Questão 2

Introdução

Neste trabalho, resolvemos o problema para **três unidades geradoras** com funções de custo quadráticas e considerando **perdas de potência** modeladas por uma matriz de coeficientes BBB. Foram utilizados dois métodos de otimização restrita estudados na disciplina: **Penalidade Exterior** e **Lagrangeano Aumentado**.

A implementação foi feita na linguagem **Python**, utilizando a **biblioteca otimo.py** fornecida pelo professor, que contém implementações dos métodos de otimização vistos em sala de aula.

Queremos resolver o seguinte problema de otimização:

**Função Objetivo (custo total):**

Com :

Restrição de Igualdade (demanda + perdas):

Perdas de Potência (PL):

com matriz B dada por:

Limites de Geração:

Implementação

Nesta tarefa foram usadas 2 Bibliotecas a Numpy e ótimo.py

- Função objetivo:

def custo\_total(P):

P1, P2, P3 = P

return (

0.15 \* P1\*\*2 + 38 \* P1 + 756 +

0.1 \* P2\*\*2 + 46 \* P2 + 451 +

0.25 \* P3\*\*2 + 40 \* P3 + 1049

)

- Matriz de Perdas e Demandas

B = np.array([

[0.000049, 0.000014, 0.000015],

[0.000014, 0.000045, 0.000016],

[0.000015, 0.000016, 0.000039]

])

D = 850

- Perda de potencia:

def perda\_potencia(P):

return P @ B @ P

- Restrições:

def restricao\_demanda(P): return np.sum(P) - D - perda\_potencia(P)

def r1(P): return P[0] - 150

def r2(P): return P[1] - 100

def r3(P): return P[2] - 50

def r4(P): return 600 - P[0]

def r5(P): return 400 - P[1]

def r6(P): return 200 - P[2]

- Agrupamento das restrições e especificações de cada método, está etapa é necessária para utilizar a função da biblioteca ótimo

restricoes = [restricao\_demanda, r1, r2, r3, r4, r5, r6]

tipos\_penalidade = np.array(['>', '>', '>', '>', '>', '>', '>'])

tipos\_lagrangeano = np.array(['=', '>', '>', '>', '>', '>', '>'])

Parâmetros de inicialização:

x0 = np.array([300.0, 300.0, 150.0])

busca\_1d = SecaoAurea(precisao=1e-6)

irrestrito = GradienteConjugado(busca\_1d, precisao=1e-6)

Metodos de otimização

- Penalidade exterior

def resolver\_penalidade\_exterior():

metodo = PenalidadeExterior(precisao=1e-6)

sol = metodo.resolva(

custo\_total, x0, restricoes, tipos\_penalidade,

irrestrito, penalidade=1.0, aceleracao=2.0

)

return sol

- Lagrangeano aumentado:

def resolver\_lagrangeano\_aumentado():

metodo = LagrangeanoAumentado(precisao=1e-6)

sol = metodo.resolva(

custo\_total, x0, restricoes, tipos\_lagrangeano,

irrestrito, penalidade=1.0, aceleracao=2.0

)

return sol

Execução e resultados  
print("=== Método 1: Penalidade Exterior ===")

res1 = resolver\_penalidade\_exterior()

P1 = res1.x

print("P =", P1)

print("Custo total =", custo\_total(P1))

print("Perda =", perda\_potencia(P1))

print("Soma das potências =", np.sum(P1))

print("Demanda + perda =", D + perda\_potencia(P1))

print("\n=== Método 2: Lagrangeano Aumentado ===")

res2 = resolver\_lagrangeano\_aumentado()

P2 = res2.x

print("P =", P2)

print("Custo total =", custo\_total(P2))

print("Perda =", perda\_potencia(P2))

print("Soma das potências =", np.sum(P2))

print("Demanda + perda =", D + perda\_potencia(P2))

Os resultados Foram:

=== Método 1: Penalidade Exterior ===

P = [294.89018536 399.26406173 175.57573583]

Custo total = 75542.92538094848

Perda = 19.730023098416886

Soma das potências = 869.7299829141984

Demanda + perda = 869.7300230984168

=== Método 2: Lagrangeano Aumentado ===

P = [294.87820089 399.29121844 175.56088855]

Custo total = 75542.93033791147

Perda = 19.730307958824337

Soma das potências = 869.7303078830519

Conclusão

O problema de despacho econômico foi resolvido por meio dos métodos da **Penalidade Exterior** e do **Lagrangeano Aumentado**, ambos implementados com a biblioteca otimo.py. As soluções obtidas por ambos os métodos apresentaram resultados semelhantes em termos de custo total, alocação de potência entre as unidades geradoras e atendimento à demanda acrescida das perdas de transmissão.