

AULA 03 - PL ÚNICO

Objetivo é apresentar uma formulação para a resolução de um problema de despacho hídrotérmico determinístico, ou seja, quando se pressupõe que as vazões às UHEs são conhecidas para todos os estágios ou períodos de planejamento. A resolução será obtida através de um Problema de Programação Linear Único, ou seja, complementando todos os estágios.

FUNÇÃO OBJETIVO
<inserir fórmula>

CÁLCULO GENERALISTA NÚMERO DE VARIÁVEIS DE DECISÃO

$$N_{est} \cdot N_{UHE} \cdot 3 + N_{est} \cdot N_{UTE} + N_{est}$$

DIMENSÕES DO PROBLEMA(RESTRIÇÕES)

- Uma restrição de Igualdade de Balança Hídrico = $N_{est} \cdot N_{UHE}$
- Número de restrições de igualdade e AD = $N_{est} \cdot N_{UHE} + N_{est}$
- Uma restrição para cada estágio i e para cada UHE j

Se estágio (i=1): $V_{F1,j} = VI_j + AFL_{1,j} - V_{Ti,j} - V_{Vi,j}$

Se estágio (i≠1): $V_{F1,j} = V_{F1-1,j} + AFL_{1,j} - V_{Ti,j} - V_{Vi,j}$

- Uma restrição para cada estágio i: <inserir fórmula 14:02>
- Restrições de canalização Para cada UHEj: <inserir as condições>

Para cada UTEj: $0 \geq gt_{i,j} \geq GT_{MAXj}$

$$0 \geq def_i \geq \infty$$

DEPACHO HIDROTÉRMICO DETERMINÍSTICO

- DADOS DE ENTRADA: Usinas hidrelétricas, usinas térmicas e sistema -DADOS DE SAÍDA: Volume final, turbinado e vertido para cada usina hidrelétrica e cada estágio de planejamento CMA(associado à equação de balanço hídrico), Geração térmica, déficit de energia, CMO, custo total de operação para cada estágio de planejamento, Gráfico temporais dos problemas de otimização

PROBLEMA DE INSTÂNCIA MÍNIMA

$Min C_1 \cdot GT_{1,1} + C_2 \cdot GT_{2,1} + C_{DEF} \cdot GT_{1,1}$ BALANÇO HÍDRICO : AD: <inserir equações>
RESTRIÇÕES DE CANALIZAÇÕES:

<inserir condições>

DADOS DO SISTEMA

In []: *#Inserir os dados*

FUNÇÃO OBJETIVO

<inserir a fórmula 43:56>

```
In [1]: #Python na aula 03
        #LaTeX com a sintaxe das fórmulas

        from cvxopt.modeling import variable, solvers
        from cvxopt.modeling import op

        def pl_unico(sistema, cenario, imprime):
            Num_UHE = len(sistema["UHE"])

            #Variáveis de Decisão Organizadas Matricialmente
            #Exmplo de acesso á variável de decisão de volume final
            #CRIAÇÃO DE MATRIZES
            #No PYTHON a numeração das linhas e colunas iniciam no zzero

            vf = []
            #Com esse for percorremos por todas as usinas,o comando enumerate proporciona
            #O comando variable é uma lista com tantas posições forem os estágios
            #Nessa lista cada coluna será o volume final de determinada usina é esse valor
            #colunas = estágios e linhas = volume final de cada uhes
            for i,usin in enumerate(sistema["UHE"]):
                vf.append(variable(sistema["DGer"] ["Nest"], "Volume Final na Usina "+usin["Nest"]))

            #FUNÇÃO OBJETIVO
            # Função de custo futuro
            #fob inicia em 0 e vamos incrementando as parcelas dentro de um for alinhado
            #estag varia de 0, 1 e 2, porque são 3 estágios. declarados em dados do sistema

            fob = 0
            for estag in range(sistema["DGer"] ["Nest"]):
                #esse for externo é realacionado aos estágios que fez o range que vai de 0
                #Esse loop alinhando percorre todas as usinas térmicas e a cada estagio var

                for i, usin in enumerate(sistema["UTE"]):
                    #A matriz com a variáveis de decisão térmica é multiplicada pelo custo de
                    #A variável de controle usin assume os dados de todas as usinas térmicas
                    #Assim a fob vai sendo completada pelo(+=) com todas as parcelas custos
                    fob += usin["custo"]*gt[i][estag] #Verif a variável custo(renomear)

                #Para cada estágio, temos que continuar incremento o fob COM O CUSTO DE L
                #Essa linha inclui as parcelas de deficit
                fob += sistema["DGer"] ["CDef"]*deficit[estag]

                #Por fim, incluir as parcelas de volume vertido
                #O enumerate percorrer a lista de todas as usinas e a cada passagem do for
                for i, usin in enumerate(sistema["UHE"]):
                    fob += 0.01*vv[i][estag]
            #Finalizando a FUNÇÃO OBJETIVO
```

DEFINIÇÃO DAS RESTRIÇÕES

Os blocos de códigos das restrições devem está inserido no bloco acima

BALANÇO HÍDRICO

- Uma restrição cada estágio i e para cada UHE j

Se estágio ($i=1$): $V_{F1,j} = VI_j + AFL_{1,j} - V_{Ti,j} - V_{Vi,j}$

Se estágio ($i \neq 1$): $V_{F1,j} = V_{F1-1,j} + AFL_{1,j} - V_{Ti,j} - V_{Vi,j}$

```
In [ ]: #RESTRICÕES
#Cria-se uma lista vazia e inclui-se cada restrição
restricoes = []

#BALANÇO HÍDRICO

#for para percorrer o número de UHEs, por causa do enumerate:
#cada passagem do laço a variável usin vai assumir os dados dessa usina e o i assume
for i, usin in enumerate(sistema["UHE"]):
    #Nesse laço, percorremos todos os estágios com o comando range
    for estag in range(sistema["DGer"] ["Nest"]):
        if estag == 0:
            #Se estivermos no primeiro estágio, utilizamos o Volume Inicial
            restricoes.append(vf[i][estag] == float(usin["VI"]) + float(usin["AFL"])
        else:
            #Caso contrário usamos o volume final no mês anterior
            restricoes.append(vf[i][estag] == vf[i][estag-1] + float(usin["AFL"])[estag])
```

ATENDIMENTO À DEMANDA

Para saber qual a posição CMA de qualquer UHE

Através de $= Nest \cdot NUHE + Nest$

```
In [ ]: #Averar uma equação de AD para cada estágio
#Então no loop, a variável de controle estag variando de 0 até Nest=3
#Quando estamos programando é importante saber criar, entender e acessar os vetores
for estag in range(sistema["DGer"] ["Nest"]):
    AD = 0
    #Inclusão das parcelas passo a passo
    #Sobre as UHEs
    for i, usin in enumerate(sistema["UHE"]):
        AD += usin["Prod"] * vt[i][estag]
    #Sobre as UTEs
    #Durante o for, i assume os valores 0 e 1 e usin = as informações de cada uma
    for i, usin in enumerate(sistema["UTE"]):
        #Aqui incrementa-se os valores obtidos anteriormente
        AD += gt[i][estag]
    AD += deficit[estag]
    #Com as restrição pronta, adiciona à lista de restrições
    restricoes.append(AD == sistema["DGer"] ["Carga"][estag]) #consultar Carga
```

CANALIZAÇÕES

@ Colocar as condições de canalização aqui

```
In [ ]: #CTRL C + CTRL V do código aula 02 = comentário com @

#Laço externo percorre todos os estágios
for estag in range(sistema["DGer"]["Nest"]):
    for i, usin in range(sistema["UHE"]):
        #Acrescentando as restrições à lista vazia
        #Condições
        #Não precisa colocar limite superior para o volume vertido
        #O PPL evita colocar recurso nessa variável pq ela está penalizada na função
        #(?) essa penalização: fob += 0.01*vv[i][estag]]
        restricoes.append(vv[i][estag]>=0)

        #Térmicas
        for i, usin in range(sistema["UHE"]):

        #Deficit
        restricoes.append(deficit[estag]>= 0)

#Todas essas informações tem que ser inseridas por usinas e por estágio
#Fim Restrições de canalização
```

PROBLEMA DE OTIMIZAÇÃO LINEAR

Os blocos de códigos de otimização devem ser inseridos no bloco de código da FUNÇÃO OBJETIVO

```
In [ ]: problema = op(fob, restricoes)
#Dense e GLPK pertencem a solver da biblioteca CVXOPT
problema.solve("dense", "glpk")

lista_uhe = []
#Primeiramente, Laço para percorrer todas as usinas elétricas
for i, usin in enumerate(sistema["UHE"]):
    #Pula = acesso correto para cada um dos CMAs
    pula = i*sistema["DGer"]["Nest"]
    #As listas vazias são para acrescentar os resultados dos problemas de otimização
    cma = []
    volf = []
    volt = []
    volv = []

    #Resultados da fob, multiplicador de Lagrange, etc
    for estag in range(sistema['DGer']['Nest']):
        #Para extrair o valor encontrado utiliza .value
        #Para o multiplicador de Lagrange extrairse através do .multiplier
        cma.append(restricoes[pula+estag].multiplier.value[0])
        volf.append(vf[i][estag].value()[0])
        volt.append(vt[i][estag].value()[0])
        volv.append(vv[i][estag].value()[0])
        #No final desse loop teremos uma lista com os valores de CMA e volume de t

    #Dicionário de dados com os resultados de todos os estágios
    elemento = {
        "vf": volf,
        "vt": volt,
        "vv": volv,
        "cma": cma
    }
    #E por fim, adiciona à lista de usinas elétricas
    lista_uhe.append(elemento)
```

```

#Fim do loop, a lista_uhe irá conter uma lista de usinas e em cada elemento dela
#teremos os detalhes sobre as variáveis de decisão em cada um dos estágios

#PARA AS UTEs
lista_ute = []
for i, usin in enumerate(sistema["UTE"]):
    #Para cada usina térmica criamos uma lista para guardar os valores a geração
    gtermica = []
    for estag in range(sistema["DGer"]["Nest"]):
        gterter.append(gt[i][estag].value()[0])
    #Adiciona ao dicionário os valores das variáveis de decisão
    elemento = {
        "gt": gtermica
    }
    #Adiciona na lista de usinas térmicas
    lista_ute.append(elemento)

#ARMAZENAR OS DADOS DA OTIMIZAÇÃO RELATIVOS AO SISTEMA = Custo total e CMO
#Pular todas as restrições de balanço hídrico e ir para as restrições de AD
pula = Num_UHE*sistema["DGer"]["Nest"]
#Inserir os CMOs de cada estágio
cmo = []
#Para percorrer os estágios
for estag in range (sistema["DGer"]["Nest"]):
    #A cada passagem do laço, acrescenta um dados e insere o multiplicador de
    #Então pula os restrições de BH e soma a variável de controle estag
    cmo.append(restricoes[pula+estag].multiplier.value[0])

#Dicionário para os dados de otimização
Dger = {
    "Custo Total": fob.value()[0], #Acessando diretamente o fob
    "CMO": cmo #Lista do loop acima
}

#Compactar todas as variáveis de decisão, multiplicadores e variáveis duais da
#Em um único dicionário para os problemas de otimização
resultado = {
    "DGer": Dger, #Recebe o dicionário
    "UHE": lista_uhe, #Recebe a lista
    "UTE": lista_ute #Recebe a lista
}

```

IMPRIMIR OS RESULTADOS:

- Dos problemas de otimização
- Variáveis de decisão

```

In [ ]: #Parâmetro de entrada imprime
#Se for TRUE imprime os resultados dos problemas de otimização
#Se for FALSE, não imprime
if imprime:
    print("Custo de Operação de todos os estágios: ", fob.value())

    print("Volume Final por UHE em cada Estágio em (hm³)")
    for i, usin in enumerate(sistema["UHE"]):
        print(vf[i])
        print(vt[i])
        print(vv[i])
    print("Geração por UTE em cada Estágio em (MWmed)")
    for i, usin in enumerate(sistema["UTE"]):

```

```

        print(gt[i])

    print("Déficit de Energia em cada Estágio em (MWmed)")
    print(deficit)

#Para finalizar, retornar o resultado, dicionário que compacta todos os resultados
    return(resultado)
#FIM

```

PROGRAMA PRINCIPAL

UTILIZA A FUNÇÃO DO PL UNICO(PPL) E PLOTA GRÁFICOS

```

In [ ]: #Parâmetros (sistema, cenário, não quero imprimir os resultados das variáveis de de
#Alterar manualmente para True**
resultado = pl_unico(sistema, 1, imprime=False)
print(resultado)

```

- GRÁFICO DAS USINAS HIDRELÉTRICA

```

In [ ]: #Interessante imprimir gráficos para demonstrar a evolução do CMO ou do volume final
import numpy as np
import matplotlib.pyplot as plt
#Define um vetor de zero a 2. em 3 estágios
x = np.arange(0, sistema["DGer"]["Nest"],1)#1=sistema...-1
plt.figure(figsize=(10,8))

#Para mais usinas foi usado o comando for
for i, usin in enumerate(resultado["UHE"]):
    plt.plot(x+1, usin["vf"], marker="d")#x=estágios e d=diamond

plt.title("Evolução do Volume Armazenado Final (hm³)")
plt.xlabel("Período/mês de Planejamento")
plt.ylabel("Volume armazenado (hm³)")
plt.xticks(x+1) #Comando para colocar valores no eixo x e o "+1" para corrigir
plt.grid() #Comando para adicionar grades no gráfico
plt.tight_layout() #Para organizar a figura no size(10,8)
plt.show() #Para mostrar o gráfico

```

- GRÁFICO DA GERAÇÃO DAS UTEs

```

In [ ]: x = np.arange(0, sistema["DGer"]["Nest"],1)
plt.figure(figsize=(10,8))

for i, usin in enumerate(resultado["UTE"]):
    plt.plot(x+1, usin["gt"], marker="d")#x=estágios e d=diamond

#Como coloca legenda?
#O que mais adicionar nesse gráfico?
plt.title("Evolução do Volume Armazenado Final (hm³)")
plt.xlabel("Período/mês de Planejamento")
plt.ylabel("Volume armazenado (hm³)")
plt.xticks(x+1)
plt.grid()
plt.tight_layout()
plt.show()

```

Extras <https://drive.google.com/open?id=lejU38KljKiewONJVWHV3wcJ0kZoFumfu> (verificar)