

Linear Programming Approach to Economic Dispatch Using Piecewise Linear Cost Curves

1.0 Overview

In EE 303, we saw one way to solve the economic dispatch problem (with fixed demand and multiple units) minimum cost problem. In module E3, we formed a Lagrangian function, applied Karush-Kuhn-Tucker (KKT) conditions, and then solved the resulting set of equations. The solution to these equations requires only one step, if the original cost curves are quadratics since the resulting equations are all linear and may be put in the form of $\underline{Ax}=\underline{b}$. It is possible to use a generalization of this approach if the cost curves have some other form beside quadratic, but solution requires an iterative approach. The solution approach is called a Newton approach. This approach works as long as cost curves are convex.

There is another approach that we do not have time to see, but at least I want to mention it. It is called Lambda iteration. This approach works very well independent of the form of the cost curves, as long as they are convex.

We would like to illustrate one more approach that connects to the material addressed by Dr. Kumar. Here we will utilize linear programming. To do so, however, we recognize that the cost curves are not linear and therefore do not satisfy the requirements of a linear program. We can circumvent this problem, however, by using a piecewise linear approximation of the cost curves.

Some of the material in these notes is adapted from [1].

2.0 Piecewise linear approximation of cost curves

Consider the nonlinear cost curve shown in Fig. 1.

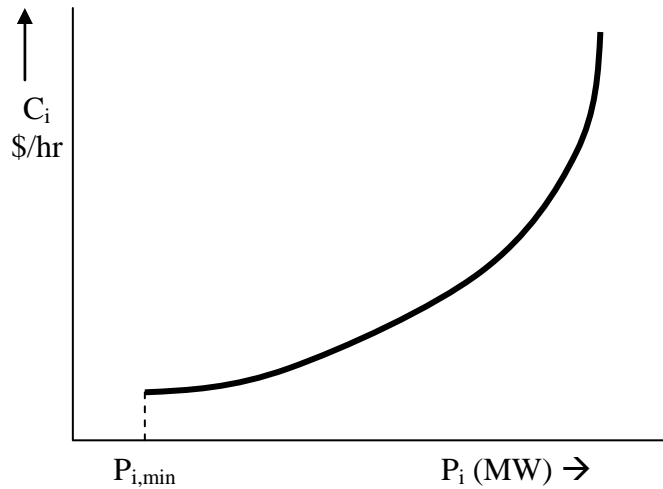


Fig. 1: Typical Cost Curve

We can approximate the nonlinear function of Fig. 1 with a series of straight-line segments as illustrated in Fig. 2.

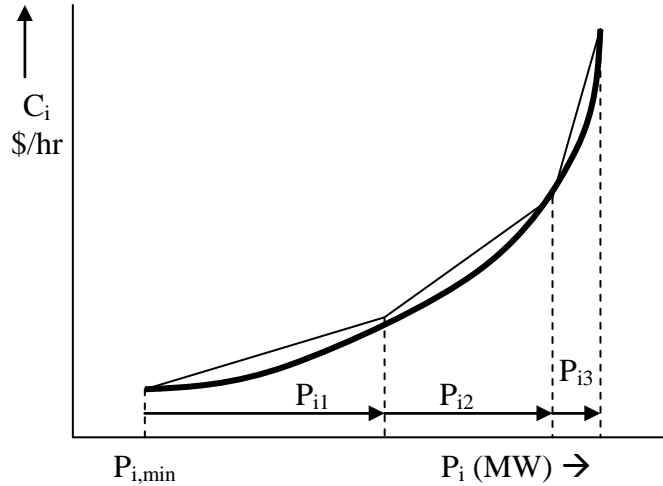


Fig. 2: Typical Cost Curve

In Fig. 2, the variables P_{i1} , P_{i2} , and P_{i3} represent generation *increments* that range from 0 to some maximum value $P_{i1,max}$, $P_{i2,max}$, and $P_{i3,max}$, respectively.

It is important to understand what P_{ik} is NOT. It is not the generation value itself.

Thus, we have that

$$0 \leq P_{ik} \leq P_{ik,max}$$

and

$$P_i = P_{i,min} + P_{i1} + P_{i2} + P_{i3}$$

Let's denote the slope of each one of the line segments as s_{i1} , s_{i2} , and s_{i3} , respectively, or s_{ik} in general. Then the increment in cost function C_i corresponding to each line segment is given by

$$\Delta C_i = s_{ik} P_{ik}.$$

This relation is illustrated in Fig. 3 for the first line segment.

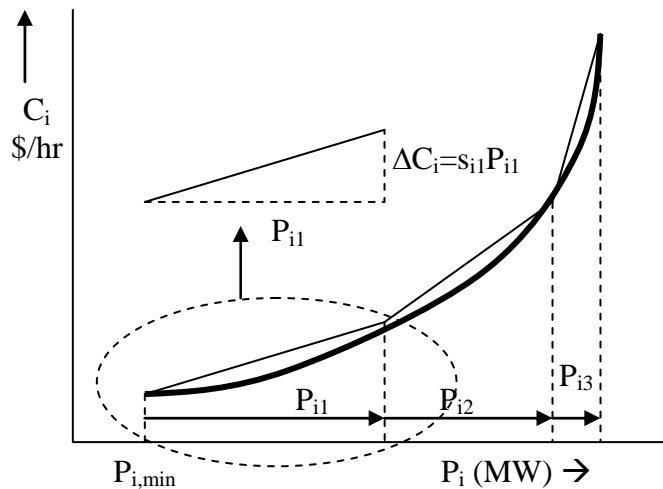


Fig. 3: Relation between P_{ik} and Cost Function for Segment 1

So the cost function can be approximated using the line segments, and that approximation may be improved to any desired level by increasing the number of line segments used.

Let's denote the approximate cost function as K_i , which can be expressed as a function of the new variables P_{ik} , according to:

$$K_i(P_{i1}, P_{i2}, P_{i3}) = C_i(P_{i,\min}) + S_{i1}P_{i1} + S_{i2}P_{i2} + S_{i3}P_{i3}$$

This function is clearly linear in the variables.

Example 1: Consider a three unit system with the following cost curves expressed as quadratics, and associated minimum and maximum generation ranges. Convert each cost curve to a piecewise linear expression using three variables per curve.

$$C_1(P_1) = 0.00533P_1^2 + 11.669P_1 + 213.1, \\ 50 \leq P_1 \leq 200$$

$$C_2(P_2) = 0.00889P_2^2 + 10.333P_2 + 200 \\ 37.5 \leq P_2 \leq 150$$

$$C_3(P_3) = 0.00741P_3^2 + 10.833P_3 + 240 \\ 45 \leq P_3 \leq 180$$

These cost curves are illustrated in Fig. 4.

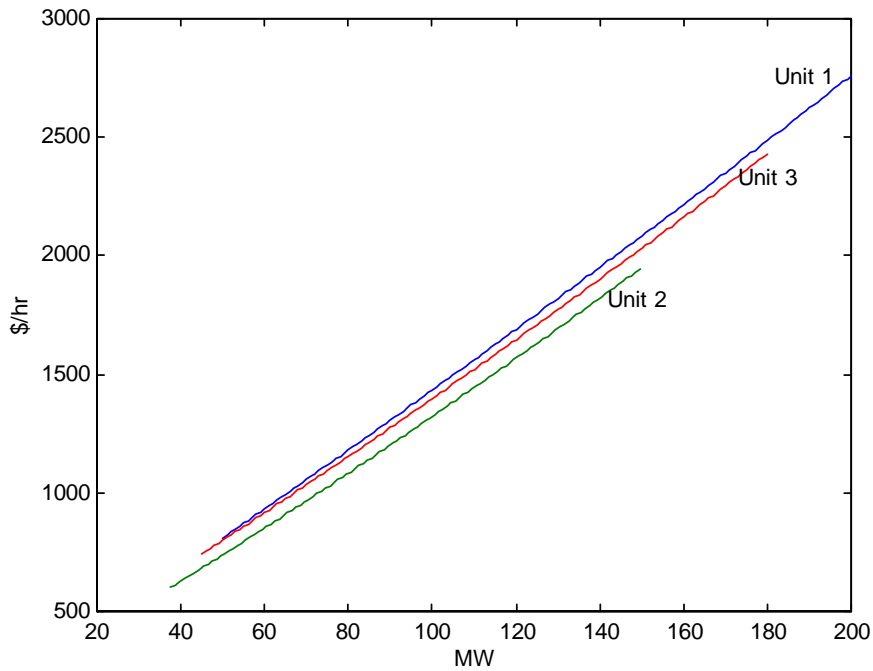


Fig. 4: Cost curves for example 1

The three cost curves appear linear in Fig. 4. This is because in the given range, the linear term dominates the quadratic term. If we plot these functions for a wider range of power generation level, their nonlinearity is more pronounced, as indicated in Fig. 5.

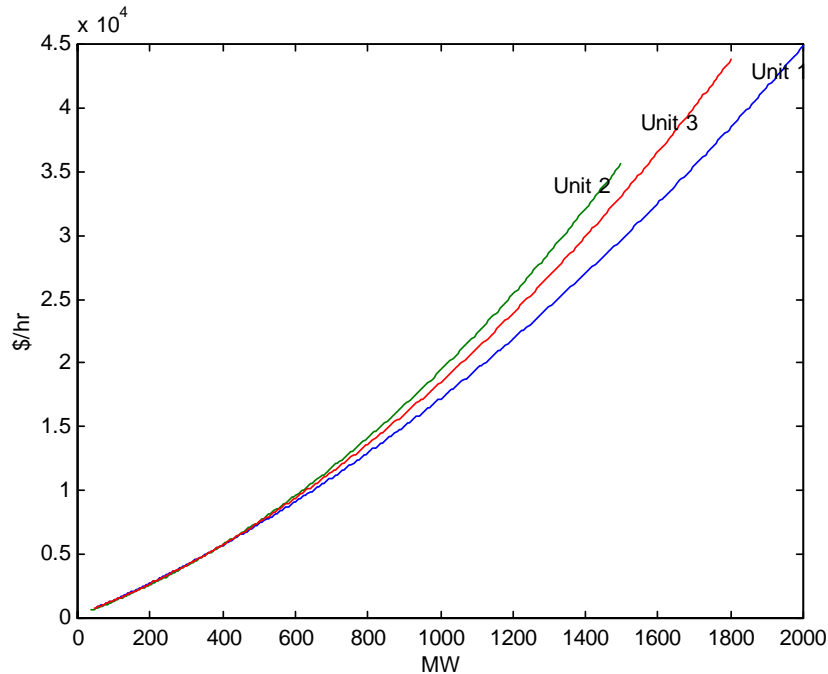


Fig. 5: Cost curves of example 1 over an expanded operating range

Given the restriction on operating range, one might consider computing a single slope for each cost curve, and this is sometimes done as an approximation. But, it is definitely less accurate to do so, and over time, costs introduced by the suboptimal solutions can be significant.

One can observe non-negligible change in slope of these three curves through computation of those slopes.

We select break points of these three curves according to Table 1.

Table 1: Break points for PW linear approximation of cost curves given as MW, Cost

Unit	BP#1 (unit min)	BP#2	BP#3	BP#4 (unit max)
1	50.0, 809.9	100, 1433	160, 2217	200, 2760
2	37.5, 5325	70, 9699	130, 1693	150, 1950
3	45.0, 742.5	90, 1275	140, 1902	180, 2430

Now we compute the slopes for each of the three segments 1-2, 2-3, and 3-4 for each curve. For example,

$$s_{11} = (1433 - 809.91) / (100 - 50) = 12.46$$

The results of these calculations are given in Table 2.

Table 2: Slopes of piecewise linear approximations of cost curves

Unit, i	s_{i1}	s_{i2}	s_{i3}
1	12.46	13.07	13.58
2	11.29	12.11	12.82
3	11.83	12.54	13.20

3.0 The objective function

The approximate piecewise linear cost curves for the three units are given below:

$$K_1(P_{11}, P_{12}, P_{13}) = C_1(P_{1,\min}) + s_{11}P_{11} + s_{12}P_{12} + s_{13}P_{13}$$

$$K_2(P_{21}, P_{22}, P_{23}) = C_2(P_{2,\min}) + s_{21}P_{21} + s_{22}P_{22} + s_{23}P_{23}$$

$$K_3(P_{31}, P_{32}, P_{33}) = C_3(P_{3,\min}) + s_{31}P_{31} + s_{32}P_{32} + s_{33}P_{33}$$

The objective function of our linear program is then given as the sum of these approximated cost curves. However, in minimizing this function, we recognize that the first terms, $C_i(P_{i,\min})$, have no influence on the values of the decision variables in the LP since they are constant. So we will drop them in the LP formulation. However, we should be careful that in expressing the value of the objective function at the optimal dispatch (which is the total cost rate at the optimal dispatch), we should include these terms back in.

Note carefully that $C_i(P_{i,\min})$ is NOT the constant term in the quadratic expression but rather it is the function C_i evaluated at $P_{i,\min}$. This would be $C_1(50)=796.55$, $C_2(37.5)=600$, $C_3(45)=742.49$, so we add $796.55+600+742.49=2139.04$ to the objective function of the linear program once we solve it.

We denote the objective function of the linear program as Z :

$$\begin{aligned} Z = & s_{11}P_{11} + s_{12}P_{12} + s_{13}P_{13} \\ & + s_{21}P_{21} + s_{22}P_{22} + s_{23}P_{23} \\ & + s_{31}P_{31} + s_{32}P_{32} + s_{33}P_{33} \end{aligned}$$

Substitution of the numerical values of the s_{ik} 's results in:

$$\begin{aligned} Z = & 12.46P_{11} + 13.07P_{12} + 13.58P_{13} \\ & + 11.29P_{21} + 12.11P_{22} + 12.82P_{23} \\ & + 11.83P_{31} + 12.54P_{32} + 13.20P_{33} \end{aligned}$$

In matrix form, this is

$$Z = [12.46 \quad 13.07 \quad 13.58 \quad 11.29 \quad 12.11 \quad 12.82 \quad 11.83 \quad 12.54 \quad 13.20] \begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \\ P_{21} \\ P_{22} \\ P_{23} \\ P_{31} \\ P_{32} \\ P_{33} \end{bmatrix}$$

$$= \underline{c}^T \underline{x}$$

where \underline{c} and \underline{x} are given below:

$$\underline{c} = \begin{bmatrix} 12.46 \\ 13.07 \\ 13.58 \\ 11.29 \\ 12.11 \\ 12.82 \\ 11.83 \\ 12.54 \\ 13.20 \end{bmatrix}, \quad \underline{x} = \begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \\ P_{21} \\ P_{22} \\ P_{23} \\ P_{31} \\ P_{32} \\ P_{33} \end{bmatrix}$$

4.0 Inequality Constraints

The original inequality constraints are
 $50 \leq P_1 \leq 200$, $37.5 \leq P_2 \leq 150$, $45 \leq P_3 \leq 180$

However, in order to model the cost curves as piecewise linear, we had to utilize 9 variables. So we need to convert the above constraints on the three generation levels to constraints on the 9 new variables.

Lower Bounds: It should be recognized that all of the new variables must be non-negative. This comes from the fact that we cannot have a negative amount of generation increment. But we can have any amount of generation increment between zero and the upper bound. So the lower bound on all 9 variables is zero.

Upper bounds: The upper bounds on the variables are the maximum increment possible for the corresponding linear segment. The data in Table 1, repeated below for convenience, is helpful here. For example, the first segment of unit 1 has range from 50 to 100 MW, so the upper bound on P_{11} is $100-50=50$. Likewise, the second segment of unit 1 has range 100 to 160, so the upper bound on P_{12} is $160-100=60$.

Table 1: Break points for PW linear approximation of cost curves given as MW, Cost

Unit	BP#1 (unit min)	BP#2	BP#3	BP#4 (unit max)
1	50.0, 809.9	100, 1433	160, 2217	200, 2760
2	37.5, 5325	70, 9699	130, 1693	150, 1950
3	45.0, 742.5	90, 1275	140, 1902	180, 2430

Continuing in this fashion, we obtain the upper bounds for all of the 9 decision variables according to the below:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \\ P_{21} \\ P_{22} \\ P_{23} \\ P_{31} \\ P_{32} \\ P_{33} \end{bmatrix} \leq \begin{bmatrix} 50 \\ 60 \\ 40 \\ 32.5 \\ 60 \\ 20 \\ 45 \\ 50 \\ 40 \end{bmatrix}$$

5.0 Equality Constraint

We denote the total load plus losses to be supplied as $LOAD$. The original equality constraint is that the sum of the generation must equal the total load plus losses to be supplied, i.e.,

$$P_1 + P_2 + P_3 = LOAD$$

It may seem that the sum of the new 9 variables should also equal the total load plus losses to be supplied, but this is not quite true. The reason why is that the new 9 variables are *increments*, and therefore do not include the minimum generation level of each unit.

Therefore, we must subtract the total minimum generation level from the total load plus losses to be supplied. So the equality constraint, in terms of the new 9 variables, is:

$$\begin{aligned} P_{11} + P_{12} + P_{13} + P_{21} + P_{22} + P_{23} + P_{31} + P_{32} + P_{33} \\ = \text{LOAD} - (P_{1,\min} + P_{2,\min} + P_{3,\min}) \end{aligned}$$

We can model this in matrix form as:

$$\begin{bmatrix}
 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{bmatrix}
 \begin{bmatrix}
 P_{11} \\
 P_{12} \\
 P_{13} \\
 P_{21} \\
 P_{22} \\
 P_{23} \\
 P_{31} \\
 P_{32} \\
 P_{33}
 \end{bmatrix}
 =
 \begin{bmatrix}
 LOAD - \left(\sum_{i=11}^3 P_{i,\min} \right) \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0
 \end{bmatrix}$$

6.0 Solution by Matlab

The linear programming function call in Matlab is “linprog.” You can do “help linprog” to learn about it, per the following:

help linprog

LINPROG Linear programming.

X=LINPROG(f,A,b) solves the linear programming problem:

$$\min_x f^*x \quad \text{subject to:} \quad A*x \leq b$$

X=LINPROG(f,A,b,Aeq,beq) solves the problem above while additionally satisfying the equality constraints $Aeq*x = beq$.

X=LINPROG(f,A,b,Aeq,beq,LB,UB) defines a set of lower and upper bounds on the design variables, X, so that the solution is in the range $LB \leq X \leq UB$. Use empty matrices for LB and UB if no bounds exist. Set $LB(i) = -\text{Inf}$ if X(i) is unbounded below;

set $UB(i) = \text{Inf}$ if $X(i)$ is unbounded above.

$X = \text{LINPROG}(f, A, b, Aeq, beq, LB, UB, X0)$ sets the starting point to $X0$. This option is only available with the active-set algorithm. The default interior point algorithm will ignore any non-empty starting point.

$X = \text{LINPROG}(f, A, b, Aeq, Beq, LB, UB, X0, \text{OPTIONS})$ minimizes with the default optimization parameters replaced by values in the structure OPTIONS , an argument created with the OPTIMSET function. See OPTIMSET for details. Use options are Display , Diagnostics , TolFun , LargeScale , MaxIter . Currently, only 'final' and 'off' are valid values for the parameter Display when LargeScale is 'off' ('iter' is valid when LargeScale is 'on').

$[X, \text{FVAL}] = \text{LINPROG}(f, A, b)$ returns the value of the objective function at X :
 $\text{FVAL} = f' * X$.

$[X, \text{FVAL}, \text{EXITFLAG}] = \text{LINPROG}(f, A, b)$ returns EXITFLAG that describes the exit condition of LINPROG .

If EXITFLAG is:

- > 0 then LINPROG converged with a solution X .
- 0 then LINPROG reached the maximum number of iterations without converging.
- < 0 then the problem was infeasible or LINPROG failed.

$[X, \text{FVAL}, \text{EXITFLAG}, \text{OUTPUT}] = \text{LINPROG}(f, A, b)$ returns a structure OUTPUT with the number of iterations taken in OUTPUT.iterations , the type of algorithm used in OUTPUT.algorithm , the number of conjugate gradient iterations (if used) in $\text{OUTPUT.cgiterations}$.

$[X, \text{FVAL}, \text{EXITFLAG}, \text{OUTPUT}, \text{LAMBDA}] = \text{LINPROG}(f, A, b)$ returns the set of Lagrangian multipliers LAMBDA , at the solution: LAMBDA.ineqlin for the linear inequalities A , LAMBDA.eqlin for the linear equalities Aeq , LAMBDA.lower for LB , and LAMBDA.upper for UB .

NOTE: the LargeScale (the default) version of LINPROG uses a primal-dual method. Both the primal problem and the dual problem must be feasible for convergence. Infeasibility messages of either the primal or dual, or both, are given as appropriate. The primal problem in standard form is

$$\min f' * x \text{ such that } A * x = b, x \geq 0.$$

The dual problem is

$$\max b' * y \text{ such that } A' * y + s = f, s \geq 0.$$

The code to solve this problem using Matlab is provided below.

```
%Load is system load plus losses
Load=217.87;
%summingen is the sum of minimum generation levels
summingen=50+37.5+45;

%Build objective function vector.
c=[12.46 13.07 13.58 11.29 12.11 12.82 11.83 12.54 13.20]';

%We will not use A or b but will instead use LB and UB.
A=[];
b=[];

%Build Aeq matrix for equality constraints. Since there is only one
%equality constraint, only the top row has non-zero elements. All of these
%elements are "1" because the constraint is
%      sum of variables=load-sum of minimum generation
Aeq=zeros(9);
Aeq(1,:)=1;

%Build right-hand side of equality constraint. It will be vector of zeros
%except for element in first row, which is load-sum of minimum generation
beq=zeros(9,1);
beq(1)=Load-summingen;

%Build upper and lower bounds on decision variables.
LB=[0 0 0 0 0 0 0 0 0]';
UB=[50 60 40 32.5 60 20 45 50 40]';
[X,FVAL,EXITFLAG,OUTPUT,LAMBDA]=LINPROG(c,A,b,Aeq,beq,LB,UB);
```

Note that in this code, the inequalities were entirely modeled using the LB and UB facilities. The problem could also have been solved using A and b to model the upper bounds and LB to model the lower bounds.

Solution to this problem is given by just typing on the Matlab command line “X” and “FVAL.” This solution is given below:

$$\begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \\ P_{21} \\ P_{22} \\ P_{23} \\ P_{31} \\ P_{32} \\ P_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 32.5 \\ 7.87 \\ 0 \\ 45 \\ 0 \\ 0 \end{bmatrix}, \quad Z=994.5807$$

The actual unit generation levels are:

$$P_i = P_{i,\min} + P_{i1} + P_{i2} + P_{i3}$$

With minimum generation levels at units 1, 2, and 3 of 50, 37.5, and 45, respectively, the unit generations are:

$$P_1=50+0+0+0=50 \text{ MW}$$

$$P_2=37.5+32.5+7.87+0=77.87 \text{ MW}$$

$$P_3=45+45+0+0=90 \text{ MW}$$

Remembering that we must add to Z (the objective function of the linear program) the sum of the cost functions evaluated at the minimum generation level, which is $C_1(50)+C_2(37.5)+C_3(45)$ and was determined above to be 2153.855, we find the total cost rate at the optimum dispatch for the 217.87 loading is $2153.855+994.5807=3148.44$ \$/hr. This solves the problem ☺.

7.0 Lagrange multipliers

It is of interest to examine the Lagrange multipliers from the solution returned by the “linprog” function in matlab. The structure called LAMBDA contains the Lagrange multipliers (the dual variables) for each constraint. Specifically,

- LAMBDA.ineqlin contains the Lagrange multipliers for the linear inequalities A,
- LAMBDA.eqlin contains the Lagrange multipliers for the linear equalities Aeq,
- LAMBDA.lower contains the Lagrange multipliers for LB,
- LAMBDA.upper contains the Lagrange multipliers for UB.

Since we did not utilize the constraints associated with A in our Matlab solution, LAMBDA.ineqlin is empty.

But LAMBDA.eqlin, the multipliers for the equality constraint, contains the vector:

-12.1100

0

0

0

0

0

0

0

0

The multipliers for equality constraints 2-9 are zero since we only had one constraint. The multiplier for the first equality constraint is -12.11. The interpretation of this is that, if we *increase* the RHS of this equality constraint by 1.0, we will see the objective improve by -12.11. Since we are solving a minimization problem, this means that if we increase the load from 217.87, to 218.87, the objective function (Z) will improve by -12.11. Since the multiplier is negative, it means that the total cost will actually go up.

We can easily check this with Matlab. Changing the value of LOAD from 217.87 to 218.87 and resolving, we find that the new value of Z is 1006.7. Subtracting this from the old value of Z (=994.58), we get $994.58 - 1006.7 = -12.12$. This Lagrange multiplier is called the “system lambda” in the electric power industry.

Likewise, examination of the Lagrange multipliers on the inequality constraints through LAMBDA.lower and LAMBDA.upper show the following:

LAMBDA.lower

ans =

0.3500
0.9600
1.4700
0.0000
0.0000
0.7100
0.0000
0.4300
1.0900

This indicates that P_{11} , P_{12} , P_{13} , P_{23} , P_{32} , and P_{33} are all at their lower limit of 0, and they show us also the amount of improvement in the objective function if we *decreased* the right-hand-side of the corresponding constraint by 1.0.

Note that the Lagrange multiplier for the equality constraint indicates objective function improvement for increase in RHS, whereas the Lagrange multiplier for these lower bound inequality constraints indicate objective function improvement for decrease in RHS. The reason is this:

All constraints must be in the form: $x \leq b$. When they are in this form, then the corresponding Lagrange multipliers give objective function improvement for increase in RHS.

But the lower bound inequality constraints are in the form: $b \leq x$. To put them in standard form, we multiply both sides by -1 to get: $-b \geq -x$ or $-x \leq -b$. In this form, the corresponding Lagrange multipliers give objective function improvement for increase in RHS. But an increase to $-b$ is a decrease to b .

Observe the contents of LAMBDA.upper

ans =

0.0000

0.0000

0.0000

0.8200

0.0000

0.0000

0.2800

0.0000

0.0000

HW # 11: Repeat this problem for LOAD=400 MW using LP. Give generation levels, objective function value, and Lagrange multipliers. Determine a test to see if the Lagrange multipliers are correct, and implement your test on the Lagrange multipliers corresponding to the equality constraint and to one binding inequality constraint.

References

1. A. J. Wood and B. F. Wollenberg, Power, Generation, Operation and Control, second edition, John Wiley & Sons, New York, NY, 1996.