AULA 05 - PROGRAMAÇÃO DINÂMICA DUAL DETERMINÍSTICA - PDDD

para a resolução de um problema de despacho hidrotérmico determinístico, caso no qual as afluências às usinas hidrelétricas (UHEs) são consideradas conhecidas em todo o horizonte de planejamento

DRIVE:
https://drive.google.com/a/engenharia.ufjf.br/file/d/1LyDvvyCypdqWidKw2dchem_brwyT0NZh/vi usp=sharing

CUSTO FUTURO

$FOB = Min \quad C_{inicial} + \alpha$

$Z_{sup} = \sum CI$

$\alpha = 0$ por causa do fim de mundo

- GASTO - ESTIMATIVA SUPERIOR E INFERIOR

Zsup equivale a estimativa superior, no caso, o somatório dos custos imediatos de todos os estágios. o ideal é que Zsup e Zinf sejam iguais, obtido apenas na 4ª iteração(slide)**59:30**. Demonstração de um gráfico custo pelo volume inicial e traçar as deviradas(taxa de variação) da curva da função que equivale ao α(equação da reta) $r : \alpha = a \cdot v + b$ e o coeficiente angular (termo "a") equivale ao Custo Marginal da Água = CMA.

- FUNÇÃO DE CUSTO FUTURO (FCF)

FLUXOGRAMA DA PDDD imagem 1:04:21

- **FORWARD**

Listas de Funções de Custo Futuro $\Rightarrow$ estag = 0 $\Rightarrow$ Volume inicial das UHEs $\Rightarrow$ Variáveis de controle de convergência = Zsup e Zinf = 0 $\Rightarrow$ Resolver o primeiro problema de programação linear - PPL do estag=0 $\Rightarrow$ É um problema de estágio 0? $\Rightarrow$ SIM $\Rightarrow$ Zinf = FOB $\Rightarrow$ SE NÃO OU DEPOIS DO QUESTIONAMENTO ANTERIOR $\Rightarrow$ Zsup += Custo imediato $\Rightarrow$ Volume inicial = Volume final de cada usina $\Rightarrow$ estag ++ (adiciona estágios) $\Rightarrow$ Se ainda houver estágio retornamos para PPL, se não paramos.

- **BACKWARD** Se Zsup e Zinf não convergir, inicia-se o processo de backward. Se convergir o processo é finalizado.

QUANTIDADES DE VARIÁVEIS DE DECISÃO POSSÍVEIS:

Nest = 1

Num_UHE = vt, vv

Num_UTE = gt

Déficit

$N_{combinações} = Ndisc^{N_{usin}}$

CÁLCULO GENERALISTA

$Num_{UHE} * 2 + Num_{UTE} + 1$

FUNÇÃO OBJETIVO DE CADA PPL

$Min \sum_{j=1}^{NumUTE} CO_j \cdot GT_j \cdot def + CDEF + \alpha_{estadofuturo}$

RESTRIÇÕES DE BALANÇO HÍDRICO

Associado ao número de UHEs

$Vf = VI + AFL_{estagios,j} - V_t - V_v$

Quando as usinas estão em cascata: $Vf_b = VI_b + AFL_{b_{estagios,j}} - V_{t_b} - V_{v_b} + V_{t_a} + V_{v_a}$

$\square A \rightarrow \square B$

RESTRIÇÕES DE ATENDIMENTO À DEMANDA

AD = 1

$CARGA_{estagio} = \sum_{j}^{NumUHE} (\rho_j \cdot V_{t_j}) + \sum_{j}^{NumUTE} (GT_j) + def$

CÁLCULO GENERALISTA DAS RESTRIÇÕES

$NumUHE + 1$

RESTRIÇÕES DE CANALIZAÇÃO

$V_{min_j} \leq V_{f_j} \leq V_{máx_j}$

$0 \leq V_{t_j} \leq ENGOL$

$0 \leq V_{v_j} \leq \infty$

$0 \leq GT_j \leq GT_{máx_j}$

$0 \leq def \leq \infty$

$0 \leq \alpha \leq \infty$

CORTES - DECOMPOSIÇÃO DE BENDERS

- INEQUAÇÃO (Equação da reta α): O somatório é estipulando que haverá várias usinas

$\alpha \geq \sum_{I_{usin}}^{N_{usin}} A_{usin,cortes} \cdot V_{f_{usin}} + B_{cortes}$

- O número de cortes na FCF está associada a cada estágio e será sempre igual ao número total de iterações.

$$N_{iterações} \cdot 2 \cdot N_{estágios} + N_{estágios}$$

Gráficos comparando os resultados do PL ÚNICO, PDD e PDDD, demonstrado pelo Volume Armazanado Final e sua ρ produtibilidade.

$*$ Na PDE depende apenas do números de iterações

DADOS DO SISTEMA

In [ ]:

```
#INSERIR OS DADOS
```

ALGORÍTMO DA PROGRAMAÇÃO DINÂMICA DUAL DETERMINÍSTICA - PDDD

In [3]:

```
import numpy as np
from numpy import abs
import time

def pddd(sistema, cenario, imprime):
    #@
    pote_de_corte = []

    t = time.time ()

    tol = 0.01
    iteracao = 0
    ZINF = [0.]
    ZSUP = [np.inf]

    while np.abs(ZSUP [iteracao] - ZINF[iteracao]) > tol:
        #####################FORWARD
        ############################
        #Lista para quando acionar o Backward, porque precisa consultar
os resultados do Forward
        #Para verificar se obtivemos um resultado Ótimo
        memoria = []
        #O ZSUP irá acumular todos os custos imediatos obtidos ao longo
do processo
        #Por isso é iniciado por zero.
        ZSUP[iteracao] = 0.
```

```
        #Laço para percorrer os estágios
        for estag in range(sistema["DGer"]["Nest"]):
            # Para guardar o Volume inicial

            VI = []
            if estag == 0:
                for i, usin in enumerate(sistema ["UHE"]):
                    VI.append(usin["VI"])
            #Se não for o estágio 0, usamos o Volume final como
referência
            else:
                for i, usin in enumerate(resultado["UHE"]):
                    VI.append(usin["vf"])

            #Lita para a afluência e guardar os dados de acordo com o
cenário
            AFL = []


            #########FUNÇÃO DE DESPACHO HIDROTÉRMICO
            #Parâmetros de entrada = variáveis de decisão
            resultado = despacho_pddd(sistema, VI, AFL, pote_de_corte,
estag+1, imprime = False)
            #ZSUP acumula o custo imediato que é a diferença de Custo
total de custo futuro
            ZSUP[iteracao] += resultado["DGer"]["CustoTotal"] -
resultado["DGer"]["CustoFuturo"]

            if estag == 0:
                #zinf = fob
                ZINF[iteracao] = resultado["DGer"]["CustoTotal"]
            #Para finalizar o laço
            #Para acrescentar todos resultados de PPL
            memoria.append(resultado)


        #Se ZSUP e ZINF convergirem interropem o while com "break"
        #E obtemos um resultado ótimo
        if np.abs(ZSUP[iteracao] - ZINF[iteracao]) <= tol:
            break
        ZSUP.append(ZSUP[iteracao])
```

```
        ZINF.append(ZINF[iteracao])
        #Contador de iteracao
        iteracao += 1
        #FIM DO PROCESSO FORWARD



        #INÍCIO DO PROCESSO BACKWARD
        #Loop reverse de estágios (do fim para o início)
        #-1 é para inverter a contagem
        for estag in np.range(sistema["DGer"]["Nest"]-1,-1,-1):
            #
            VI = []
            if estag == 0:
                for i, usin in enumerate(sistema ["UHE"]):
                    VI.append(usin["VI"])
            #Se não for o estágio 0, usamos a lista MEMÓRIA como referência
            #Volume é volume final do estágio anterior
            else:
                for i, usin in enumerate(memoria[estag-1]["UHE"]):
                    VI.append(usin["vf"])

            #Lita para a afluência e guardar os dados de acordo com o cenário
            #Programada assim, pq nosso cenário é determinístico
            AFL = []
            for i, usin in enumerate(sistema ["UHE"]):
                AFL.append(usin["Afl"][estag][cenario])

        #Quando o problema for resolvido todos os resultados serão colocados na variável resultado
        #E nesse momento tem um corte e cáclulado no backward
        resultado = despacho_pddd(sistema, VI, AFL, pote_de_corte, estag+1, imprime = False)
        term_indep = resultado ["DGer"]["CustoTotal"]

        coeficiente = []
        for i, usin in enumerate(sistema ["UHE"]):
            coeficiente.append(-usin["cma"])
            term_indep -= VI[i]*coeficiente[i]
```

```
#DICIONÁRIO DO CORTE (INEQUAÇÃO)
#Corresponde a uma discretização
corte = {
    "Estagio":  estag,
    "Termo_indep" : term_indep,
    "Coefs": coeficiente
}

#Inserir os valores de corte na lista pote de cortes
pote_de_corte.appen(corte)
```

PLOTAR O GRÁFICO 1:40:13

COM OS VALORES DE PL ÚNICO, PDD E PDDD

In [ ]:

```
```