Aula 02 - Planejamento de Sistemas elétricos'

RESUMOS NO CADERNO E OBSIDIAN(Notebook Pessoal)

- Aula 00 e 01
- Aula 02a: Implementação Python da Técnica de Programação Dinâmica Estocástica (Adaptada)

In [64]:
```
usina = [ "UHE MARCATO", 100., 20., 0.95, 60.]
print(usina)
print(type(usina))
```

```
['UHE MARCATO', 100.0, 20.0, 0.95, 60.0]
<class 'list'>
```

DICIONÁRIO

In [65]:
```
usina = {
 "Nome": "UHE MARCATO",
 "Vmax": 100.,
 "Vmin": 20.,
 "Prod": 0.95,
 "Engol": 60.
}
print(usina["Engol"])
```

```
60.0
```

LISTA E DICIONÁRIO

In [66]:
```
lista_uhe = []
usina = {
 "Nome": "UHE MARCATO",
 "Vmax": 100.,
 "Vmin": 20.,
 "Prod": 0.95,
 "Engol": 60.
}
#comando append adiciona o dicionário na lista vazia
lista_uhe.append(usina)
print(len(usina))

usina = {
```

```
 "Nome": "UHE MARCATO 2",
 "Vmax": 300.,
 "Vmin": 50.,
 "Prod": 0.85,
 "Engol": 100.
}

lista_uhe.append(usina)
print(lista_uhe[0])
print(lista_uhe[1])

#comando for:para percorrer os elementos da lista
for usin in lista_uhe:
    print(usin["Nome"], usin["Engol"])
```

```
5
{'Nome': 'UHE MARCATO', 'Vmax': 100.0, 'Vmin': 20.0, 'Prod': 0.95, 'Engol': 60.0}
{'Nome': 'UHE MARCATO 2', 'Vmax': 300.0, 'Vmin': 50.0, 'Prod': 0.85, 'Engol': 100.
0}
UHE MARCATO 60.0
UHE MARCATO 2 100.0
```

APLICANDO GT E DADOS GERAIS E DICIONÁRIO COM TODOS OS DADOS DO SISTEMAS

In [67]:

```
lista_uhe = []
usina = {
 "Nome": "UHE MARCATO",
 "Vmax": 100.,
 "Vmin": 20.,
 "Prod": 0.95,
 "Engol": 60.,
 "AFL": [
    [ 23, 16],
    [ 19, 14],
    [ 15, 11]
 ]
     #Cenários = colunas e estágios=linhas
}
lista_uhe.append(usina)
#Lista de UTE - Usinas térmicas
lista_ute = []
#Primeira Usina Térmica
usina = {
    "Nome": "GT1",
```

```
    "Capacidade": 15.,
    "Custo": 10.
 }
lista_ute.append(usina)
#Segunda usina
usina = {
    "Nome": "GT2",
    "Capacidade": 10.,
    "Custo": 25.
}
lista_ute.append(usina)

#Dados Gerais - variavéis
d_gerais = {
    "CDef": 500.,
    "Carga": [50.,50.,50.,],#mes1, mês2, mês3, ...
    "N_Disc":3,
    "N_Est": 3,
    "N_Scenes": 2
}

#Sistema - Dicionário compactando todas as informações
sistema = {
    "DGer": d_gerais,
    "UHE": lista_uhe,
    "UTE": lista_ute
}
print(sistema["UTE"],[0],["Nome"])
```

```
[{'Nome': 'GT1', 'Capacidade': 15.0, 'Custo': 10.0}, {'Nome': 'GT2', 'Capacidade': 10.0, 'Custo': 25.0}] [0] ['Nome']
```

PROBLEMA DE DESPACHO TÉRMICO DA AULA 01

Tabela com os seguintes dados:

- Discretizações(Armazenamento(hm^3))
- Afluência
- Decisões Ótimas(Variavéis):
    1. UHEs (Volume final, Volume turbinado e Volume vertido),
    2. UTEs (Usar usina térmica 1 ou 2, GT1 e GT2)
    3. Def (Défict)
- Custo imediato
- Custo Ótimo
- Custo de Operação

FUNÇÃO OBJETIVO

Custo Imediato

$$Min \quad C_1 \cdot GT_1 + C_2 \cdot GT_2 + Cdef \cdot DEF + 0.01 \cdot V_v$$

BALANÇO HÍDRICO

$$V_f = V_i + AFL - V_t - V_v$$

ATENDIMENTO À DEMANDA

$$CARGA = \rho \cdot V_t + GT_1 + GT_2 + DEF$$

RESTRIÇÕES DE CANALIZAÇÃO

Limites superiores e inferiores das variavéis de decisão

$$20 \leq V_f \leq 100$$

$$0 \leq V_t \leq 60$$

$$0 \leq V_v \leq \infty$$

$$0 \leq GT_1 \leq 15$$

$$0 \leq GT_2 \leq 25$$

$$0 \leq DEF \leq \infty$$

In [72]:

```
import cvxopt
from cvxopt.modeling import variable, solvers
#Variable - define as variavéis de decisão
#Solvers - O problema de otimização

Vi = 60
AFL = 11
Num_UHE = len(sistema["UHE"])
Num_UTE = len(sistema["UTE"])
print(Num_UTE)

#Variavéis de Decisão de volume final
#Por isso a necessidade de Num_UTE e Num_UHE
```

```
vf = variable(Num_UHE, "Volume final da Usina")
vt = variable(Num_UHE, "Volume Turbinado da Usina")
vv = variable(Num_UHE, "Volume Vertido da Usina")
gt = variable(Num_UTE, "Geração na Usina Térmica")
deficit = variable(1, "Déficit de Energia no Sistema")


print(vf.name)
print(deficit.value)
#Após resolver o problema de otimização "none"

#CONSTRUÇÃO DA FUNÇÃO OBJETIVO
#MinC1·GT1+C2·GT2+Cdef·DEF+0.01·Vv
#MinC1·GT1+C2·GT2
fob = 0
for i,usin in enumerate(sistema["UTE"]):
    fob += usin['Custo']*gt[i]

#Cdef·DEF
fob += sistema["DGer"]["CDef"]*deficit[0]

#0.01·Vv
#E percorrer todas as usinas
for i,usin in enumerate(sistema["UHE"]):
    fob += 0.01*vv[i]

print(fob)


#DEFINIÇÃO DAS RESTRIÇÕES
#BALANÇO HÍDRICO
#Vf=Vi+AFL-Vt-Vv
restricoes = []
for i, usin in enumerate(sistema["UHE"]):
    restricoes.append(vf[i] == Vi + AFL - vt[i] - vv[i] )

#ATENDIMENTO À DEMANDA
#CARGA=ρ·Vt+GT1+GT2+DEF
#ρ·Vt
AD = 0
for i, usin in enumerate(sistema["UHE"]):
```

```
    AD += usin ["Prod"]*vt[i]

#GT1+GT2
for i, usin in enumerate(sistema["UTE"]):
    AD += gt[i]

#DEF
AD += deficit[0]

restricoes.append(AD == sistema["DGer"]["Carga"][2]) #2=estágios
print(restricoes[0])


#RESTRIÇÕES DE CANALIZAÇÃO
#Limites superiores e inferiores das variavéis de decisão
    #20≤Vf≤100
    #0≤Vt≤60
    #0≤Vv≤∞

for i, usin in enumerate(sistema["UHE"]):
    restricoes.append(vf[i] >= usin["Vmin"])
    restricoes.append(vf[i] <= usin["Vmax"])
    restricoes.append(vt[i] >= usin["Engol"])
    restricoes.append(vt[i] <= usin["Engol"])
    restricoes.append(vv[i] >= 0)

    #0≤GT1≤15
    #0≤GT2≤25
for i, usin in enumerate(sistema["UTE"]):
    restricoes.append(gt[i] >= 0)
    restricoes.append(gt <= usin["Capac"])

    #0≤DEF≤∞
restricoes.append( deficit[0] >= 0)
```

```
2
Volume final da Usina
None
linear function of length 1
linear term: linear function of length 1
coefficient of variable(2,'Geração na Usina Térmica'):
[ 1.00e+01  2.50e+01]
coefficient of variable(1,'Déficit de Energia no Sistema'):
[ 5.00e+02]
coefficient of variable(1,'Volume Vertido da Usina'):
[ 1.00e-02]

scalar equality
constraint function:
affine function of length 1
constant term:
[-1.11e+02]
linear term: linear function of length 1
coefficient of variable(1,'Volume final da Usina'):
[ 1.00e+00]
coefficient of variable(1,'Volume Turbinado da Usina'):
[ 1.00e+00]
coefficient of variable(1,'Volume Vertido da Usina'):
[ 1.00e+00]
```

RESOLVENDO PROBLEMAS DE OTIMIZAÇÃO

In [76]:

```
from cvxopt.modeling import op
#inserir a função objetivo e suas retrições
problema = op(fob, restricoes)

problema.solve('dense','glpk')
#Entender por que deu errado
#Usina Hídrica
print('Custo Total: ', fob.value())
for i, usin in enumerate(sistema["UHE"]):
    print(vf.name,i, "é",vf[i].value(),"hm³")
    print(vt.name,i, "é",vt[i].value(),"hm³")
    print(vv.name,i, "é",vv[i].value(),"hm³")

#Usina Térmica
for i, usin in enumerate(sistema["UTE"]):
    print(gt.name,i, "é",gt[i].value(),"MWmed")

#Déficit
print(deficit.name, "é",deficit[0].value(),"MWmed")

for i, usin in enumerate(sistema["UHE"]):
```

```
    print("O valor da água na usina,",i,"é: ",
restricoes[i].multiplier.value)

    print("O custo marginal de Operação é: ",
restricoes[Num_UHE].multiplier.value)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [76], in <cell line: 5>()
      2 #inserir a função objetivo e suas retrições
      3 problema = op(fob, restricoes)
----> 5 problema.solve('dense','glpk')
      6 #Entender por que deu errado
      7 #Usina Hídrica
      8 print('Custo Total: ', fob.value())

File ~\Anaconda3\lib\site-packages\cvxopt\modeling.py:2596, in op.solve(self, form
at, solver, **kwargs)
   2579 def solve(self, format='dense', solver = 'default', **kwargs):
   2581     """
   2582     Solves LP using dense or sparse solver.
   2583
   (...)
   2593     variables are set to a proof of dual infeasibility.
   2594     """
-> 2596     t = self._inmatrixform(format)
   2598     if t is None:
   2599         lp1 = self

File ~\Anaconda3\lib\site-packages\cvxopt\modeling.py:2575, in op._inmatrixform(se
lf, format)
   2572         mmap[i] = sum(mmap[i])
   2574 for e in  equalities:
-> 2575     mmap[e] = constraints[1].multiplier[eslc[e]]
   2576 return (op(cost, constraints), vmap, mmap)

IndexError: list index out of range
```

In [ ]:

In [ ]: