Dan's Baker

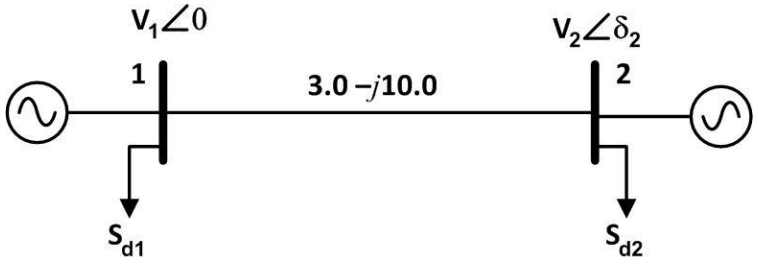
EENG 581: Power System Operation and Management

HW4: Economic Dispatch

Learning Objectives:

- Solve the fixed-loss economic dispatch problem using the Lagrangean method.
- Solve the economic dispatch problem with a nonlinear power loss model.

1. Consider the 2-bus power system shown below. The two units control the voltage magnitudes of the two buses at 1.0 per-unit. The loads at the two buses are $S_{d1} = 3.3 + j0.82$ and $S_{d2} = 0.52 + j0.15$, both in per-unit based on a 100 MVA base power. The cost parameters of the two units are provided below. Assume that the power losses in the system amount to 1% of the total load (which is a reasonable assumption for a transmission system). Formulate and solve the fixed-loss economic dispatch using the Lagrangean method.



Unit No.	Capacity (MW)	Quadratic Cost Coefficients		
		a ($/hr)	*b* ($/MWh)	*c* ($/(MW×MWh))
1	180	115.2	28.7	0.025
2	315	265.6	25.1	0.012

2. Suppose the input-output characteristics of three generating units are as given below. Determine the economic operation point for these three units when delivering a total of **(a)** 500MW and **(b)** 800MW power demand. Ignore the losses.

Unit No.	Minimum Generation Limit (MW)	Maximum Generation Limit (MW)	Quadratic Input-Output Coefficients		
			a (Btu/hr)	*b* (Btu/MWh)	*c* (Btu/(MW×MWh))
1	100	250	6	0.5	0.0006
2	100	250	5	0.6	0.0005
3	100	350	3	0.4	0.0007

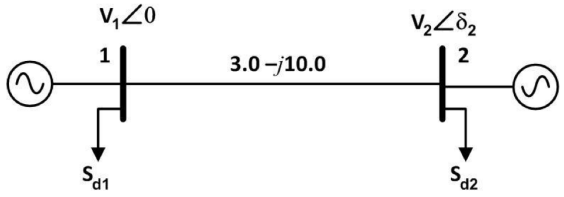
3. Assuming for the power system in problem **1** the system losses are given as a function below, solve the economic dispatch problem using a λ-iterative approach.

$$p^{loss} = 0.008 \times p_1 + 0.0001 \times p_1^2 + 0.0102 \times p_2 + 0.00005 \times p_2^2 + 0.00002 \times p_1 \cdot p_2$$

4. Consider the power system in problem **1**. Assuming the system losses are given as a function below, solve the economic dispatch problem using the Newton's method. Then find the participation factors of the two units.

$$p^{loss} = 0.008 \cdot p_1 + 0.0001 \cdot p_1^2 + 0.0102 \cdot p_2 + 0.00005 \cdot p_2^2 + 0.00002 \cdot p_1 \cdot p_2$$

1. Consider the 2-bus power system shown below. The two units control the voltage magnitudes of the two buses at 1.0 per-unit. The loads at the two buses are $S_{d1} = 3.3 + j0.82$ and $S_{d2} = 0.52 + j0.15$, both in per-unit based on a 100 MVA base power. The cost parameters of the two units are provided below. Assume that the power losses in the system amount to 1% of the total load (which is a reasonable assumption for a transmission system). Formulate and solve the fixed-loss economic dispatch using the Lagrangean method.



Unit No.	Capacity (MW)	Quadratic Cost Coefficients		
		a ($/hr)	*b* ($/MWh)	*c* ($/(MW×MWh))
1	180	115.2	28.7	0.025
2	315	265.6	25.1	0.012

Formulation

$$\min_{P_g} \underbrace{\sum_{g=1}^{2} f_g(P_g)}_{F} = \sum_{g=1}^{2} a_g + b_g \cdot P_g + c_g \cdot P_g^2$$

$$s.t. \quad \sum_{g=1}^{2} P_g = P^d + P^{loss}$$

where $P^d = P^{d1} + P^{d2} = 3.82 P^u$ and $P^{loss} = 0.01 P^d$

$P^u = 382$ MW

We'll ignore generation constraints for now.

Solution w/ Lagrangean method

$$\mathcal{L} = F + \lambda \left(P^d + P^{loss} - \sum_{g=1}^{2} P_g \right)$$

$$= \sum_{g=1}^{2} a_g + b_g P_g + c_g P_g^2 + \lambda \left(P^d + P^{loss} - \sum_{g=1}^{2} P_g \right)$$

$$\begin{cases} \forall g, \ \dfrac{\partial \mathcal{L}}{\partial P_g} = 0 \longrightarrow \dfrac{\partial F}{\partial P_g} = \lambda & (1) \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = 0 \longrightarrow P^d + P^{loss} - \sum_{g=1}^{2} P_g = 0 & (2) \end{cases}$$

Considering (1): $\forall g \quad b_g + 2 c_g P_g = \lambda$

$$P_g = \frac{\lambda - b_g}{2 c_g} \qquad (3)$$

And plugging into (2)

$$P^d + P^{loss} = \sum_{g=1}^{2} \frac{\lambda - b_g}{2 c_g}$$

λ is not a function of g, so

$$P^d + P^{loss} = \lambda \sum_g \frac{1}{2 c_g} - \sum_g \frac{b_g}{2 c_g}$$

$$\lambda = \frac{P^d + P^{loss} + \sum_g \frac{b_g}{2 c_g}}{\sum_g \frac{1}{2 c_g}}$$

$$\boxed{\lambda = 32.52}$$ $/additional MW

and we plug this back in to 3

```
p[1] = 76.48 MW
p[2] = 309.34 MW
```

These are both within the generator limits.

```julia
G = [1, 2]
a = [115.2, 265.6]
b = [28.7, 25.1]
c = [0.025, 0.012]

P_d = 382
P_loss = 0.01 * P_d

λ = ((P_d + P_loss) + sum(b[g] / (2c[g]) for g ∈ G)) / sum(1 / (2c[g]) for g ∈ G)
println("λ = $(round(λ, digits=2))")
println("")

for g ∈ G
    p = (λ - b[g]) / (2c[g])
    println("p[$g] = $(round(p, digits=2)) MW")
end
```

Or, if we wanted to use optimization software, ...

```julia
using JuMP
import GAMS
using DataFrames

model = Model(GAMS.Optimizer)
set_optimizer_attribute(model, "solver", "CPLEX")

G = [1, 2]
a = [115.2, 265.6]
b = [28.7, 25.1]
c = [0.025, 0.012]

P_d = 382
P_loss = 0.01 * P_d
p_max = [180, 315]

@variable(model, P[G] ≥ 0)
@objective(model, Min, sum(a[g] + b[g] * P[g] + c[g] * P[g]^2 for g ∈ G))
@constraint(model, sum(P[g] for g ∈ G) == P_d + P_loss)
# could also add max gen constraint
# @constraint(model, [g ∈ G], P[g] ≤ p_max[g])

optimize!(model)

v = all_variables(model)
df = DataFrame(
    variable=v,
    val=value.(v),
)
show(df, eltypes=false)
println("\n\n\tObjective value: \$$(objective_value(model))")
```

Which gives same result

variable	val
P[1]	76.4839
P[2]	309.336

---

2. Suppose the input-output characteristics of three generating units are as given below. Determine the economic operation point for these three units when delivering a total of **(a)** 500MW and **(b)** 800MW power demand. Ignore the losses.

Unit No.	Minimum Generation Limit (MW)	Maximum Generation Limit (MW)	Quadratic Input-Output Coefficients		
			a (Btu/hr)	b (Btu/MWh)	c (Btu/(MW×MWh))
1	100	250	6	0.5	0.0006
2	100	250	5	0.6	0.0005
3	100	350	3	0.4	0.0007

a) Again we have $\lambda = \dfrac{P^d + \sum_g \frac{b}{2c_g}}{\sum_g \frac{1}{2c_g}}$

$\lambda = 0.71 \frac{btu}{MW} \Longrightarrow P_g = \dfrac{\lambda - b_g}{2c_g}$

```
p[1] = 172.9 MW
p[2] = 107.48 MW
p[3] = 219.63 MW
```

b) Now it gets more interesting

$\lambda = 0.83 \frac{btu}{MW} \rightarrow$

```
p[1] = 271.03 MW
p[2] = 225.23 MW
p[3] = 303.74 MW
```

but this exceeds gen 1 limit, so we fix gen 1 at 250MW exclude it from the problem and recalculate λ

$\lambda = 0.84 \frac{btu}{MW} \rightarrow$ $P_1 = 250$ MW

```
p[2] = 237.5 MW
p[3] = 312.5 MW
```

and now all units are within limits

Code for problem 2

```julia
function calculate_economic_dispatch(P_d, G, a, b, c)
    λ = ((P_d) + sum(b[i] / (2c[i]) for (i, g) ∈ enumerate(G))) / sum(1 / (2c[i]) for (i, g) ∈ enumerate(G))
    println("λ = $(round(λ, digits=2))")
    println("")

    for (i, g) ∈ enumerate(G)
        p = (λ - b[i]) / (2c[i])
        println("p[$g] = $(round(p, digits=2)) MW")
    end
end

gens = 1:3
a = [6, 5, 3]
b = [0.5, 0.6, 0.4]
c = [0.0006, 0.0005, 0.0007]
calculate_economic_dispatch(500, gens, a, b, c)
calculate_economic_dispatch(800, gens, a, b, c)

# recalculate without first unit
calculate_economic_dispatch(800 - 250, gens[2:end], a[2:end], b[2:end], c[2:end])
```

We again could do this using optimization software and we get the same results.

```julia
using JuMP
import GAMS
using DataFrames

gens = 1:3
a = [6, 5, 3]
b = [0.5, 0.6, 0.4]
c = [0.0006, 0.0005, 0.0007]

function solve_economic_dispatch(P_d)

    model = Model(GAMS.Optimizer)
    set_optimizer_attribute(model, "solver", "CPLEX")

    p_max = [250, 250, 350]
    p_min = [100, 100, 100]

    @variable(model, P[G] ≥ 0)
    @objective(model, Min, sum(a[g] + b[g] * P[g] + c[g] * P[g]^2 for g ∈ gens))
    @constraint(model, sum(P[g] for g ∈ gens) == P_d)
    # gen constraints
    @constraint(model, [g ∈ gens], P[g] ≤ p_max[g])
    @constraint(model, [g ∈ gens], P[g] ≥ p_min[g])

    optimize!(model)

    v = all_variables(model)
    df = DataFrame(
        variable=v,
        val=value.(v),
    )
    show(df, eltypes=false)
    println("\n\n\tObjective value: \$$(objective_value(model))")
end

solve_economic_dispatch(500)
solve_economic_dispatch(800)
```

Demand = 500MW

DataFrame

variable	val
P[1]	172.906
P[2]	107.467
P[3]	219.627

Demand = 800MW

variable	val
P[1]	250.0
P[2]	237.436
P[3]	312.564

3. Assuming for the power system in problem **1** the system losses are given as a function below, solve the economic dispatch problem using a λ-iterative approach.

$$P^{loss} = 0.008 \times P_1 + 0.0001 \times P_1^2 + 0.0102 \times P_2 + 0.00005 \times P_2^2 + 0.00002 \times P_1 \cdot P_2$$

$$\frac{\partial q}{\partial P_1} = 0.008 + 0.0002 P_1 + 0.00002 P_2$$

From this eq $b_g + 2c_g P_g - \lambda\left(1 - \frac{\partial q}{\partial P_g}\right) = 0$ (see solution to #4)

$$\frac{\partial q}{\partial P_2} = 0.0102 + 0.0001 P_2 + 0.00002 P_1$$

for P_1: $b_1 + 2c_1 P_1 - \lambda + \lambda(0.008 + 0.0002 P_1 + 0.00002 P_2) = 0$

$$(2c_1 + 0.0002\lambda) P_1 + 0.00002\lambda P_2 = \lambda - 0.008\lambda - b_1$$

for P_2:

$$0.00002\lambda P_1 + (2c_2 + 0.0001\lambda) P_2 = \lambda - 0.0102\lambda - b_2$$

so we can solve for P_1 and P_2

$$\begin{bmatrix} 2c_1 + 0.0002\lambda & 0.00002\lambda \\ 0.00002\lambda & 2c_2 + 0.0001\lambda \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} = \begin{bmatrix} \lambda - 0.008\lambda - b_1 \\ \lambda - 0.0102\lambda - b_2 \end{bmatrix}$$

Then, with the results we get for P_1 and P_2, we see how far off we are from $P_1 + P_2 = P_d + P^{loss}$, adjust λ accordingly, and iterate.

iteration	λ $/MW	P_1 MW	P_2 MW	mismatch MW
fixed losses starting point	32.52	60.0241	258.669	70.4415
1	33.2244	71.8588	283.202	35.336
2	33.5778	77.7694	295.457	17.8382
3	33.7562	80.7466	301.631	9.03331
4	33.8465	82.2526	304.754	4.58173
5	33.8923	83.016	306.337	2.32572
6	33.9156	83.4034	307.14	1.18103
7	33.9274	83.6001	307.548	0.599865
8	33.9334	83.7	307.755	0.304713
9	33.9364	83.7507	307.861	0.154793
10	33.938	83.7765	307.914	0.0786365
11	33.9388	83.7896	307.941	0.0399487
12	33.9392	83.7962	307.955	0.0202947
13	33.9394	83.7996	307.962	0.0103102
14	33.9395	83.8013	307.966	0.00523781
15	33.9395	83.8022	307.967	0.00266093
16	33.9395	83.8026	307.968	0.00135182
17	33.9396	83.8029	307.969	0.000686756

solution ↗

```julia
using PrettyTables
using DataFrames

G = [1, 2]
a = [115.2, 265.6]
b = [28.7, 25.1]
c = [0.025, 0.012]
P_d = 382

# initial guess as the answer from fixed loss ED
λ = 32.52

q(p1, p2) = 0.008p1 + 0.0001p1^2 + 0.0102p2 + 0.00005p2^2 + 0.00002p1 * p2

p1_arr = []
p2_arr = []
λ_arr = []
mismatch_arr = []

iteration = 0
while true
    iteration += 1
    push!(λ_arr, λ)

    A = [
        2c[1]+0.0002λ 0.00002λ
        0.00002λ 2c[2]+0.0001λ
    ]
    rhs = [λ - 0.008λ - b[1]
        λ - 0.0102λ - b[2]]

    p_new = (A^-1) * rhs

    p1 = p_new[1]
    p2 = p_new[2]

    push!(p1_arr, p1)
    push!(p2_arr, p2)

    mismatch = P_d + q(p1, p2) - (p1 + p2)
    push!(mismatch_arr, mismatch)

    if abs(mismatch) < 1e-3
        break
    end

    # update λ for next iteration
    λ = λ + 0.01 * mismatch
end
df = DataFrame(
    Iteration=["fixed losses starting point", 1:iteration-1...],
    λ=λ_arr,
    p1=p1_arr,
    p2=p2_arr,
    mismatch=mismatch_arr,
)
header = (["iteration", "λ", "p₁", "p₂", "mismatch"], ["", "\$/MW", "MW", "MW", "MW"])
pretty_table(df, header=header, header_crayon=crayon"yellow bold")
```

4. Consider the power system in problem **1**. Assuming the system losses are given as a function below, solve the economic dispatch problem using the Newton's method. Then find the participation factors of the two units.

$$p^{loss} = 0.008 \cdot p_1 + 0.0001 \cdot p_1^2 + 0.0102 \cdot p_2 + 0.00005 \cdot p_2^2 + 0.00002 \cdot p_1 \cdot p_2$$

Again, $\min_{P_g} \sum_g f_g(P_g) = \sum_g a_g + b_g P_g + c_g P_g^2$

$$\text{st.} \quad \sum_g P_g = P^d + \underbrace{q(P_1, P_2)}_{P^{loss}}$$

$$\mathcal{L} = F + \lambda \left(P^d + P^{loss} - \sum_g P_g \right)$$

$$= \sum_g a_g + b_g P_g + c_g P_g^2 + \lambda \left(P^d + P^{loss} - \sum_g P_g \right)$$

$$\begin{cases} \forall g, \ \dfrac{\partial \mathcal{L}}{\partial P_g} = 0 \longrightarrow 0 = \dfrac{\partial F}{\partial P_g} + \lambda \left(\dfrac{\partial q}{\partial P_g} - 1 \right) \quad \text{or, equivalently} \quad \dfrac{\partial F}{\partial P_g} - \lambda \left(1 - \dfrac{\partial q}{\partial P_g} \right) = 0 \quad (1) \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = 0 \longrightarrow P^d + q(P_1, P_2) - \sum_{g=1}^{2} P_g = 0 \quad (2) \end{cases}$$

$$b_g + 2c_g P_g - \lambda \left(1 - \frac{\partial q}{\partial P_g} \right) = 0$$

Newton's method: $P_g = P_g^0 + \Delta P_g$, $\lambda = \lambda_0 + \Delta\lambda$, plug into (1) and (2)

(1) becomes $\forall g, \left(b_g + 2c_g P_g^0 - \lambda_0 \left(1 - \frac{\partial q}{\partial P_g} \right)\Big|_{P_g^0} \right) + 2c_g \Delta P_g + \Delta\lambda \left(1 - \frac{\partial q}{\partial P_g} \right)\Big|_{P_g^0} = 0$

(2) becomes $\sum_g P_g^0 - P^d - q(P_1^0, P_2^0) + \sum_g \left(1 - \frac{\partial q}{\partial P_g} \right) \Delta P_g = 0$

$$x = \begin{bmatrix} P_1 \\ P_2 \\ \lambda \end{bmatrix} \qquad J = \begin{bmatrix} \frac{\partial f_1}{\partial P_1} & \frac{\partial f_1}{\partial P_2} & \frac{\partial f_1}{\partial \lambda} \\ \frac{\partial f_2}{\partial P_1} & \cdots & \cdots \\ \frac{\partial f_3}{\partial P_1} & \cdots & \cdots \end{bmatrix}$$

where $f_1 = b_1 + 2c_1 P_1 - \lambda \left(1 - \frac{\partial q}{\partial P_1} \right)$

and $\frac{\partial q}{\partial P_1} = 0.008 + 0.0002 P_1 + 0.00002 P_2$

$f_2 = b_2 + 2c_2 P_2 - \lambda \left(1 - \frac{\partial q}{\partial P_2} \right)$

and $\frac{\partial q}{\partial P_2} = 0.0102 + 0.0001 P_2 + 0.00002 P_1$

$f_3 = P_1 + P_2 - q(P_1, P_2) - P^d$

$$J = \begin{bmatrix} 2c_1 + \lambda(0.0002) & \lambda(0.00002) & \frac{\partial q}{\partial P_1} - 1 \\ \lambda(0.00002) & 2c_2 + \lambda(0.0001) & \frac{\partial q}{\partial P_2} - 1 \\ 1 - \frac{\partial q}{\partial P_1} & 1 - \frac{\partial q}{\partial P_2} & 0 \end{bmatrix}$$

$$J \Delta x = -F \quad \text{where} \quad F = \begin{bmatrix} f_1(p_1, p_2, \lambda) \\ f_2(p_1, p_2, \lambda) \\ f_3(p_1, p_2 \end{bmatrix}$$

$$\Delta x = -J^{-1} F$$

$$x^{\nu+1} - x^{\nu} = -J^{-1} F$$

$$x^{\nu+1} = x^{\nu} - J^{-1} F$$

iteration	p_1 MW	p_2 MW	λ $/MW
fixed losses starting point	76.48	309.34	32.52
1	83.8257	307.941	33.9388
2	83.8031	307.969	33.9396
3	83.8031	307.969	33.9396

Participation Factors:

```
Participation factor for g1: 0.338
Participation factor for g2: 0.702
```

Code to get these results:

```julia
using PrettyTables
using DataFrames

G = [1, 2]
a = [115.2, 265.6]
b = [28.7, 25.1]
c = [0.025, 0.012]

function check_for_convergence(x_arr)
    cur = x_arr[end]
    prev = x_arr[end-1]
    for i in eachindex(cur)
        if abs(cur[i] - prev[i]) > 1e-6
            return false
        end
    end
    return true
end

function solve_economic_dispatch(Δ_p=0)

    P_d = 382 + Δ_p

    dq_dp1(p1, p2) = 0.008 + 0.0002p1 + 0.00002p2
    dq_dp2(p1, p2) = 0.0102 + 0.0001p2 + 0.00002p1

    # losses
    q(p1, p2) = 0.008p1 + 0.0001p1^2 + 0.0102p2 + 0.00005p2^2 + 0.00002p1 * p2

    f1(p1, p2, λ) = b[1] + 2c[1] * p1 - λ * (1 - dq_dp1(p1, p2))
    f2(p1, p2, λ) = b[2] + 2c[2] * p2 - λ * (1 - dq_dp2(p1, p2))
    f3(p1, p2) = p1 + p2 - q(p1, p2) - P_d


    # we start with our solution to fixed loss economic dispatch
    # shape of x is [p1,p2,λ]
    x_arr = [[76.48, 309.34, 32.52]]

    iteration = 1
    while true
        x = (p1, p2, λ) = x_arr[end]

        J = [
            2c[1]+0.0002λ 0.00002λ dq_dp1(p1, p2)-1
            0.00002λ 2c[2]+0.0001λ dq_dp2(p1, p2)-1
            1-dq_dp1(p1, p2) 1-dq_dp2(p1, p2) 0
        ]

        F = [f1(p1, p2, λ); f2(p1, p2, λ); f3(p1, p2)]

        x_new = x - (J^-1) * F
        push!(x_arr, x_new)
        if (check_for_convergence(x_arr))
            println("converged in $iteration iterations")
            break
        end
        iteration += 1
    end

    df = DataFrame(
        Iteration=["fixed losses starting point", 1:iteration...],
        p1=[x[1] for x ∈ x_arr],
        p2=[x[2] for x ∈ x_arr],
        λ=[x[3] for x ∈ x_arr]
    )
    header = (["iteration", "p₁", "p₂", "λ"], ["", "MW", "MW", "\$/MW"])
    pretty_table(df, header=header, header_crayon=crayon"yellow bold")
    return x_arr[end]
end

(p1, p2, λ) = solve_economic_dispatch()
(p1_larger_load, p2_larger_load, λ) = solve_economic_dispatch(1) # incrementing the load by 1 MW to get participation factors

pf_1 = p1_larger_load - p1
pf_2 = p2_larger_load - p2

println("Participation factor for g1: $(round(pf_1, digits=3))")
println("Participation factor for g2: $(round(pf_2, digits=3))")
```

Can check our answer using JuMP w/ a nonlinear solver (Ipopt)

```julia
using JuMP
import GAMS
using DataFrames
using HiGHS
using Ipopt

G = [1, 2]
a = [115.2, 265.6]
b = [28.7, 25.1]
c = [0.025, 0.012]
P_d = 382

model = Model(Ipopt.Optimizer)
p_max = [180, 315]

@variable(model, P[G] ≥ 0)
@objective(model, Min, sum(a[g] + b[g] * P[g] + c[g] * P[g]^2 for g ∈ G))
q_loss = 0.008P[1] + 0.0001P[1]^2 + 0.0102P[2] + 0.00005P[2]^2 + 0.00002P[1] * P[2]

@constraint(model, sum(P[g] for g ∈ G) == P_d + q_loss)
# could also add max gen constraint
@constraint(model, [g ∈ G], P[g] ≤ p_max[g])

optimize!(model)

v = all_variables(model)
df = DataFrame(
    variable=v,
    val=value.(v),
)
show(df, eltypes=false)
println("\n\n\tObjective value: \$$(objective_value(model))")

for c in all_constraints(model, include_variable_in_set_constraints=false)
    println(c)
    println("dual: $(dual(c)) \n\n")
end
```

variable	val
P[1]	83.8031
P[2]	307.969