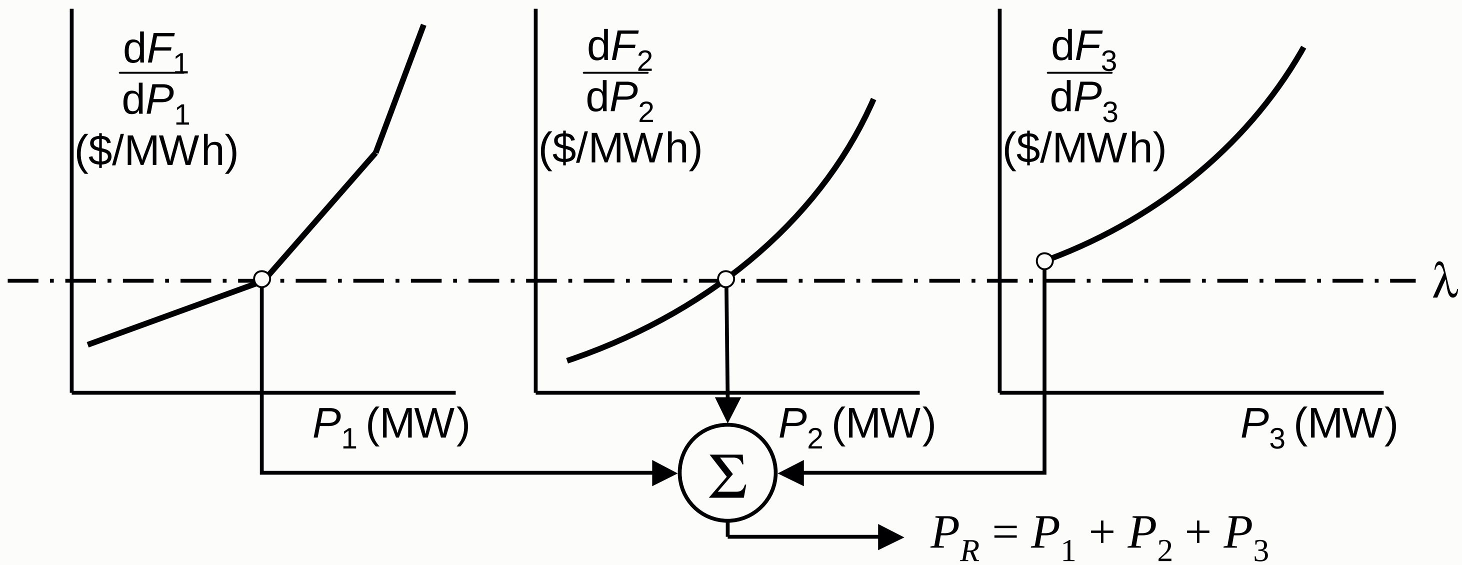
EEL 6266
Power System Operation and Control

Chapter 3

Numerical Methods for Economic Dispatch

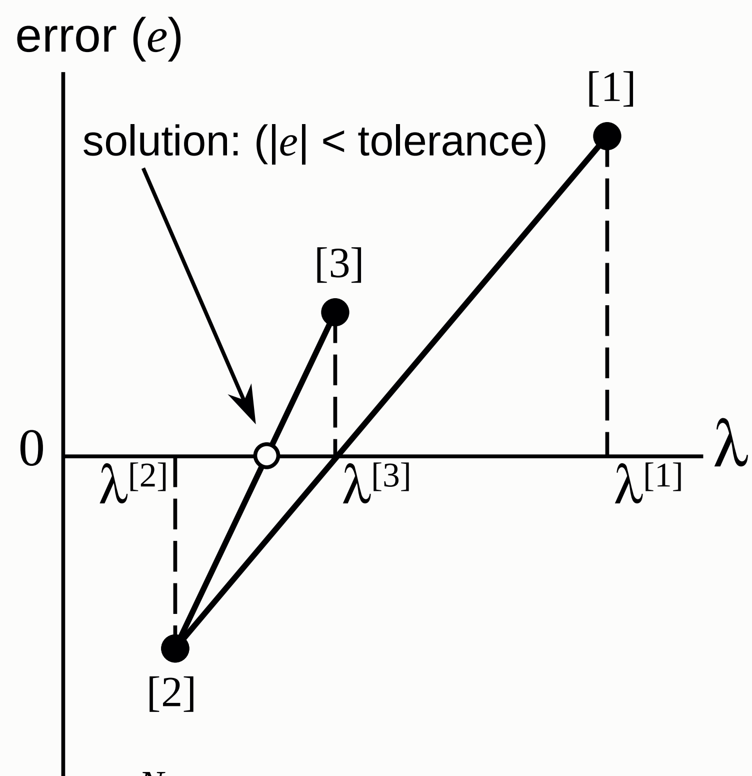
The Lambda-Iteration Method

- The solution to the optimal dispatch can be approached by graphical methods
  - plot the incremental cost characteristics for each generator
  - the operating points must have minimum cost and satisfy load
    - that is, find an incremental cost rate, λ that meets the demand P_R
    - graphically:



The Lambda-Iteration Method

- An iterative process
  - assume an incremental cost rate λ and find the sum of the power outputs for this rate
    - the first estimate will be incorrect
  - if the total power output is too low, increase the λ value, or if too high, decrease the λ value
    - with two solutions, a closer value of total power can be extrapolated or interpolated
  - the steps are repeated until the desired output is reached



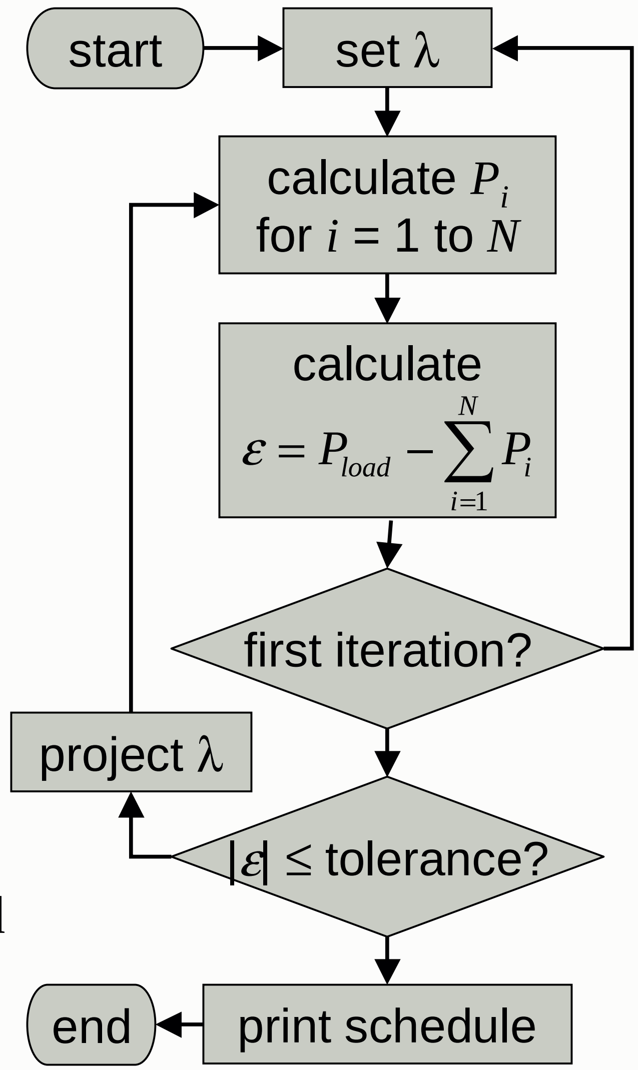
$$e = \sum_{i=1}^{N} P_i - P_R$$

Lambda projection

© 2002, 2004 Florida State University

The Lambda-Iteration Method

- This procedure can be adopted for a computer implementation
  - the implementation of the power output calculation is rather independent of the solution method
    - each generator output could be solved by a different method
  - as an iterative procedure, a stopping criterion must be established
    - two general stopping rules are appropriate for this application
      - total output power is within a specified tolerance of the load demand
      - iteration loop count exceeds a maximum value



The Lambda-Iteration Method

- Example
  - consider the use of cubic functions to represent the input-output characteristics of generating plants

    $$H\ (\text{MBtu/h}) = A + BP + CP^2 + DP^3 \quad (P \text{ in MW})$$

  - for three generating units, find the optimum schedule for a 2500 MW load demand using the lambda-iteration method
    - generator characteristics:

	A	***B***	***C***	***D***	P_{max}	P_{min}
Unit 1	749.55	6.95	9.68×10^{-4}	1.27×10^{-7}	320	800
Unit 2	1285.0	7.051	7.375×10^{-4}	6.453×10^{-8}	300	1200
Unit 3	1531.0	6.531	1.04×10^{-3}	9.98×10^{-8}	275	1100

    - assume that the fuel cost to be $1/MBtu
    - set the value of λ on the second iteration at 10% above or below the starting value depending on the sign of the error

The Lambda-Iteration Method

- Example
  - initial iteration: $\lambda_{start} = 8.0$
    - incremental cost functions

$$\lambda = \mathrm{d}F_1/\mathrm{d}P_1 = 6.95 + 2\left(9.68\times10^{-4}\right)P_1 + 3\left(1.27\times10^{-7}\right)P_1^2$$

$$\lambda = \mathrm{d}F_2/\mathrm{d}P_2 = 7.051 + 2\left(7.375\times10^{-4}\right)P_2 + 3\left(6.453\times10^{-8}\right)P_2^2$$

$$\lambda = \mathrm{d}F_3/\mathrm{d}P_3 = 6.531 + 2\left(1.04\times10^{-3}\right)P_3 + 3\left(9.98\times10^{-8}\right)P_3^2$$

    - find the roots of the three incremental cost functions at $\lambda = 8.0$
      - P_1 = (–5575.6, 494.3), P_2 = (–8215.9, 596.7), P_3 = (–7593.4, 646.2)
      - use only the positive values within the range of the generator upper and lower output limits
    - calculate the error

$$e = 2500 - (494.3) - (596.7) - (646.2) = 762.9 \text{ MW/h}$$

    - with a positive error, set second λ at 10% above λ_{start}: $\lambda^{[2]} = 8.8$

The Lambda-Iteration Method

- Example
  - second iteration: $\lambda^{[2]} = 8.8$
    - find the roots of the three incremental cost functions at $\lambda = 8.8$
      - $P_1 = (-5904, 822.5)$, $P_2 = (-8662, 1043.0)$, $P_3 = (-7906, 958.6)$
    - calculate the error

$$e = 2500 - (822.5) - (1043) - (958.6) = -324.0 \text{ MW/h}$$

      - error out of tolerance
    - project λ

$$\lambda^{[3]} = \frac{\lambda^{[2]} - \lambda^{[1]}}{e^{[1]} - e^{[2]}}\left(e^{[2]}\right) + \lambda^{[2]} = \frac{8.8 - 8.0}{762.9 + 324.0}(-324.0) + 8.8 = 8.5615$$

    - continue with third iteration

The Lambda-Iteration Method

- Example
  - results of all iterations

Iteration	λ	Total Generation	P_1	P_2	P_3
1	8.0	1737.2	494.3	596.7	646.2
2	8.8	2824.1	822.5	1043.0	958.6
3	8.5615	2510.2	728.1	914.3	867.8
4	8.5537	2499.9	725.0	910.1	864.8

- Issues
  - under some initial starting points, the lambda-iteration approach exhibits an oscillatory behavior, resulting in a non-converging solution
    - try the example again with a starting point of $\lambda_{start} = 10.0$

The Gradient Method

- Suppose that the cost function is more complex
  - example: $F(P) = a_0 + a_1\, P + a_2\, P^{2.5} + a_3\, e^{\frac{P-a_4}{a_5}}$
  - the lambda search technique requires the solution of the generator output power for a given incremental cost
    - possible with a quadratic function or piecewise linear function
    - hard for complicated functions; we need a more basic method
- The gradient search method uses the principle that the minimum is found by taking steps in a downward direction
  - from any starting point, $x^{[0]}$, one finds the direction of steepest descent by computing the negative gradient of F at $x^{[0]}$: $-\nabla F\left(x^{[0]}\right) = -\begin{bmatrix} \partial F/dx_1 \\ \vdots \\ \partial F/dx_n \end{bmatrix}$

The Gradient Method

- to move in the direction of maximum descent from $x^{[0]}$ to $x^{[1]}$:

$$x^{[1]} = x^{[0]} - \alpha \nabla f\left(x^{[0]}\right)$$

  - α is a scalar that when properly selected guarantees that the process converges
  - the best value of α must be determined by experiment

- for the economic dispatch problem, the gradient technique is applied directly to the Lagrange function

$$L = \sum_{i=1}^{N} F_i\left(P_i\right) + \lambda\left(P_{load} - \sum_{i=1}^{N} P_i\right)$$

- the gradient function is: $$\nabla L = \begin{bmatrix} \partial L/\partial P_1 \\ \vdots \\ \partial L/\partial P_N \\ \partial L/\partial \lambda \end{bmatrix} = \begin{bmatrix} (\mathrm{d}/\mathrm{d}P_1)F_1(P_1) - \lambda \\ \vdots \\ (\mathrm{d}/\mathrm{d}P_N)F_N(P_N) - \lambda \\ P_{load} - \sum_{i=1}^{N} P_i \end{bmatrix}$$

  - this formulation does not enforce the constraint function

The Gradient Method

- Example
  - solve the economic dispatch for a total load of 800 MW using these generator cost functions

$$F_1(P_1) = 1683 + 23.76 P_1 + 0.004686 P_1^2$$

$$F_2(P_2) = 930 + 23.55 P_2 + 0.00582 P_2^2$$

$$F_3(P_3) = 234 + 23.70 P_3 + 0.01446 P_3^2$$

    - use α = 100% and starting from

$$P_1^{[0]} = 300 \text{ MW}, \;\; P_2^{[0]} = 200 \text{ MW, and } \;\; P_3^{[0]} = 300 \text{ MW}$$

    - λ is initially set to the average of the incremental costs of the generators at their starting generation values:

$$\lambda^{[0]} = \tfrac{1}{3} \sum_{i=1}^{3} \frac{\mathrm{d}}{\mathrm{d}P_1} F_i\left(P_i^{[0]}\right) = \frac{1}{3}\begin{bmatrix} 23.76 + 0.009372(300) + \\ 23.55 + 0.01164(200) + \\ 23.70 + 0.02892(300) \end{bmatrix} = 28.27$$

The Gradient Method

- Example
  - Matlab program to perform the gradient search method

```
% Example 3E
gendata = [ 1683  23.76  0.004686
             930  23.55  0.00582
             234  23.70  0.01446 ];
power = [ 300, 200, 300 ];
alpha = 1.00, Pload = 800;
% find lambda0
n = length( gendata );
lambda0 = 0;
for i = 1 : n
   lambda0 = lambda0 + gendata(i,2) + 2 * gendata(i,3) * power(i);
end
lambda0 = lambda0 / 3
clear x0
x0 = power, x0(n+1) = lambda0;
% calculate the gradient
for kk = 1 : 10
   disp(kk)
   clear  gradient
   gradient = [];
   Pgen = 0, cost = 0;
   for i = 1 : n
      gradient(i) = gendata(i,2) + 2 * gendata(i,3) * x0(i) - x0(n+1);
      Pgen = Pgen + x0(i);
      cost = cost + gendata(i,1) + gendata(i,2) * x0(i) + gendata(i,3) * x0(i) * x0(i);
   end
   gradient(n+1) = Pload - Pgen;
   disp( [x0, Pgen, cost/1000] )
   x1 = x0 - gradient * alpha;
   x0 = x1;
end
```

The Gradient Method

- Example
  - the progress of the gradient search is shown in the table below

Iteration	λ	Total Generation	P_1	P_2	P_3	Cost
1	28.28	800.0	300.0	200.0	300.0	23,751
2	28.28	800.0	301.7	202.4	295.9	23,726
3	28.28	800.1	303.4	204.8	291.9	23,704
4	28.35	800.2	305.1	207.1	288.1	23,685
5	28.57	800.7	306.8	209.5	284.4	23,676
6	29.23	801.8	308.7	212.1	281.0	23,687
7	31.06	805.0	311.3	215.3	278.4	23,757
8	36.08	813.7	315.7	220.3	277.7	23,983
9	49.79	837.4	325.1	230.3	282.1	24,632
10	87.19	901.9	348.0	253.8	300.0	26,449

  - note that there is no convergence to a solution

The Gradient Method

- A simple variation
  - realize that one of the generators is always a dependent variable and remove it from the problem
    - for example, picking P_3, then $P_3 = 800 - P_1 - P_2$
    - then the total cost function becomes
      $$C = F_1(P_1) + F_2(P_2) + F_3(800 - P_1 - P_2)$$
    - this function stands by itself as a function of two variables with no load-generation balance constraint
      - the cost can be minimized by a gradient method such as:
        $$\nabla C = \begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}P_1} C \\ \frac{\mathrm{d}}{\mathrm{d}P_2} C \end{bmatrix} = \begin{bmatrix} \frac{\mathrm{d}F_1}{\mathrm{d}P_1} - \frac{\mathrm{d}F_3}{\mathrm{d}P_1} \\ \frac{\mathrm{d}F_2}{\mathrm{d}P_2} - \frac{\mathrm{d}F_3}{\mathrm{d}P_2} \end{bmatrix}$$
      - note that the gradient goes to zero when the incremental cost at generator 3 is equal to that at generators 1 and 2

The Gradient Method

- A simple variation
  - the gradient steps are performed in like manner as before

    $$x^{[1]} = x^{[0]} - \nabla C \cdot \alpha$$

    and

    $$x = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$$

- Example
  - rework the previous example with the reduced gradient

    $$\nabla C = \begin{bmatrix} \dfrac{\mathrm{d}F_1}{\mathrm{d}P_1} - \dfrac{\mathrm{d}F_3}{\mathrm{d}P_1} \\ \dfrac{\mathrm{d}F_2}{\mathrm{d}P_2} - \dfrac{\mathrm{d}F_3}{\mathrm{d}P_2} \end{bmatrix} = \begin{bmatrix} 23.76 + 2(0.004686)P_1 - 23.70 - 2(0.01446)(800 - P_1 - P_2) \\ 23.55 + 2(0.00582)P_2 - 23.70 - 2(0.01446)(800 - P_1 - P_2) \end{bmatrix}$$

  - α is set to 20.00

The Gradient Method

- Example
  - Matlab program to perform the simplified gradient search method

```
% Example 3F
gendata = [ 1683  23.76  0.004686
             930  23.55  0.00582
             234  23.70  0.01446 ];
power = [ 300, 200, 300 ];
alpha = 20.00;
Pload = 800;
% form lambda0
n = length( gendata );
clear x0
x0 = power(1:n-1);
% calculate the gradient
for kk = 1 : 10
   disp(kk)
   clear  gradient
   gradient = [];
   Pn = Pload;
   for i = 1 : n - 1
      Pn = Pn - x0(i);
   end
   cost = gendata(n,1) + gendata(n,2) * Pn + gendata(n,3) * Pn * Pn;
   for i = 1 : n - 1
      gradient(i) = gendata(i,2) + 2 * gendata(i,3) * x0(i) - gendata(n,2) - 2 * gendata(n,3) * Pn;
      cost = cost + gendata(i,1) + gendata(i,2) * x0(i) + gendata(i,3) * x0(i) * x0(i);
   end
   disp( [x0, Pn, 800, cost/1000] )
   x1 = x0 - gradient * alpha;
   x0 = x1;
end
```

The Gradient Method

- Example
  - the progress of the simplified gradient search is shown in the table below

Iteration	Total Generation	P_1	P_2	P_3	Cost
1	800.0	300.0	200.0	300.0	23,751
2	800.0	416.1	330.0	54.0	23,269
3	800.0	368.1	287.4	144.5	23,204
4	800.0	381.5	307.1	111.4	23,194
5	800.0	373.3	303.0	123.7	23,193
6	800.0	373.6	307.0	119.3	23,192
7	800.0	371.4	307.6	121.0	23,192
8	800.0	370.6	309.0	120.4	23,192
9	800.0	369.6	309.7	120.7	23,192
10	800.0	368.9	310.4	120.6	23,192

  - note that there is a solution convergence by the 6th iteration

Newton's Method

- The solution process can be taken one step further
  - observe that the aim is to always drive the gradient to zero

    $\nabla L_x = 0$
  - since this is just a vector function, Newton's method finds the correction that exactly drives the gradient to zero
- Review of Newton's method
  - suppose it is desired to drive the function $g(x)$ to zero
    - the first two terms of the Taylor's series suggest the following

      $g(x + \Delta x) = g(x) + [g'(x)]\Delta x = 0$
    - the objective function $g(x)$ is defined as: $g(x) = \begin{bmatrix} g_1(x_1, \cdots, x_n) \\ \vdots \\ g_n(x_1, \cdots, x_n) \end{bmatrix}$
    - then the Jacobian is:

$$g'(x) = \begin{bmatrix} \partial g_1/\partial x_1 & \cdots & \partial g_1/\partial x_n \\ \vdots & \ddots & \vdots \\ \partial g_n/\partial x_1 & \cdots & \partial g_n/\partial x_n \end{bmatrix}$$

Newton's Method

- the adjustment at each iteration step is $\Delta x = -[g'(x)]^{-1} g(x)$
- if the function g is the gradient vector ∇L_x, then

$$\Delta x = -\left[\frac{\partial}{\partial x}\nabla L_x\right]^{-1} \Delta L$$

- For economic dispatch problems: $L = \sum_{i=1}^{N} F_i(P_i) + \lambda\left(P_{load} - \sum_{i=1}^{N} P_i\right)$

and $$\frac{\partial}{\partial x}\nabla L_x = \begin{bmatrix} \frac{d^2 L}{dx_1^2} & \frac{d^2 L}{dx_1 dx_2} & \cdots \\ \frac{d^2 L}{dx_2 dx_1} & \frac{d^2 L}{dx_2^2} & \cdots \\ \vdots & \vdots & \ddots \\ \frac{d^2 L}{d\lambda dx_1} & \frac{d^2 L}{d\lambda dx_2} & \cdots \end{bmatrix}$$

  - note that in general, one Newton step solves for a correction that is closer to the minimum than would the gradient method

Newton's Method

- Example
  - solve the previous economic dispatch problem example using the Newton's method
    - the gradient function is the same as in the first example
      - let the initial value of λ be equal to zero
    - the Hessian matrix takes the following form:

$$[H]=\begin{bmatrix} \frac{d^2F_1}{dP_1^2} & 0 & 0 & -1 \\ 0 & \frac{d^2F_2}{dP_2^2} & 0 & -1 \\ 0 & 0 & \frac{d^2F_3}{dP_3^2} & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

    - the initial generation values are also the same as in the first example

Newton's Method

```
% Example 3G
gendata = [ 1683  23.76  0.004686
             930  23.55  0.00582
             234  23.70  0.01446 ];
power = [ 300, 200, 300 ];
Pload = 800;
% form H
n = length( gendata );
H = zeros(n+1,n+1);
for i = 1 : n
   H(i,i) = gendata(i,3) * 2;
   H(i,n+1) = -1, H(n+1,i) = -1; end
x0 = zeros(n+1,1);
x0(1:n,1) = transpose( power );
% calculate the gradient and Hessian matrices
for kk = 1 : 10
   disp(kk)
   gradient = zeros(n+1,1);
   gradient(n+1,1) = Pload;
   for i = 1 : n
      gradient(i,1) = gendata(i,2) + 2 * gendata(i,3) * x0(i,1) - x0(n+1,1);
      gradient(n+1,1) = gradient(n+1,1) - x0(i,1); end
   dx = H \ gradient;
   cost = 0;
   for i = 1 : n
      cost = cost + gendata(i,1) + gendata(i,2) * x0(i) + gendata(i,3) * x0(i) * x0(i); end
   disp( [x0', cost/1000] )
   x0 = x0 - dx;
end
```

- Example
  - Matlab program to perform the Newton's method

Newton's Method

- Example
  - the progress of the gradient search is shown in the table below

Iteration	λ	Total Generation	P_1	P_2	P_3	Cost
1	0.00	800.0	300.0	200.0	300.0	23,751
2	27.19	800.0	366.3	313.0	120.7	23,192
3	27.19	800.0	366.3	313.0	120.7	23,192
4	27.19	800.0	366.3	313.0	120.7	23,192

  - note the quick convergence to a solution
  - compare with the solution of the previous example