda ordem que trabalha com direções conjugadas desde o primeiro passo, mas em contrapartida demanda conhecimento da Hessiana ($\mathbf{H}$) da função, i.e. suas derivadas segundas. A direção é dada pela expressão $\mathbf{d_i} = (\mathbf{H}(f)_{\mathbf{x}=\mathbf{x_i}})^{-1}\nabla f(\mathbf{x_i})$. Converge em n passos ou menos para funções quadráticas.

2.2.6 BFGS

Método de primeira ordem que tenta emular o comportamento de Nwton-Raphson sem a construção explícita de Hessiana. Para tal, é criada uma matriz $\mathbf{S}$ que é atualizada com base nos valores de gradiente e ponto ao longo da corrida. Tente a convergir em $(n+1)$ passos ou menos pra funções quadráticas.

2.3 Métodos de Busca Linear

Nesta seção serão descritos brevemente os métodos de busca linear para a determinação de mínimos em dada direção. Para os exercícios deste trabalho, apenas o método da seção áurea, com intervalo inicial determinado pelo passo constante serão aplicados.

2.3.1 Passo constante

Método mais simples de busca linear, consiste em ir incrementando o valor de α por um valor constante até chegarmos a um ponto em que a função comece a crescer. Muito usado para fins de determinar um intervalo inicial para a busca por métodos mais eficientes.

Importante ter em mente que este método supõe que a direção de busca tem um sentido que aponta para o mínimo, de modo que uma checagem de se isso é verdade é essencial caso o método de geração da direção não consiga garantir isso.

2.3.2 Bisseção

Método para busca do mínimo em um intervalo $[\alpha_{min}, \alpha_{max}]$ que consiste na divisão do intervalo sempre na metade e descarte de parte dele. É feito iterativamente até que se atinja um valor de $\alpha_{max} - \alpha_{min}$ pré-determinado.

2.3.3 Seção Áurea

Método para busca do mínimo em um intervalo $[\alpha_{min}, \alpha_{max}]$ que consiste na divisão do intervalo em três seções, com os valores intermediários dependendo da razão áurea. Uma das seções mais extremas é sempre descartada. É feito iterativamente até que se atinja um valor de $\alpha_{max} - \alpha_{min}$ pré-determinado, tendo o potencial de ser mais eficiente numericamente que a bisseção por conta de sua formulação.

2.4 Implementação em Python

A implementação dos métodos se deu por meio da criação, em linguagem Python, de três objetos principais responsáveis por uma parte do processo de otimização. A estrutura de classes final é resumida na Figura 1. Em termos de organização, o código ficou quebrado em 3 arquivos principais, a saber:

- `steps.py`: Arquivo contendo a classe `GenericStep` e suas subclasses. Seu código encapsula os métodos de busca linear que são responsáveis pela determinação do α e também a execução de checagens do sentido da direção proposta.
- `functions.py`: Arquivo contendo a classe `SpecialFunction` e suas subclasses. A função dessa classe é encapsular em um único objeto a função, sua hessiana e seu gradiente para conveniência dos métodos. Cabe comentar que neste módulo também há a subclasse `NumericalSpecialFunction` que tem por objetivo calcular numericamente gradientes e hessianas de funções usando o pacote `numdifftools`. Essa classe será usada para os testes de generalização apenas onde a derivação analítica for impossível ou muito custosa.
- `optimizers.py`: Arquivo que contem a classe `GenericOptimizer` e suas subclasses. É nesse arquivo que são definidos os procedimentos para iteração dos métodos, cálculo das direções e critérios de parada (tanto o gradiente quanto um numero máximo de iterações). Importante ressaltar que os objetos neste módulo esperam receber `SpecialFunction` e `GenericStep` para performar sua tarefa.

O uso desses classes para a execução das tarefas consiste em primeiramente, alocar os objetos de passo e otimizadores que gostaríamos de usar conforme exemplo abaixo:

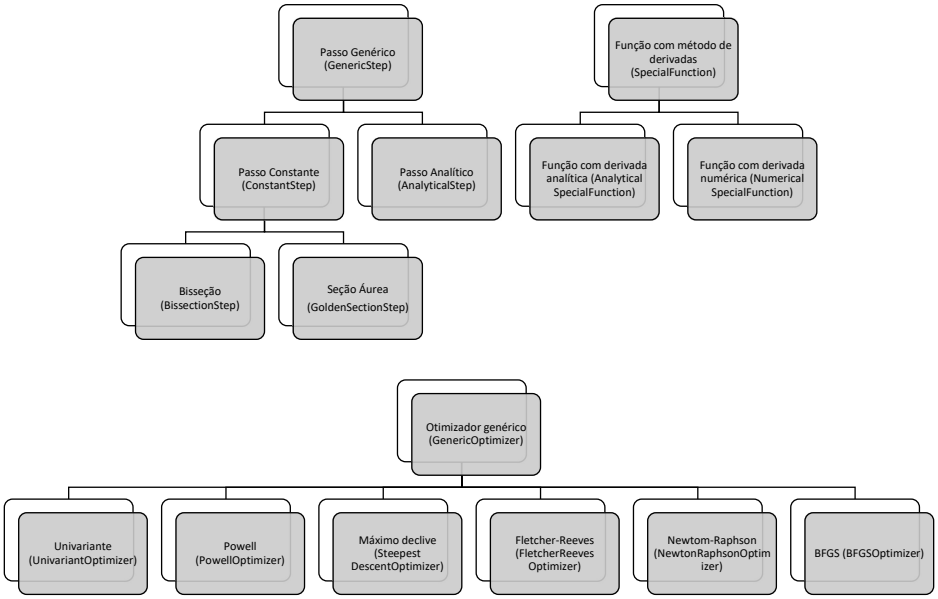


Figura 1: Estrutura de classes implementada e heranças

```
# Alocacao do Step
step = steps.GoldenSectionStep(da = 0.01, tol = 1e-8, check_direction=True, normalize=False)

#Alocacao dos optimizers
optimizrs = [
    ('Univariant', optimizers.UnivariantOptimizer(tol=1e-5)),
    ('Powell', optimizers.PowellOptimizer(tol=1e-5)),
    ('Steepest', optimizers.SteepestDescentOptimizer(tol=1e-5)),
    ('FletchRvs', optimizers.FletcherReevesOptimizer(tol=1e-5)),
    ('NewtnRaph', optimizers.NewtonRaphsonOptimizer(tol=1e-5)),
    ('BFGS', optimizers.BFGSOptimizer(tol=1e-5)),
]
```

Em seguida, a função de interesse é encapsulada em uma `AnalyticalSpecialFunction`, junto com a definição dos pontos iniciais e suas cores:

```
# Definir f, grad, e hess
def f(x1, x2):

return x1**2 - 3*x1*x2 + 4*x2**2 + x1 - x2

def gradf(x1, x2):

return np.array(
    [ 2*x1 - 3*x2 + 1,
    -3*x1 + 8*x2 - 1]
)

def hessf(x1, x2):

return np.array(
    [
    [ 2, -3],
    [-3,  8]
    ]
)

# Encapsular como SpecialFunction

f = functions.AnalyticalSpecialFunction(f, gradf, hessf)

# Definir pontos iniciais do processo
```

```
points = [
(np.array([2, 2]), 'r'),
(np.array([-1, -3]), 'b'),
]
```

Por fim, um *loop* é executado por todos os métodos de otimização para determinar os mínimos e gerar os gráficos:

```
item = 'a'

x = np.linspace(-6, 4, 100)
y = np.linspace(-6, 4, 100)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)

levels =[0, 1, 2, 5, 7, 10, 20, 30, 40, 50]

for name, optimizer in optimizrs:
    fig, ax = plt.subplots(1,1, figsize=(5, 5))
    ax.contour(X, Y, Z, levels,cmap='rainbow')
    for p_inicial, color in points:
        t_init = datetime.datetime.now()
        p_final = optimizer(f, p_inicial, step)
        t_final = datetime.datetime.now()
        print(f'{name}: p_final = [{p_final[0]:.6f}, {p_final[1]:.6f}, {f(*p_final):.6f}].T.
                                                    Execucao:{t_final - t_init}, {optimizer
                                                    .iter} iteracoes')
        ax.plot(*p_inicial, f'{color}o')
        ax.text(p_inicial[0]-0.25, p_inicial[1]+0.25, f'$P_{{inic}} = [{p_inicial[0]}, {
                                                    p_inicial[1]}]$', color=color)
        ax.plot(*p_final, 'ko')
        ax.text(p_final[0]-0.3, p_final[1]-0.5, f'$P_{{final}} = [{p_final[0]:.1f}, {p_final[1
                                                    ]:.1f}]$', color='k')
        cache = optimizer.get_history()
        for i in range(len(cache)-1):
            p_i = cache[i]
            p_f = cache[i+1]
            ax.quiver(p_i[0], p_i[1], p_f[0]-p_i[0], p_f[1]-p_i[1], color=color, angles='xy',
                                                        scale_units='xy', scale=1)#, label
                                                        ='Passo completo')
    ax.grid()
    #ax.legend()
    ax.set_xlabel('$x_1$')
    ax.set_ylabel('$x_2$')
    ax.set_title(f'${name}$')
    fig.savefig(f'images/q1{item}_{name}.pdf')
```

3 Questão 01

3.1 Enunciado

Implementar, usando o MATLAB ou Python, os métodos de otimização: (a) *Univariante*; (b) *Powell*; (c) *Steepest Descent*; (d) *Fletcher–Reeves*; (e) *BFGS*; e (f) *Newton–Raphson*. Adotar o método da Seção Áurea para a realização das buscas unidirecionais (line search). Para verificação da convergência numérica, utilizar uma tolerância de 10^{-5}. Em seguida, testar a sua implementação encontrando os pontos de mínimo das seguintes funções:

(a) $f(x_1, x_2) = x_1^2 - 3x_1x_2 + 4x_2^2 + x_1 - x_2$
Pontos iniciais: $\mathbf{x^0} = [2, 2]^T$ e $\mathbf{x^0} = [-1, -3]^T$,

(b) $f(x_1, x_2) = (1 + a - bx_1 - bx_2)^2 + (b + x_1 + ax_2 - bx_1x_2)^2,\ a = 10,\ b = 1$
Pontos iniciais: $\mathbf{x^0} = [10, 2]^T$ e $\mathbf{x^0} = [-2, -3]^T$,

3.2 Solução

3.2.1 Item a

Um resumo dos resultados obtidos para o estudo do presente item estão resumidos na Tabela 1, onde é possível ver que, salvo erros de arredondamento, todos os métodos convergiram para essencialmente o mesmo ponto depois do processo de otimização. Além disso, dada a função quadrática estudada neste exemplo, é possível

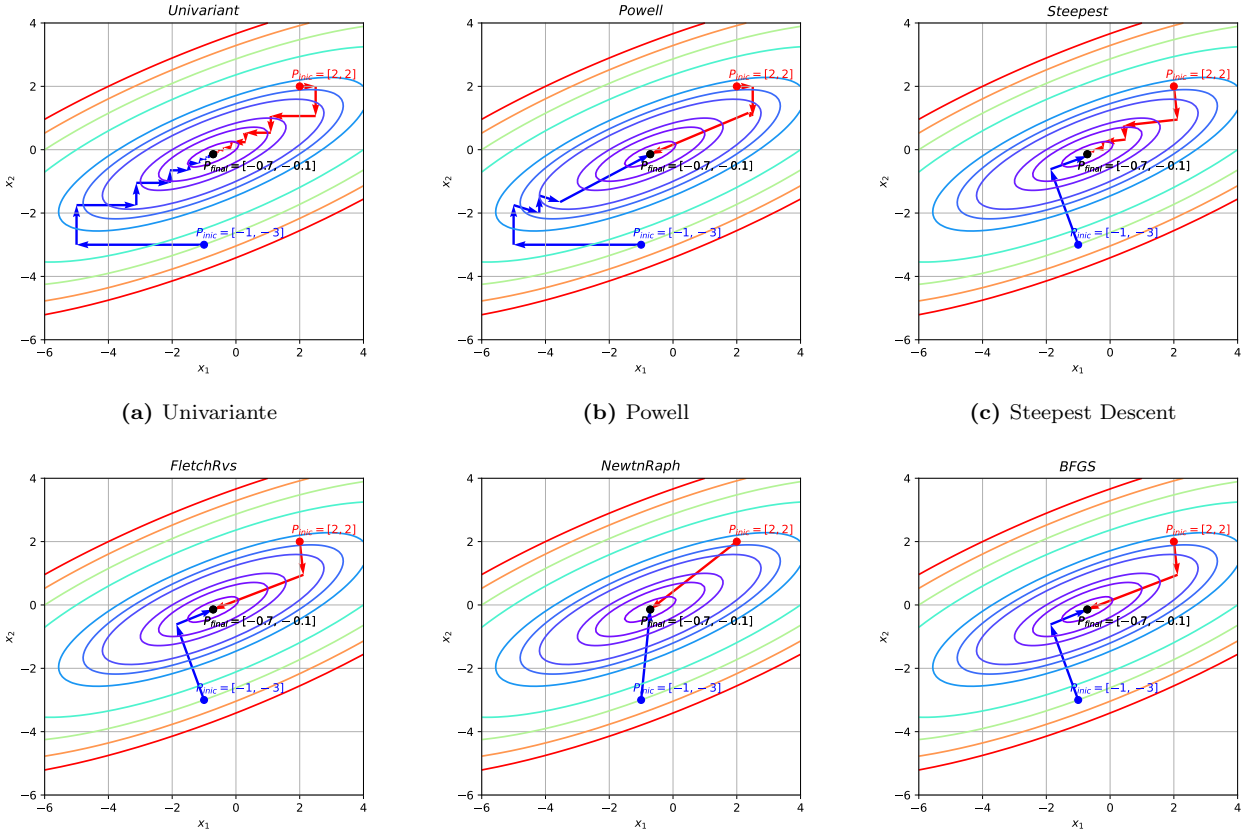
notar que nenhum dos métodos que fazem uso de direções conjugadas ultrapassou o valor de máximo número de iterações para uma função quadrática com domínio bidimensional (9 passos para o Powell, 3 para Fletcher-Reeves e BFGS e 2 para o Newton-Raphson).

	$\mathbf{x^0} = [2, 2]^T$			$\mathbf{x^0} = [-1, -3]^T$		
Método	**Ponto de mínimo**	$\mathbf{n}_{passos}$	**t (s)**	**Ponto de mínimo**	$\mathbf{n}_{passos}$	**t (s)**
Univariante	$[-0.714275, -0.142853]^T$	46	0.011	$[-0.714293, -0.142860]^T$	48	0.015
Powell	$[-0.714286, -0.142857]^T$	6	0.028	$[-0.714286, -0.142857]^T$	6	0.007
Steepest Descent	$[-0.714280, -0.142855]^T$	32	0.016	$[-0.714294, -0.142861]^T$	7	0.004
Fletcher-Reeves	$[-0.714286, -0.142857]^T$	2	0.001	$[-0.714286, -0.142858]^T$	2	0.001
Newton-Raphson	$[-0.714286, -0.142857]^T$	1	0.001	$[-0.714286, -0.142857]^T$	1	0.001
BFGS	$[-0.714286, -0.142857]^T$	2	0.001	$[-0.714286, -0.142857]^T$	2	0.001

Tabela 1: Resumo dos resultados obtidos para a função (a)

A Figura 2 apresenta as curvas de nível para a função a. Ao combinarmos ressas visualizações com os resultados da Tabela 1, é interessante comentarmos também a eficiência superior, para este caso, dos métodos que usam direções conjugadas quando comparados cm os que não usam, o que pode ser visto tanto no número de iterações quanto no tempo total de execução da otimização. É interessante também notar a dependência da eficiência do método de Steepest Descent do ponto inicial, com seu número de iterações variando de 32 a 7, o que é justificável pelo formato da função que favorece uma convergência oscilatória do gradiente para o mínimo.

Outro ponto digno de nota é o tempo consideravelmente maior demandado pelo método de Powell no primeiro ponto inicial quando comparado com outros métodos. De forma geral isso pareceu estar associado à direções conjugadas de módulo pequeno (importante ressaltar que nosso método de busca linear não normaliza essas direções por *default*), o que na etapa de busca linear demandou uma quantidade grande de passos constantes para chegarmos ao intervalo de mínimo.



(a) Univariante **(b)** Powell **(c)** Steepest Descent

(d) Fletcher-Reeves **(e)** Newton-Raphson **(f)** BFGS

Figura 2: Resultados gráficos para a função a

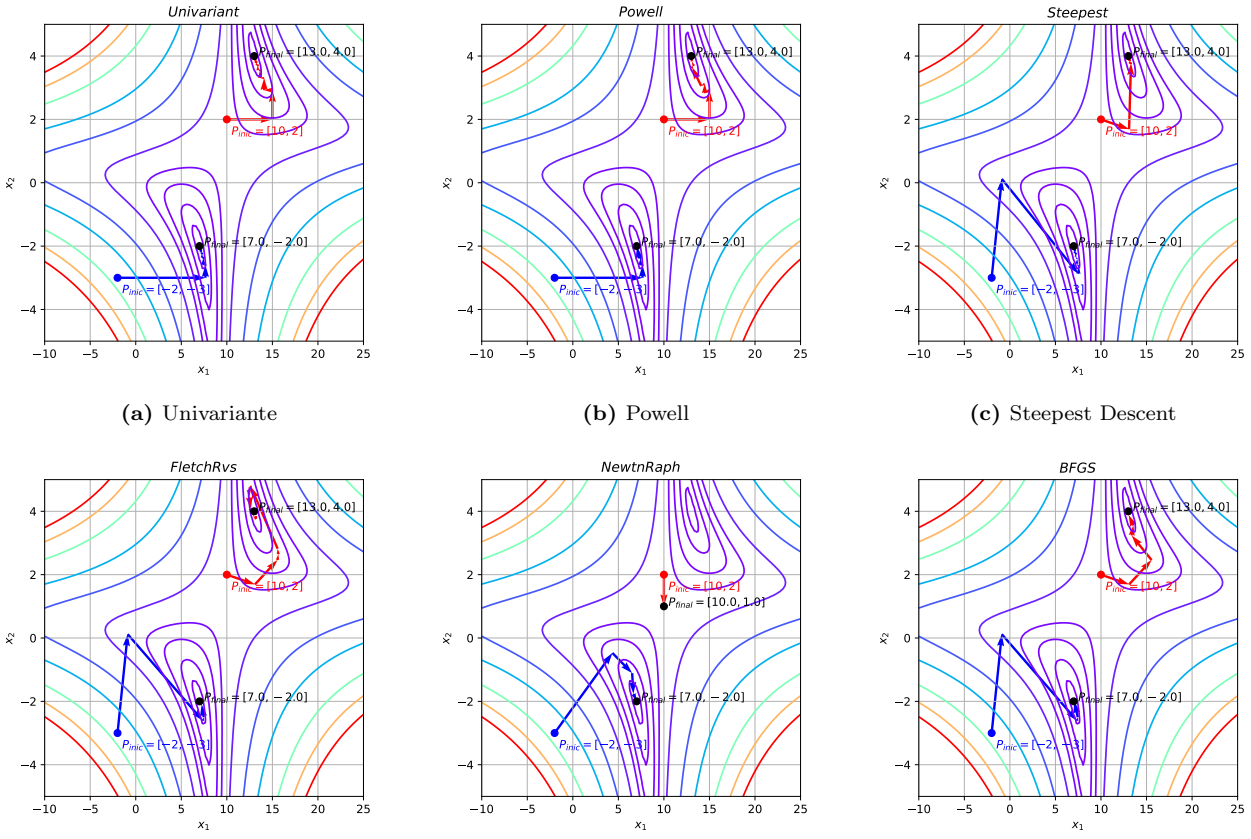
3.2.2 Item b

Um resumo dos resultados obtidos para o estudo do presente item estão resumidos na Tabela 2. Neste caso, é interessante notar que para os diferentes pontos iniciais, a otimização convergiu para valores de mínimos distintos, o que é esperado caso os pontos iniciais tenham mínimos locais distintos em suas vizinhanças. A Figura 3 apresenta visualização gráfica disso ao mostrar que ambos os pontos inicialis tem mínimos distintos em sua vizinhança, e como os métodos de OSR aqui estudados tem como critério de parada a chegada a um ponto crítico, é esperado que o mínimo mais próximo do ponto inicial seja a resposta do problema.

	$\mathbf{x^0} = [10, 2]^T$			$\mathbf{x^0} = [-2, -3]^T$		
Método	**Ponto de mínimo**	$\mathbf{n}_{passos}$	**t (s)**	**Ponto de mínimo**	$\mathbf{n}_{passos}$	**t (s)**
Univariante	$[13.000001, 3.999999]^T$	65	0.019	$[7.000001, -2.000001]^T$	61	0.018
Powell	$[13.000001, 4.000000]^T$	15	0.011	$[7.000000, -2.000000]^T$	18	0.032
Steepest Descent	$[13.000002, 3.999999]^T$	47	0.013	$[7.000002, -2.000002]^T$	42	0.009
Fletcher-Reeves	$[13.000001, 3.999999]^T$	64	0.013	$[7.000000, -2.000000]^T$	21	0.004
Newton-Raphson	$[10.000000, 1.000000]^T$	1	0.001	$[7.000000, -2.000000]^T$	6	0.005
BFGS	$[13.000000, 4.000000]^T$	9	0.008	$[7.000000, -2.000000]^T$	8	0.010

Tabela 2: Resumo dos resultados obtidos para a função (b)

Outro ponto relevante diz respeito ao valor final atingido pelo método de Newton-Raphson, que diverge dos demais métodos para o mesmo ponto inicial. Novamente, a combinação da análise da direção inicial do método de Newton com o critério de parada, que encerra o processo em pontos críticos, nos leva a entender que para esse método o ponto final atingido foi um ponto de sela que, embora não seja um mínimo, também satisfaz o critério de parada.



(a) Univariante **(b)** Powell **(c)** Steepest Descent

(d) Fletcher-Reeves **(e)** Newton-Raphson **(f)** BFGS

Figura 3: Resultados gráficos para a função b

4 Questão 02

4.1 Enunciado

Utilizando os métodos de otimização implementados na primeira questão:

(a) Determinar os deslocamentos (u_A, v_A), do ponto A, que minimizam a Energia Potencial Total Π do sistema de molas indicado na figura abaixo. Adotar o ponto inicial: $x_0 = [0.01, -0.10]^T$. A Figura 4 exemplifica esse cenário.

(b) desenvolver um estudo de convergência da solução deste problema (i.e., deslocamento do ponto A) para níveis crescentes de discretização do modelo (ou seja, considerando o número de molas $n = 2, 4, 6, ...$). Se possível, comparar as suas respostas com as soluções obtidas usando o Método dos Elementos Finitos (levando em consideração o comportamento não linear geométrico da estrutura). A rigidez de cada mola $(k_i = 1, ..., n)$ é obtida como a razão entre o módulo de rigidez axial do material e o seu comprimento. Os valores W_j (com $j = 1, ..., n$) correspondem às cargas nodais equivalentes aos pesos das molas.

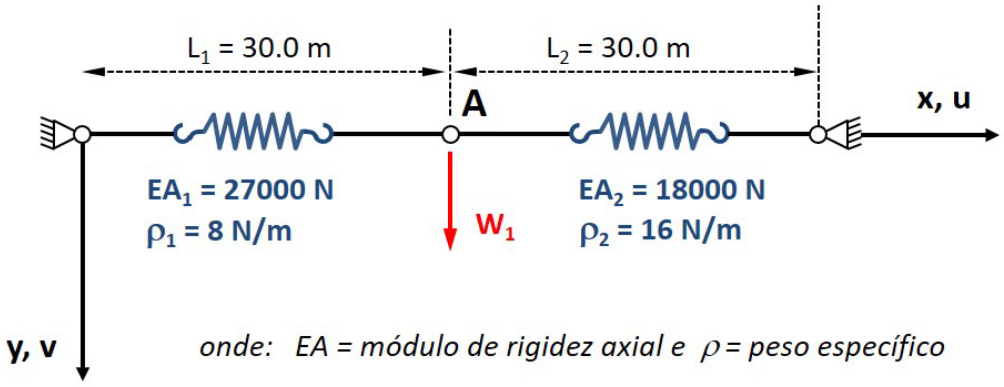


Figura 4: Esquemático do problema de duas molas

4.2 Solução

4.2.1 Determinação do mínimo

A determinação do deslocamento final do ponto de conexão entre as duas molas pode ser determinado por meio da minimização da energia potencial do sistema, dada pela equação 2 onde x_1 é o deslocamento horizontal positivo do ponto e x_2 é o deslocamento vertical negativo do mesmo:

$$\Pi = \frac{EA_1\left(-L_1 + \sqrt{x_2^2 + (L_1 + x_1)^2}\right)^2}{2L_1} + \frac{EA_2\left(-L_2 + \sqrt{x_2^2 + (L_2 - x_1)^2}\right)^2}{2L_2} - \frac{x_2}{2}\left(L_1\rho_1 + L_2\rho_2\right) \qquad (2)$$

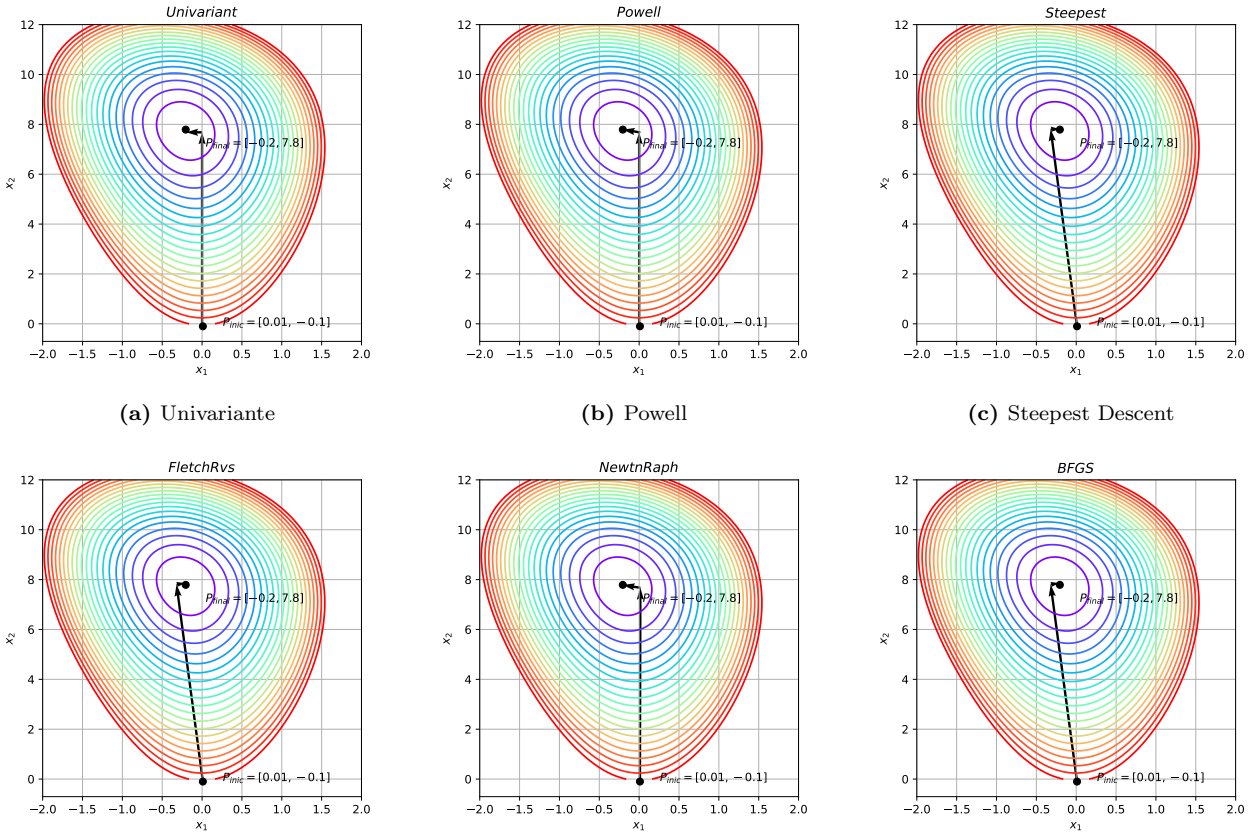
Os resultados obtidos pelo processo de minimização são resumidos na Tabela 3, onde novamente podemos ver que todos os métodos numéricos convergiram para o mesmo ponto final salvo erros numéricos. Se destaca, porém, o tempo que o método de Powell levaou para convergir (aproximadamente 1 min). Novamente isso parece estar relacionado com a determinação de direções conjugadas de módulo pequeno, o que prejudica a performance da busca linear pelo mínimo.

	$\mathbf{x^0} = [0.01, -0.10]^T$		
Método	**Ponto de mínimo**	$\mathbf{n}_{passos}$	**t (s)**
Univariante	$[-0.205109, 7.788993]^T$	69	0.028
Powell	$[-0.205109, 7.788993]^T$	28	58.743
Steepest Descent	$[-0.205109, 7.788993]^T$	8	0.002
Fletcher-Reeves	$[-0.205109, 7.788993]^T$	70	0.017
Newton-Raphson	$[-0.205109, 7.788993]^T$	200	0.063
BFGS	$[-0.205109, 7.788993]^T$	200	0.055

Tabela 3: Resumo dos resultados obtidos para a discretização em 2 molas

Outro ponto importante é a análise do número de iterações dos métodos de Newton-Raphson e BFGS. Como 200 iterações foi o valor máximo passado para os métodos, isso indica que ambos não convergiram para este problema. Comparando resultados do ponto final, fica evidente que eles atingiram valores próximos dos demais métodos, com a não convergência sendo apenas uma questão de o gradiente ter se mantido com módulo acima da tolerância. Espera-se que, caso a tolerência seja relaxada (de 10^{-5} para 10^{-4}, por exemplo), esses métodos venham a convergir.

A Figura 5 mostra as curvas de nível para o problema estudado.



(a) Univariante **(b)** Powell **(c)** Steepest Descent

(d) Fletcher-Reeves **(e)** Newton-Raphson **(f)** BFGS

Figura 5: Resultados gráficos da determinação do mínimo da função potencial

4.2.2 Análise de convergência

O estudo de análise de convergência tem por objetivo analisar a escalabilidade dos métodos implementados para funções de mais de duas direções e também avaliar o impacto dessa alta dimensionalidade na performance dos métodos de otimização. Para tal, uma formulação mais genérica do problema precisou ser elaborada.

Suponha uma mola individual i, caracterizada pelas propriedades EA_i, dL_i, ρ_i e com extremos nos pontos $[x_{1,i}, x_{2,i}]^T$ e $[x_{1,i+1}, x_{2,i+1}]^T$. A equação 3 representa a energia potencial dessa mola:

$$\Pi_i = \frac{EA_i \left(-dL_i + \sqrt{(x_{2,i+1} - x_{2,i})^2 + (dL_i + x_{1,i+1} - x_{1,i})^2} \right)^2}{2dL_i} - \frac{x_{2,i} + x_{2,i+1}}{2} (dL_i \rho_i) \tag{3}$$

Caso queiramos calcular a energia potencial de n molas com $n+1$ nós (incluindo extremos) que componham a totalidade da nossa mola discretizada, a energia potencial total será dada pela expressão 4:

$$\Pi = \sum_{i-1}^{n} \Pi_i \tag{4}$$

Para a realização desse estudo e dada a complexidade crescente do problema, algumas alterações nos parâmetros de otimização foram feitas:

- Número máximo de iterações foi para 500

- Tolerância da norma do gradiente aumentada para 10^{-3}
- Normalização da direção na busca linear
- Passo constante inicial $\Delta\alpha = 0.001$
- Ponto inicial padronizado em $\mathbf{x_0} = \mathbf{0}$

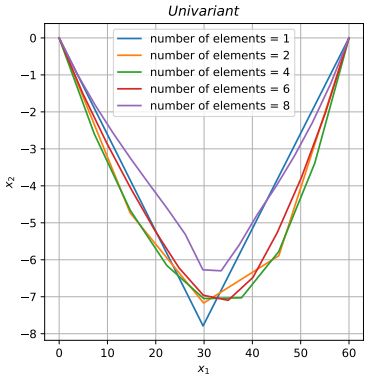
Além disso, devido à complexidade de se obter gradiente e Hessiana analíticos para o problema, a interface `NumericalSpecialFunction`, com derivação numérica, foi usada para este problema. A convergência, por sua vez, foi estudada para a discretização de cada mola do par que compõe o problema em 1, 2, 4, 6 e 8 molas, garantindo sempre a simetria de nós com relação ao centro.

Os resultados obtidos para a análise de convergência podem ser vistos na Tabela 4, com casos em que não houve convergência em 500 passos sinalizados em vermelho. Em termos de performance é possível ver que, para números de iterações relativamente grandes, o aumento da dimensionalidade do problema está intrinsecamente ligado ao aumento do tempo computacional gasto pelo método. Outro fator interessante de se notar vem da comparação do método de Newton-Raphson com BFGS, em que fica claro que o segundo, mesmo com mais iterações que o primeiro, demandou um tempo computacional menor ou igual em praticamente todos os casos, o que provavelmente está associado ao alto custo de se estimar numericamente a Hessiana de problemas de grande dimensionalidade.

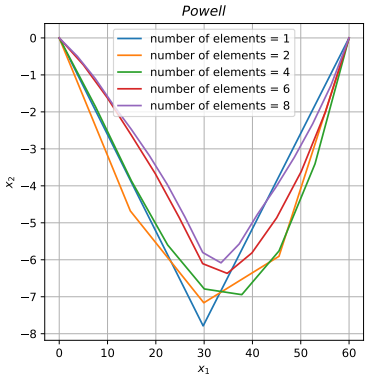
	1 mola		2 molas		4 molas		6 molas		8 molas	
Método	$\mathbf{n}_{passos}$	**t (s)**	$\mathbf{n}_{passos}$	**t (s)**	$\mathbf{n}_{passos}$	**t (s)**	$\mathbf{n}_{passos}$	**t (s)**	$\mathbf{n}_{passos}$	**t (s)**
Univariante	9	0.287	152	1.709	500	9.800	500	18.540	500	29.059
Powell	13	0.282	500	3.661	500	9.739	500	17.908	500	28.633
Steepest	8	0.282	454	3.089	500	8.708	500	15.931	500	24.517
Fletcher	9	0.274	58	0.902	178	4.045	301	10.859	419	22.782
Newton	3	0.271	6	0.788	7	2.489	8	7.538	9	15.537
BFGS	4	0.270	13	0.681	26	1.766	38	3.298	47	5.384

Tabela 4: Resumo dos resultados obtidos para a discretização em n molas

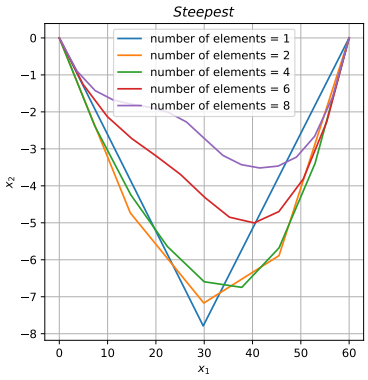
Os resultados finais podem ser vistos na Figura 6. É importante notar que os métodos em vermelho não convergiram em todos os casos, apresentando geometrias finais estranhas. A princípio isso não indica uma impossibilidade de se usar estes métodos para o probblema estudado, mas indica apenas que eles demandariam um número maior de iterações e mais custo computacional para atingir a mesma performance dos métodos que convergiram.



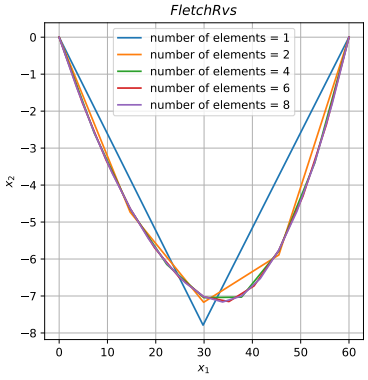
(a) Univariante



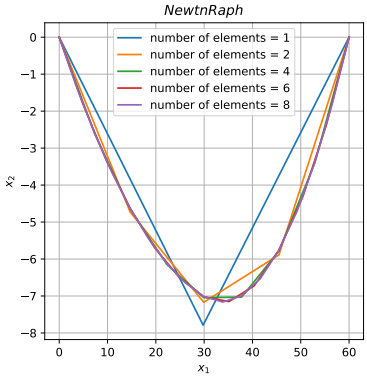
(b) Powell



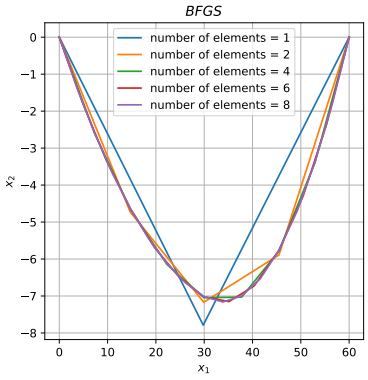
(c) Steepest Descent



(d) Fletcher-Reeves



(e) Newton-Raphson



(f) BFGS

Figura 6: Resultados da análise de convergência para discretizações maiores da mola