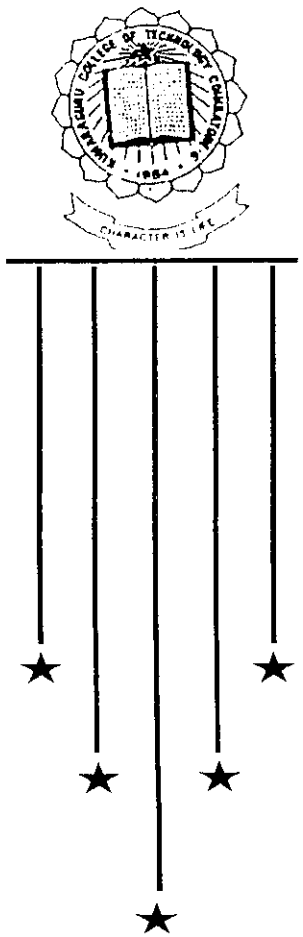
EVALUATION OF EQUIVALENT COST FUNCTION AND INTERCHANGE EVALUATION IN INTERCONNECTED POWER SYSTEM - A DIRECT METHOD

Project Report



Submitted by

S.Kavitha
B.Dilip Kaartic
L.Jeya Sudar
M.S.Suresh Kumar

Under the Guidance of

Miss.P.K.Preetha,M.Tech.,

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE

BACHELOR OF ENGINEERING IN ELECTRICAL AND ELECTRONICS ENGINEERING

OF THE BHARATHIAR UNIVERSITY

1998 - 99

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

Coimbatore - 641 006.

Department of Electrical and Electronics Engineering

Kumaraguru College of Technology

Coimbatore 641 006.

CERTIFICATE

This is to certify that the report entitled

Evaluation Of Equivalent Cost Function And Interchange Evaluation In Interconnected Power System - A Direct Method

has been submitted by

S. KAVITHA
B. DILIP KAARTIC
L. JEYA SUDAR
M.S. SURESH KUMAR

in partial fulfilment for the award of Bachelor of Engineering in Electrical and Electronics Engineering ,Bharathiyar University. Coimbatore - 641 006 during the academic year 1998 - 1999

Guide

Date

Head of the department

PROFESSOR & HEAD
Dept. of Electrical & Electronics Engg.
Kumaraguru College of Tech., Coimbatore-

Date

Certified that the candidate with University Register No..... was examined in project work Viva-Voce Examination held on.....

Internal Examiner

External Examiner

DEDICATED TO
OUR BELOVED PARENTS

Acknowledgment

We express our heart felt gratitude to our guide **Miss. P.K. Preetha, M.Tech.,** Lecturer in Electrical and Electronics Engineering Department for her able guidance and encouragement

We are forever indebted to **Dr.K.A. Palaniswamy, B.E., M.Sc.(engg), Ph.D., M.I.S.T.E, C.Eng.(I), F.I.E.,** Professor and Head of the Department of Electrical and Electronics Engineering for his constant encouragement.

We express our sincere thanks to **Dr. K.K. Padmanabhan, B.Sc., (Engg.) M.Tech., Ph.D.,** Principal, Kumaraguru college of Technology for his kind patronage.

We acknowledge the dynamic support of all our friends who helped us at every stage of this project. Last but not the least, we thank all the staff and technicians of Electrical and Electronics Engineering for their kind support and help.

SYNOPSIS

The price of electrical power has a direct bearing on the economy of every nation. If it is higher the cost of every item (i.e.) from industrial components to consumer goods of all types ought to be high. The cost of electricity in turn depends upon the expenditure involved in its production, and distribution. In such a scenario Engineers and Scientists all over the world strive hard for evolving better ways ad mans for reducing the cost of Electrical power to the maximum extent possible.

In this project a specific aspect of achieving economy in power generation is dealt with i.e.., by interconnecting networks of power utilities.

Chief objective of this project work are

1. To develop the equivalent generation cost for an interconnected system

2. To develop a direct method for economic dispatch problem

3. To reduce the computational work and time involved in conventional interchange evaluation method.

A software in 'C' language is developed to achieve the above objectives and simulation results are presented in this report.

Chapter - 1

Introduction

Interconnection of network of power utilities has become the order of the day, because of multifarious advantages derived from such an interconnection. Interconnected electric power system is more reliable, convenient to operate and offers economical operating cost. Apart from this, it offers better regulation characteristics since a load change in any of the area is taken care by all units in the interconnection. The loss of a generating unit in one of the areas can be made up from the spinning reserve. On the contrary, if an isolated power system, were to loose a large unit, the chance of the other units, being able to make up the deficit are greatly reduced. To boost up the spinning reserve, extra units would have to be run, which results in poor economy.

The most important reason for interconnecting with neighbouring systems is the better economics of operation that can be attained. When two areas A and B are operating at different incremental costs, there is a possibility to improve the overall economy.

2.1 Economic Dispatch Problem

The economic load dispatch problem involves the solution of two different problems. The first of these is the unit commitment or predispatch problem wherein it is required to select optimally out of the available generating sources to operate, to meet the expected load and provide a specified margin of operating reserve over a specified period of time. The second aspect of economic dispatch is the on-line economic dispatch where in it is required to distribute the load among the generating units actually paralled with the system in such manner so as to minimize the total cost of supplying the minute-to minute requirements of the system.

The economic dispatch problem is defined as

$$\text{Min } F_T = \sum_{i=1}^{n} F_i \quad \text{----------} \quad (2.1)$$

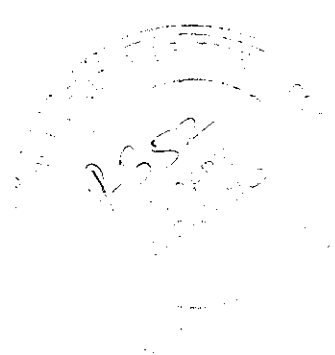
$$\text{Subject to } \sum_{i=1}^{n} P_i \quad \text{----------} \quad (2.2)$$

Where F_T is the total fuel input to the system

F_i - fuel input to i^{th} unit

P_D - the total load demand

P_i - the generation of i^{th} unit



By making use of lagrangian multiplier the auxillary function is obtained as

$$F = F_T + \lambda\,(P_D - \sum_{i=1}^{n} P_i) \quad \text{---------- (2.3)}$$

Where λ is the Lagragian multiplies

Differentiating f with respect to the generation Pi and equating to zero gives the condition for optimal operation of the system.

$$\frac{\partial F}{\partial P_i} = \frac{\partial F_T}{\partial P_i} + \lambda(0-1) = 0$$

$$\frac{\partial F_T}{\partial P_i} - \lambda = 0$$

Since $F_T = F_1 + F_2 + \ldots.. Fn$:

$$\frac{\partial F_T}{\partial P_i} = \frac{dF_i}{dP_i} = \lambda \qquad \text{----------} \quad (2.4)$$

and therefore the condition for optimum operation is

$$\frac{\partial F1}{\partial P2} = \frac{\partial F2}{\partial P2} = \frac{\partial Fi}{\partial Pi} = \lambda \qquad \text{------} \quad (2.5)$$

Here $\frac{\partial Fi}{\partial Pi}$ = incremental production cost of plant i in Rs

$$\frac{\partial P2}{P_{i\min}} \quad \frac{\partial P2}{\leq Pi \leq} \quad \frac{\partial Pi}{P_{i\max}} \qquad \text{----------} \quad (2.6)$$

2.2 Development of the Technique

A sample power system is assumed to have m plants which in turn having n generating units. Assume that the cost functions are quadratic in nature, the cost functions of the units of ith plant are given as

$$Fij = aijPij^2 + bijPij + Cij \quad \text{---------- (2.7)}$$

i = 1, 2 .....m

j = 1, 2, ...n

Where the suffix i stands for the plant and j stands for the units in that plant. aij, bij and cij are the cost Co-effiecients. Fij denotes the cost function of an unit and Pij denotes unit generation. The inequality capacity constraints of the units are

$$Pij_{min} < Pij < Pij_{max}$$

When a known load P_{Di} comes on plant i, then this load should be shared by the individual units in an economic manner. Assuming all units are put into service, the operating costof plant i will be minimum when all its units are operating and equal incremental production cost given by

$$\frac{df_{ij}}{dP_{ij}}, j = 1,2...n \text{ ----------} \qquad (2.8)$$

Substituting Fij from eqn (2.7) in eqn (2.8)

$$2ai1Pi1 + bi1 = \lambda \quad \text{--------} \quad (2.9)$$

$$2ai2Pi2 + bi2 = \lambda \quad \text{--------} \quad (2.10)$$

$$2ainPin + bin = \lambda \quad \text{--------} \quad (2.11)$$

Equating equations (2.9) and (2.10)

$$2ailPil + bil = 2ai2\ Pi2 + bi2 \quad \text{----------} \quad (2.12)$$

Rearranging similar terms

$$Pi2 = \frac{ai1}{ai2} Pi1 + \frac{(bil - bi2)}{2ai2}$$

$$= \alpha_{i2} P_{i1} + \beta_{i2} \quad \text{----------} \quad (2.13)$$

In general

$$Pij = \frac{ai1}{aij} Pi1 + \frac{bil - bij}{2aij}$$

$$Pij = \alpha ij Pi1;\ j = 2,3...n \quad \text{----------} \quad (2.14)$$

where $\alpha ij = \frac{ai1}{aij}$ and $\beta ij = \frac{bi1 - bij}{2aij}$

$\alpha_{i1} = 1$ and $\beta_{i1} = 0$

The Power balance equation for plant i is

$$\sum_{j=1}^{n} Pij = P_{D1}$$

Where P_{Di} = Power demand in Plant i

$$Pi1 + Pi2 + ... Pin = PDi \quad \text{----------} \quad (2.15)$$

Substituting the values of Pij from eqn (2.14) in eqn (2.15)

$$\alpha i1 Pi1 + \beta i1 + \alpha i2 Pi1 + \beta_{i2} + ... \alpha in P_{i1} + \beta i_n = P_{Di}$$

$$(\alpha_{i1} + \alpha_{i2} + ... \alpha_{in}) P_{i1} + (\beta_{i1} + \beta_{i2} + ... \beta_{in}) = P_{Di}$$

This equation can be written as

$$\alpha_i P_{i1} + \beta_i = P_{Di}$$

Where $\alpha_i = 1 + \varepsilon_{i2} + \alpha_{i3} + ... \alpha_{in}$

$$\beta_i = \beta_{i2} + \beta_{i3} + ... \beta_{in}$$

$$P_{i1} = \frac{P_{Di} - \beta_i}{\alpha_i}$$

Substituting Pi1 from eqn (2.16) in eqn (2.14) leads to

$$P_{ij} = \alpha_{ij} \frac{(P_D i - \beta_i)}{\alpha_i} + \beta_{ij}; j = 1,2....n \quad \text{----------} \quad (2.17)$$

This equation can be written as

$$P_{ij} = A_{ij} P_{Di} + B_{ij}; j = 1,2,....n$$

Where $Aij = \frac{\alpha_{ij}}{\alpha_{ij}}$ and

$$B_{ij} = \frac{-\alpha_{ij}\beta_i}{\alpha_i} + \beta_{ij}$$

$$A_{i1} = \frac{1}{\alpha_i} \text{ and } B_{i1} = \frac{-\beta_i}{\alpha_i}$$

Substituting the unit generations, Pij in terms of plant load PDi from eqn (2.18) in the unit cost function eqn (2.7)

$$F_{ij} = a_{ij} (A_{ij} P_{Di} + B_{ij})^2 + b_{ij} (A_{ij}P_{Di} + B_{ij}) + C_{ij} \quad \text{----------} \quad (2.19)$$

i = 1, 2, ... m

j = 1, 2, ...n

Regrouping similar terms

$$F_{ij} = a_{ij} Aij^2 P_{Di}^2 + (2a_{ij} A_{ij} B_{ij} + b_{ij}A_{ij})P_{Di} + (a_{ij}B_{ij}^2 + b_{ij} B_{ij} + C_{ij})$$

$$F_{ij} = X_{ij} P_{Di}^2 + y_{ij}P_{Di} + Z_{ij} \quad \text{--------} \quad (2.20)$$

Where Xij = aijAij

$$Y_{ij} = 2a_{ij}A_{ij}B_{ij} + b_{ij}A_{ij}$$

$$Z_{ij} = a_{ij} B_{ij}^2 + b_{ij}B_{ij} = C_{ij}$$

Therefore total operating cost of plant i is

$$Fti = \sum_{j=1}^{n} Fij$$

$$= \left[\sum_{j=1}^{n} Xij\right] PDi^2 + \left[\sum_{j=1}^{n} Yij\right] PDi + \left[\sum_{j=1}^{n} Zij\right]$$

$$Fti = X_i P_{Di}^2 + Y_i P_{Di} + Z_i \; i = 1,2...m$$

Where $$Xi = \sum_{j=1}^{n} Xij$$

$$Yi = \sum_{j=1}^{n} Yij$$

$$Zi = \sum_{j=1}^{n} Zij$$

But the total load P_{Di} on Plant i should be exactly equal to the total generation P_{Gi} of plant i. Hence eqn (2.21) can be written as

$$Fti = X_i P_{Gi}^2 + Y_i P_{Gi} + Z_i;$$

$$i = 1, 2 \ldots m$$

Eqn (2.22) represents the equivalent cost function of plant i. It gives the total cost for any generation (load)

2.3 Optimal sharing of load among plants

A sample powersystem with m generating plants each having n generating units is considered. Let P_D be the total system load demand. Even though each plant is containing a number of generating units in parallel, they are treated as if made of a single equivalent unit (plant) whose cost functions are given by eqn (2.22). The total system load P_D should be shared by the individual plants in an economic manner. Assuming all plants are put into service, the total operating cost will be minimum when all the plants are operating at equal incremental production cost, λ' given by

$$\frac{df_{ti}}{dP_{Gi}}; i = 1,2,.....m \quad \text{----------} \quad (2.23)$$

Where Fti = total operating cost ofplant i

P_{Gi} = total generation of plant i

Substituting Fti from eqn (2.22) in eqn (2.23)

$$2X_1P_{G1} + Y_1 = \lambda' \quad \text{----------} \quad (2.24)$$

$$2X_2P_{G2} + Y_2 = \lambda' \quad \text{----------} \quad (2.25)$$

$$2X_mP_{Gm} + Y_m = \lambda' \quad \text{----------} \quad (2.26)$$

Equating eqn's (2.24) and (2.25)

$$2X_1P_{G1} + Y_1 = 2X_2P_{G2} + Y_2$$

Rearranging

$$P_{G2} = \frac{X_1}{X_2} P_{G1} + \frac{Y_1 - Y_2}{2X_2}$$

$$= \gamma_2 P_{G1} + \delta_2$$

In general

$$P_{G1} = \frac{X_1}{X_1} P_{G1} + \frac{Y_1 - Y_1}{2X_i}$$

$$\gamma i P_{G1} + \delta i; i = 1,2.....m \qquad \text{----------(2.27)}$$

system power balance equation is

$$\sum_{i=1}^{m} P_{Gi} - P_D = 0$$

$$P_{G1} + P_{G2} + ... P_{Gm} = P_D \qquad \text{----------(2.28)}$$

Where P_{Gm} represents the generation of m^{th} plant and $P_{D'}$ the total system demand.

Substituting the value of P_{Gi} from eqn (2.27) in eqn (2.28)

$$\gamma_1 P_{G1} + \delta_1 + \gamma 2 P_{G1} + \delta_2 + ... \gamma n P_{G1} + \delta_n = P_D$$

$$(1 + \gamma_2 + ... \gamma_n) P_{G1} + (\delta_d ..... \delta_n) = P_D$$

This equation can be written as

$$P_{G1} = \frac{P_D - \delta}{\gamma} \qquad \text{----------} \quad (2.29)$$

Substituting PG1 from eqn (2.27) in eqn (2.25)

$$P_{Gi} = \gamma_i \frac{(P_D - \delta)}{\gamma} + \delta_i$$

$$P_{Gi} = L_i P_D + M_i ; \qquad \text{--------} \quad (3)$$

i = 1,2, 3, ... m

Where $L_i = \frac{\gamma_i}{\gamma}; M_i = \frac{-\delta_i \delta}{\gamma} + \delta_i$

Li and Mi are known constants for the given system. Siince these constants determine the division of load P_D among participating plants, these are called participating factor of plants.

Substituiting eqn (3) in equivalent cost function of plants, we get,

$$F_t = X_{PD}^2 + Y_{PD} + Z \qquad \text{----------} \quad (3.1)$$

Where Ft denotes total operating cost of interconnected system

$= \Sigma F_{ti}$

X,Y,Z are the cost coefficients of the system. These can be expressed in terms of plant coefficients Eqn (3.1) represents the equivalent generation cost function of the interconnected system.

2.4 Optimisation Algorithm

1. The unit cost function co-efficients aij, bij and cij, the inequality capacity constraints $P_{ij\ min}$, $P_{ij\ max}$ and the system demand P_D are read in.

2. The equivalent cost function co-efficients Xi, Yi, Zi of equivalent plants/areas in accordance with eqn (2.20) are computed

3. The participation factors Li & Mi of plants or plant equivalent of areas are computed using eqn (2.28)

4. Total system demand, P_D is substituted in eqn (2.28) to get plant/area generation P_{Gi}.

a) If any of the plant generation PGi is less than the plants minimum generation capacity limit, PGi < PGi min, then set that PGi = 0. Recalculate the paraticipating factors Li and Mi taking the units Xi = Yi = Zi = 0 Goto step 4

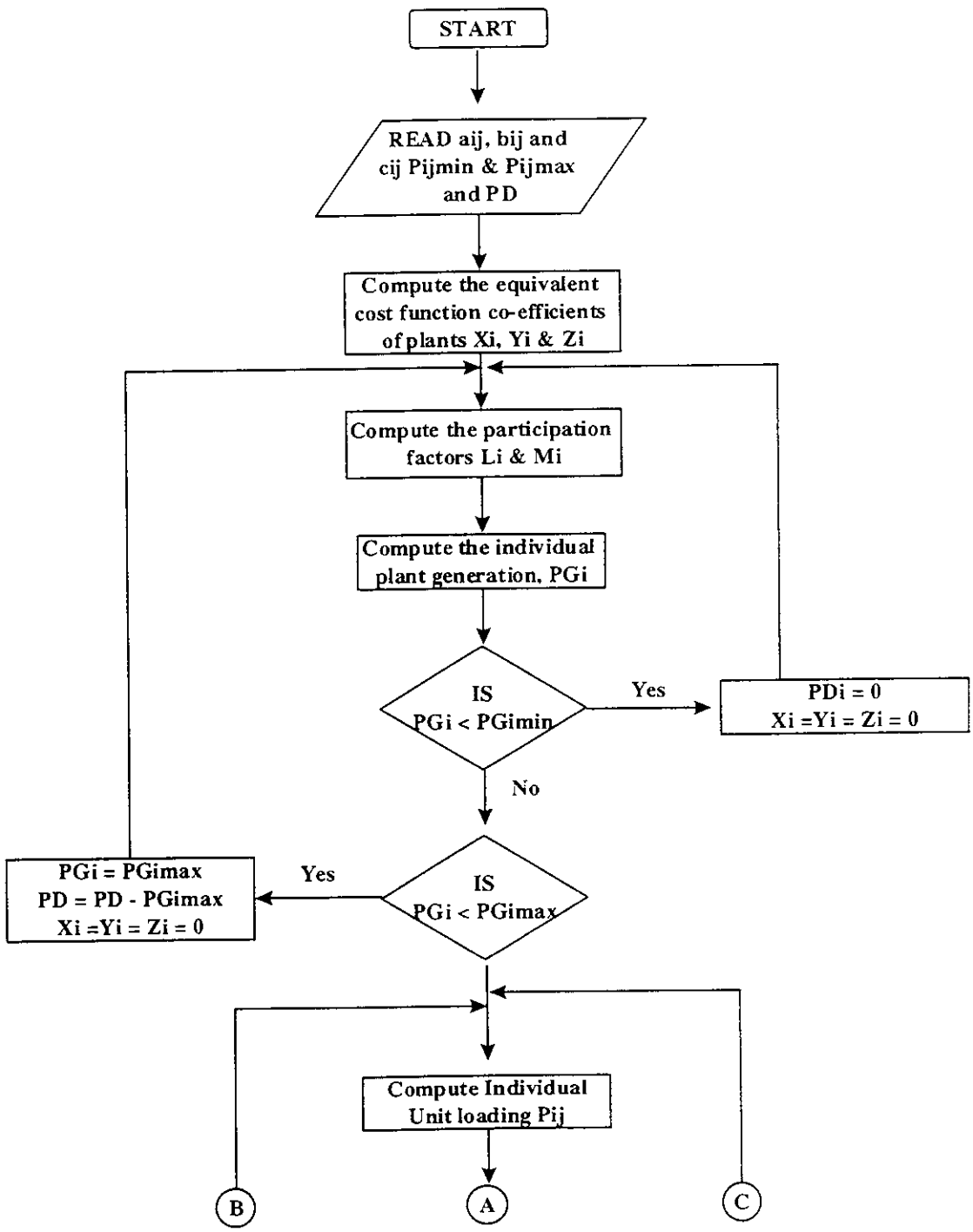
b) On the other hand, if any of the plant generation PGi > PGimax, then PGi = PGimax and PD = PD - PGi max are set and Xi, Yi, and Zi are made equal to zero while evolving Li and Mi using eqn (2.28) and step 4 repeated.

5) The individual unit loadings Pij are computed using Eqn (2.18) taking PDi = PGi

a) If any of the unit generation Pij < Pijmin, then Pij = 0 and α ij and β ij are made equal to zero while evolving Aij and Bij from eqn (2.18) and step 4 is repeated.

b) If any of the unit generation exceeds its maximum limiting value Pij > Pij max then Pij = Pijmax are set and α ij & β ij are made equal to zero while evolving Aij and Bij from eqn (2.18) and step 4 is repeated.

A detailed flow-chart is shown in Fig 2.1


START
READ aij, bij and cij Pijmin & Pijmax and PD
Compute the equivalent cost function co-efficients of plants Xi, Yi & Zi
Compute the participation factors Li & Mi
Compute the individual plant generation, PGi
IS PGi < PGimin
Yes
PDi = 0 Xi =Yi = Zi = 0
No
IS PGi < PGimax
Yes
PGi = PGimax PD = PD - PGimax Xi =Yi = Zi = 0
Compute Individual Unit loading Pij
B
A
C

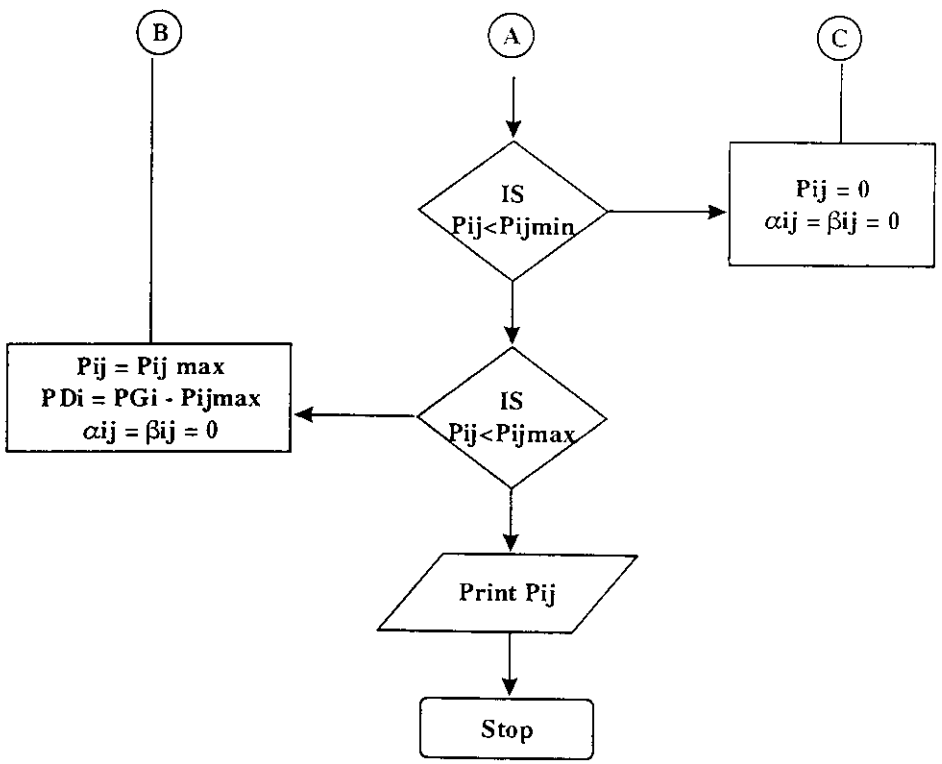


Fig 2.1 Flow chart for Optimum Economic Dispatch

A program has been developed in C language using this algorithm and has been tested. The selected system is assumed to have 3 plants each having 3 units. The cost functions of the units in each plant along with the inequality capacity cosntraints are given. Assuming the system Load PD is varying from 100 MW to 800 MW in steps of 100MW, the optimum generation schedule obtained are given.

This method seems to be very simple but at the same time potential enough to evolve the equivalent cost function of plant having any number of generating units. Since this technique does not involve any approximatory assumptions, the resulting optimal generation schedule is found to be exact. This technique can be extended to develop a direct method for interchange evaluation.

Chapter 3
Method of Interchange Evaluation

In conventional interchange evaluation method the interchange among the areas are initially zero i.e the power demand in each area is met by its own generating unit. This necessitates separate dispatch calcualtion for each area which are normally iterative in nature. Thereafter an iterative interchange evaluation is performed allowing free flow of power among the interconnected areas. so in conventional method an M area system requires M+1 iterative economic dispatch calculations.

The proposed method replaces all the iterative dispatch calculations by a simple direct technique.

3.1 Development of the Technique

The equivalent cost function of a plant is

$$Fti = X_i P_D i^2 + Y_i P_D i + Zi,\ i = 1, 2, ...m \quad \text{-------(3.1)}$$

Where P_{Di} is the power demand of i^{th} plant

Xi, Yi, and Zi are the plant cost co-effiecients which can be expressed interms ofunit cost Co-efficients.

Also unit generation can be expressed as linear function of plant demands

$$\text{(ie) } Pij = AijP_{Di} + Bij \quad \text{----------} \quad (3.2)$$

Where Aij, Bij are plant participation factors which can be expressed interms of unit cost co-efficient.

So for a given area once the commitment of units are over economic dispatch calculation can be obtained from this equation by merely substituting the value of area demand P_{Di}

$$P_{Gi} = L_i P_D + Mi \quad \text{----------} \quad (3.3)$$

So to perform economic dispatch calcuation of interconnected network.

1. Substitute the value of $_{PD}$, the system demand in equation (3.3) and obtain area geneation P_{Gi}

2. Substitute the above value of P_{Gi} instead of P_{Di} in equation (3.2) to obtain individual unit generation.

3.2 Algorithm For Economic Interchange Evaluation

1. Read in the cost coefficients a_{ij}, b_{ij}, c_{ij}, the inequality capacity constraints P_{ij}min, P_{ij}max and the area demands P_{Di}.

2. Compute α ij and β ij using equation (3.2)

3. a). Compute P_{ij} on substitution P_{Di} in equation (3.2)

   b). If $P_{ij} < P_{ij}$min then set that $P_{ij=0}$ recalculate α ij & β ij taking aij =bij = cij = 0;

   c). If $P_{ij} > P_{ij}$max; then take that $P_{ij} = P_{ij}$max; Make that units $a_{ij} = b_{ij} = C_{ij} = 0$; $P_{Di} = P_{Di} - P_{ij}$max ; Goto step 2.

4. Compute the total operating cost (individual area dispatch) using equation (1) and $Ft = \sum_{i}^{M} \sum_{j}^{N} Fij$

5. a) Compute total system demand using

   $$P_D = \sum_{i=1}^{M} P_{Di}$$

   b) Calculate x,y,z and Li, and Mi using equation (3:3)

c) Substitute P_D from step 5(a) into equation (3:3) and get area generation P_{GI}

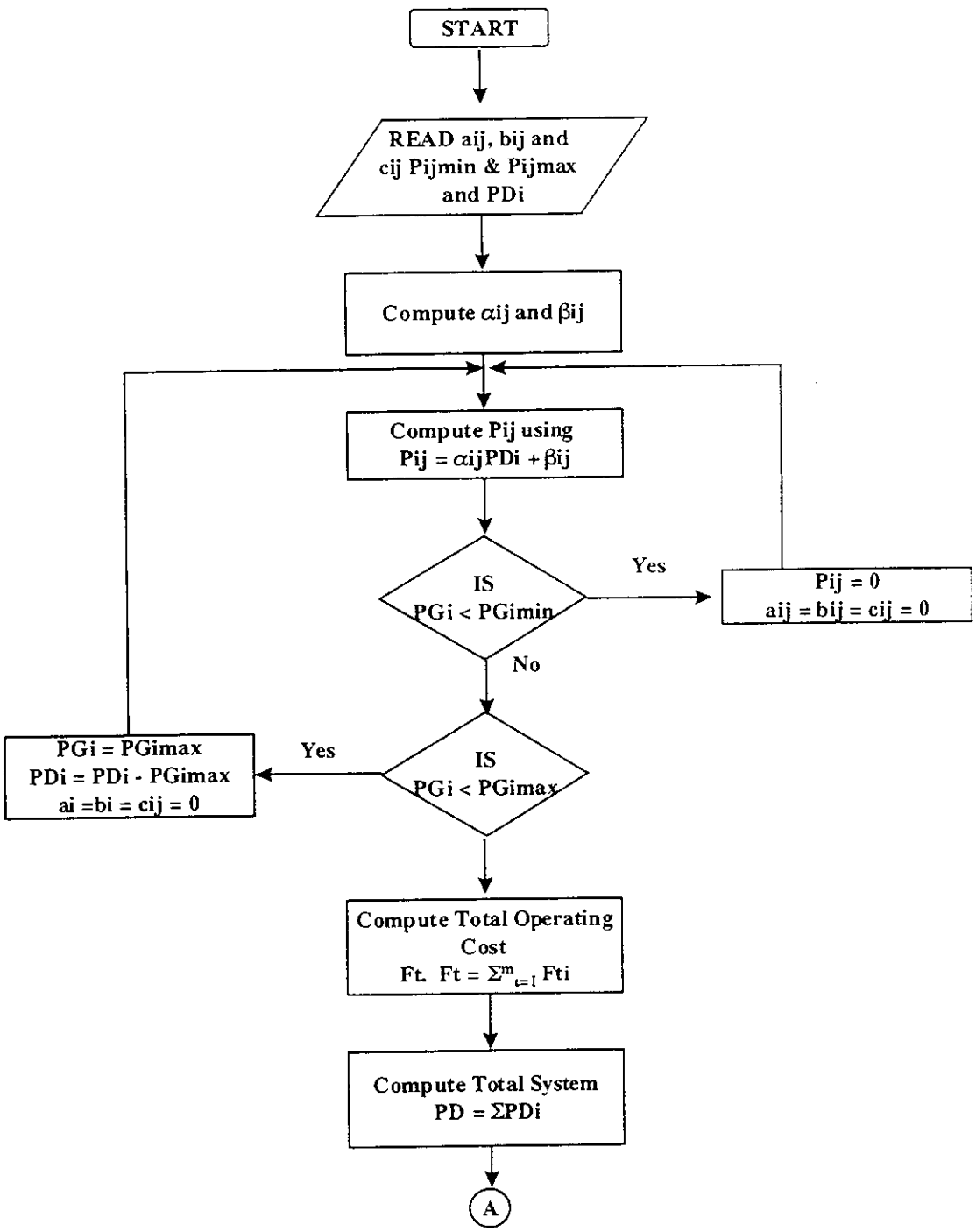
d) Substitute this P_{GI} instead of P_{Di} in equation (3:2) and get unit generations of area i. If any of the unit generation violates its capacity constraints, repeat step 3(b) and 3(c).

e) Compute the total operating cost (combined economic dispatch calculation) using equation (2.7) and $Ft = \sum_{i}^{M} \sum_{j}^{N} Fij$

6. Compute the net saving as the difference between f_t in step 5(e) and f_t step 4.

7. Distribute the net saving among participating areas / utilities on an equitable basis.

The detailed procedure to compute the interchange evaluation of interconnected power system is depicted in the Flow chart.


START
READ aij, bij and cij Pijmin & Pijmax and PDi
Compute αij and βij
Compute Pij using Pij = αijPDi + βij
IS PGi < PGimin
Yes
Pij = 0 aij = bij = cij = 0
No
IS PGi < PGimax
Yes
PGi = PGimax PDi = PDi - PGimax ai =bi = cij = 0
Compute Total Operating Cost Ft. Ft = Σ^n_{i=1} Fti
Compute Total System PD = ΣPDi
A

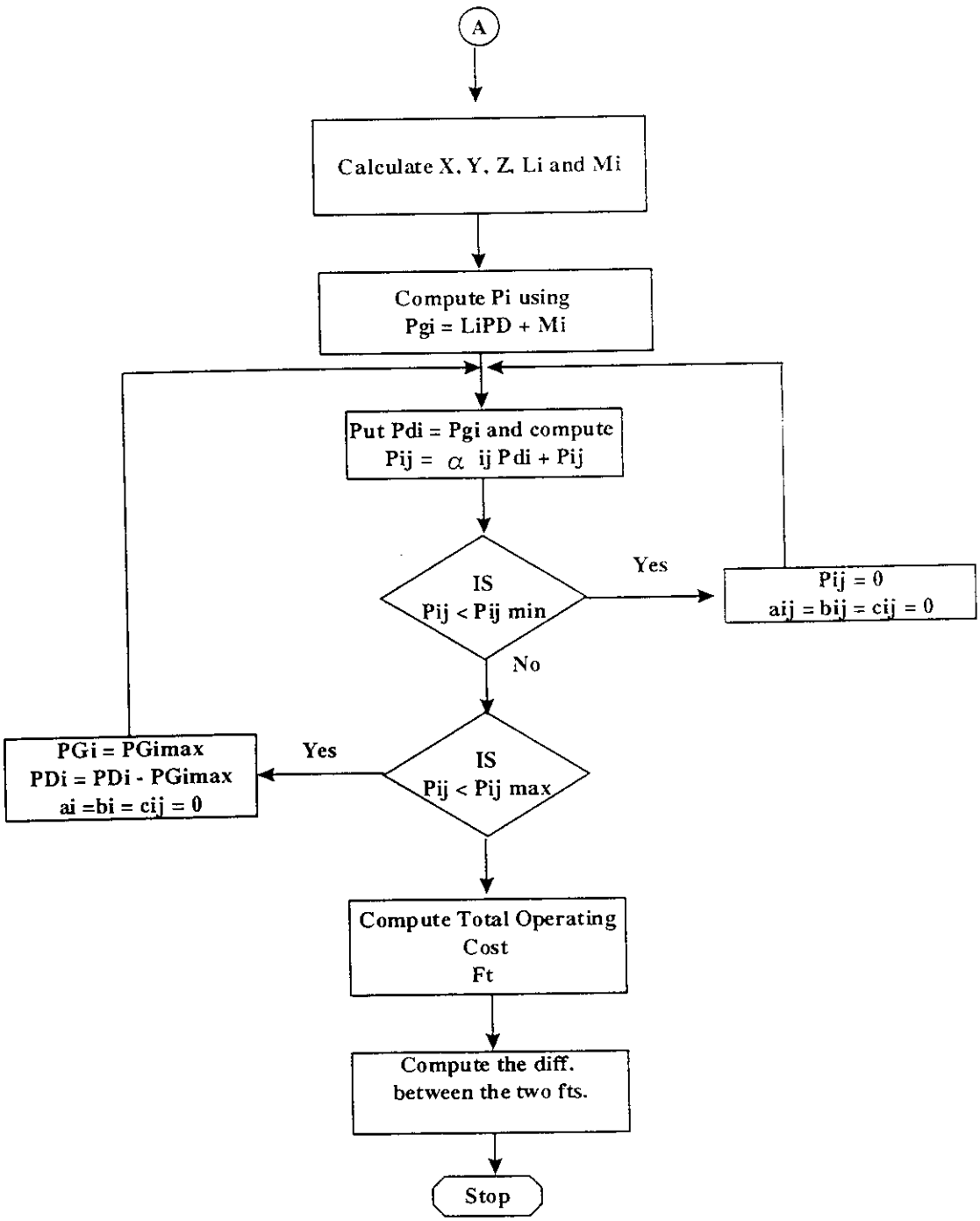


Fig 3.1 Flow chart for Interchange evaluation of utilities

3.4 Computer Program

A program has been developed in C language using the algorithm presented in 3.2 and has been tested.

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
float a[10][10],c[10][10],b[10][10],cf[10],nbet1[10],alp3[10],bet3[10];
float alpha[10][10],beta[10][10],alp[10],bet[10],ft1[10],sf,F1[10][10],FT1;
float pgmi[30],pgma[30],pdi[10],A[10][10],B[10][10],pg[10],pt[10],FT2;
float pijmi[10][10],pijma[10][10],alpha1[10][10],alpha2[10][10];
float X[10][10],Y[10][10],Z[10][10],beta1[10][10],beta2[10][10],beta3[10][10];
float x1[10],y1[10],z1[10],p[10][10],alp1[10],alp2[10],bet1[10],bet2[10];
float nalp[10],nbet[10],pt1[10],pt2[10],x2[10],y2[10],z2[10],f[10][10];
int j,k,i,l,i1,g,m,d,n,temp[10],pd,sum,sum1,t2,t3,e,h;
int t1=0,u,v,o,g1;
float r1,d1,di[10],ri[10],alpha3[10][10],F2[10][10],ft2[10],lamda1[10];
float li[10],mi[10],ri1[10],di1[10],ri2[10],di2[10],li2[10],lamda2[10];
float li1[10],mi1[10],q,q1,r11,d11,it[10],r12,d12,mi2[10];
void main()
{
pow(5,2);
clrscr();
printf("\n Enter the number of rows m & no of columns n");
scanf("\n %d",&m);
scanf("\n %d",&n);

printf("\n Enter the elements of matrix a");
for (i=1;i<=m;i++)
{
 for (j=1;j<=n;j++)
 {
   scanf("\n %f",&a[i][j]);
 }
}

printf("\n Enter the elements for matrix b");
for (i=1;i<=m;i++)
  {
   for (j=1;j<=n;j++)
     {
        scanf("\n %f",&b[i][j]);
```

```
    }
  }

printf ("\n Enter the elements for matrix c");
 for (i=1;i<=m;i++)
   {
   for (j=1;j<=n;j++)
     {
        scanf ("\n %f",&c[i][j]);
     }
   }

printf ("\n Enter the Power Demand Pd");
scanf ("\n %d",&pd);

printf("\n Enter the Pgi minimum");
 for (i=1;i<=m;i++)
   {
    scanf("\n %f",&pgmi[i]);
   }

printf("\n Enter the Pgi maximum");
 for (i=1;i<=m;i++)
   {
       scanf("\n %f",&pgma[i]);
   }

printf("\n Enter the Pij minimum");
 for(i=1;i<=m;i++)
   {
       for (j=1;j<=n;j++)
         {
          scanf("\n %f",&pijmi[i][j]);

         }
   }

printf ("\n Enter the Pij maximum");
 for (i=1;i<=m;i++)
```

```
    {
     for (j=1;j<=n;j++)
         {
          scanf("\n %f",&pijma[i][j]);
         }
    }

/* calculating alpha(i,j) */
 for (i=1;i<=m;i++)
    {
      for (j=1;j<=n;j++)
          {
          alpha[i][j]=a[i][1]/a[i][j];
          }
     }

 for (i=1;i<=m;i++)
    {
     for (j=1;j<=n;j++)
         {
          /*  printf ("\nalpha = %f", alpha[i][j]);  */
            alpha1[i][j]=alpha[i][j];
            alpha2[i][j]=alpha[i][j];
            alpha3[i][j]=alpha[i][j];
          }
           /*  getch();  */
     }


/* calculating alpha(i) */
  for (i=1;i<=m;i++)
    {
        alp[i]=0;
        for (j=1;j<=n;j++)
          {
           alp[i]= alp[i]+alpha[i][j];
          }
     }
```

```
  for (i=1;i<=m;i++)
    {
     /*  printf("\n alp %f", alp [i]);  */
         alp1[i]=alp[i];
         alp2[i]=alp[i];
     }
       /*   getch();  */

/* calculating beta(i,j) */
  for (i=1;i<=m;i++)
    {
       for (j=1;j<=n;j++)
         {
          beta[i][j]=((b[i][1] - b[i][j]) / (2 *(a[i][j])));
          beta1[i][j]=beta[i][j];
          beta2[i][j]=beta[i][j];
          beta3[i][j]=beta[i][j];
          /*  printf ("\n beta[i][j] = %f",beta[i][j]);  */
          }
          /*  getch(); */
     }

/* calculating */

  for (i=1;i<=m;i++)
    bet[i]=0;
    {
     for (j=1;j<=m;j++)
        {
         bet[i] = bet[i] +beta[i][j];
          /*  getch(); */
        }
          /*  getch(); */
     }
/* printing beta(i)    */
  for (i=1;i<=m;i++)
```

```
    {
       /*   printf ("\n bet[i] = %f",bet[i]);   */
          bet1[i] = bet[i];
          bet2[i] = bet[i];
       /*   getch();          */
     }
/* printing Aij's ndBij's*/
  for (i=1;i<=m;i++)
    {
       for (j=1;j<=n;j++)
         {
              A[i][j] = alpha[i][j]/alp[i];
              B[i][j] = ((-alpha[i][j]*bet[i])/alp[i])+beta[i][j];
          /* printf("\nA ij's = %f", A[i][j]);
              printf("\nBij's = %f", B[i][j]);   */
         }
              /*  getch(); */
     }
/* calculating X(i,j),Y(i,j) & Z(i,j) */
  for (i=1;i<=m;i++)
    {
       for (j=1;j<=n;j++)
         {
          X[i][j] = (a[i][j] * (A[i][j]*A[i][j]));
          Y[i][j] = (2*a[i][j] * A[i][j] * B[i][j]) + (b[i][j]*A[i][j]);
          Z[i][j] = ((a[i][j] * (B[i][j]*B[i][j])) + (b[i][j]* B[i][j]) + c[i][j]);
         }
    }
/* printing */
/*
  for (i=1;i<=m;i++)
    {
       for (j=1;j<=n;j++)
         {
          printf ("\n xi %dj %d %f ",i,j,X[i][j]);
          printf ("\n yi %dj %d %f",i,j,Y[i][j]);
          printf ("\n zi %dj %d %f",i,j,Z[i][j]);
         }
```

```
                getch();
}       */

/* calculating x1(i,j),y1(i,j),z1(i,j) */
  for (i=1;i<=m;i++)
    {
        x1[i]=0;
        y1[i]=0;
        z1[i]=0;
       for (j=1;j<=n;j++)
        {
          x1[i]=x1[i]+X[i][j];
          y1[i]=y1[i]+Y[i][j];
          z1[i]=z1[i]+Z[i][j];
        }
    }

/* printing  */
/*
   for (i=1;i<=m;i++)
     {
        printf ("\n  xi= %f",x1[i]);
        printf("\n yi= %f",y1[i]);
        printf("\n zi= %f",z1[i]);
      }
        getch();    */

/* calculating ri,di */
  q=x1[1];
  q1=y1[1];
  for (i=1;i<=m;i++)
    {
        ri[i]=q/x1[i];
        di[i]=(q1-y1[i])/(2*x1[i]);
    }
/* Printing */
/*
  for (i=1;i<=m;i++)
    {
```

```
        printf("\n gamma i %f",ri[i]);
        printf("\n delta i %f",di[i]);
    }
        getch();  */

/* calculating r1 & d1 */
  d1=0;
 r1=0;
 for (i=1;i<=m;i++)
   {
       r1=r1+ri[i];
       d1=d1+di[i];
    }

/* printing*/
/*
  printf ("\n sum gamma %f",r1);
  printf ("\n sum delta %f",d1);
  getch(); */

/* calculating li & mi */
  for (i=1;i<=m;i++)
    {
       li[i]=ri[i]/r1;
       mi[i]= ((((-ri[i]) *d1) / r1) + di[i]);
    }

/* printing*/
/*
  for (i=1;i<=m;i++)
    {
       printf("\n li %f",li[i]);
       printf("\n mi %f",mi[i]);
     }
        getch();  */

/* calculate pgi */
  for (i=1;i<=m;i++)
    {
```

```
        pg[i]= (li[i] * pd) + mi[i];
    }

/* printing*/
  for (i=1;i<=m;i++)
    {
        printf("\n pgi's %f",pg[i]);
    }
     getch();

/* creating temporary arrays */
  for (i=1;i<=m;i++)
    {
        ri1[i] = ri[i];
        ri2[i] = ri[i];
        di1[i] = di[i];
        di2[i] = di[i];
    /*  printf ("\n temp ri1 %f " ,ri1[i]);
        printf ("\n temp di2 %f " ,di2[i]);
        getch();                          */
    }
 /* checking pgi < pgimin */
   r11=0;
   d11=0;
   for (i=1;i<=m;i++)
     {
       if (pg[i] < pgmi[i])
        {
         ri1[i]=0;
         di1[i]=0;
        }
     }

 /* calculating new r11 d11 */
    for (i=1;i<=m;i++)
      {
        r11=r11+ri1[i];
        d11=d11+di1[i];
```

```
    }

   for (i=1;i<=m;i++)
     {
       li1[i]=ri1[i]/r11;
       mi1[i]=(((-ri1[i]*d11)/r11)+di1[i]):
     }

/* printing */
/*
  for (i=1;i<=m;i++)
    {
       printf("\n li1[i] %f",li1[i]);
       printf ("\n mi1[i] %f",mi1[i]);
    }
       getch();  */

/* calculating new pgi */
  for (i=1;i<=m;i++)
    {
        pg[i]= (li1[i]*pd) + mi1[i];
    }

/* printing */
  for (i=1;i<=m;i++)
    {
       printf ("\n new pgi %f",pg[i]);
    }
       getch();

/* calculating pgi > pgmax */
  l=1;
  for (i=1;i<=m;i++)
    {
   if (pg[i] > pgma[i])
       {
         pd=pd-pgma[i];
        temp[l]=i;
        ri2[i]=0;
```

```
            di2[i]=0;
            l++;
           }
      }
    l=l-1;
    if(l==0)
     {
  for (i=1;i<=m;i++)
    {
       printf("\n Pgi's = %f",pg[i]);
       getch();
    }
       getch();
    }
/*  calculate r12,d12 */
    if (l!=0)
     {
         r12=0;
         d12=0;
    for (i=1;i<=m;i++)
      {
          r12=r12 + ri2[i];
          d12=d12 + di2[i];
      }


/*  calculating li & mi */
   for (i=1;i<=m;i++)
     {
         li2[i]=ri2[i]/r12;
         mi2[i]=(-ri2[i] * d12)/ r12 + di2[i];
     }

   for (e=l;e>=1;e—)
     {
         i = temp[l]; /* i1=i */
         li2[i]=0;
         mi2[i]=pgma[i];
    /*  printf("\n li2[i] = %f",li2[i]);
```

```
        printf("\n mi2[i] = %f",mi2[i]);  */

    }
/*  for (i=l;i>=1;i—)
     {
        li2[i]=li1[i];
        mi2[i]=mi1[i];
       } */
/* calculating pgi's */
  for (i=1;i<=m;i++)
    {
       pg[i]=(li2[i]*pd)+mi2[i];

    }
         getch();
/* printing all the pgi's */
  printf ( "\n pgi vales are");
  for (i=1;i<=m;i++)
    {
       pdi[i]=pg[i];
       printf ("\n result= %f",pg[i]);
     }
        getch();
     }
         getch();
/* temporary pgi in pti */
  for (i=1;i<=m;i++)
    {
        pt[i] =pg[i];
        pt1[i] = pg[i];
        pt2[i] = pg[i];
    }

/* calculating pij */
  for (i=1;i<=m;i++)
    {
      for (j=1;j<=n;j++)
        {
        if (pt[i]==0)
```

```
        {
         alpha1[i][j]=0;
         beta1[i][j]=0;
        }
      p[i][j] = alpha1[i][j] * ((pt[i]-bet1[i])/alp1[i]) + beta1[i][j];

        }
    }

/* printing pij's */
  for (i=1;i<=m;i++)
    {
       for (j=1;j<=n;j++)
        {
        printf("\n 1pij = %f",p[i][j]);
        getch();
        }
    }
  for (i=1;i<=m;i++)
    {
       for (j=1;j<=n;j++)
        {
        if (p[i][j] > pijma[i][j])
        o=1;
        }
     }
  for (i=1;i<=m;i++)
    {
      for (j=1;j<=n;j++)
        {
        if (p[i][j] < pijmi[i][j])
        o=2;
        }
     }
    if (o==1)
     {
     do
       {
        g1=0;
```

```
        for (i=1;i<=m;i++)
          {
          for (j=1;j<=n;j++)
            {
          if (p[i][j] > pijma[i][j])
              {
               p[i][j]=pijma[i][j];
               printf("\n 2pij = %f ",p[i][j]);
               getch();

               pt[i]=pt[i]-pijma[i][j];
               printf("\n pt[i] =%f",pt[i]);
               getch();

              alpha1[i][j]=0;
              beta1[i][j]=0;
              alpha2[i][j]=0;
              beta2[i][j]=pijma[i][j];
              }
           }
  alp1[i]=0;
  bet1[i]=0;

    for (j=1;j<=n;j++)
       {

       {
        alp1[i]=alp1[i]+alpha1[i][j];
        bet1[i]=bet1[i]+beta1[i][j];
        }
/*      printf("\n alp1[i]= %f",alp1[i]);
         printf("\n bet1[i]= %f",bet1[i]);   */
        }
    for (j=1;j<=n;j++)
       {
      p[i][j] = alpha2[i][j] * ((pt[i]-bet1[i]) / alp1[i]) + beta2[i][j];
      printf("\n 2pij= %f",p[i][j]);
      getch();
       }
```

```
    }
 for (i=1;i<=m;i++)
   {
    for (j=1;j<=n;j++)
        {
         if (p[i][j] > pijma[i][j])
          {
          g1=1;

          }
        }
   }
 }while(g1==1);
}

 if (o==2)
  {
  for (i=1;i<=m;i++)
     {
     u=0;
     v=0;

 if (pg[i]==0)
  {
 for (j=1;j<=n;j++)
    {
        alpha3[i][j]=0;
        beta3[i][j]=0;
        u=1;
        v=1;
    }
  }
 for (j=1;j<=n;j++)
    {
  if (p[i][j] < pijmi[i][j])
     {
    alpha3[i][j]=0;
    beta3[i][j]=0;
     }
```

```
  }

 alp3[i]=0+u;
 bet3[i]=0+v;


 for (j=1;j<=n;j++)
   {
     alp3[i]=alp3[i]+alpha3[i][j];
     bet3[i]=bet3[i]+beta3[i][j];

   }
for (j=1;j<=n;j++)
  {
 p[i][j] = alpha3[i][j] * ((pt1[i] - bet3[i]) / alp3[i]) + beta3[i][j];
 printf("\n 3pij`s =%f",p[i][j]);
  getch();
  }
 }
}

 if (o==1)
  {
 for (i=1;i<=m;i++)
   for (j=1;j<=n;j++)
    {
   A[i][j]=alpha1[i][j]/alp1[i];
   B[i][j]=((-alpha1[i][j]*bet1[i])/alp1[i])+beta1[i][j];
/*  printf("\n Aij`s=%f",A[i][j]);
    printf("\n Bij`s=%f",B[i][j]);  */

    }
  }

 if(o==2)
  {
 for (i=1;i<=m;i++)
   for (j=1;j<=n;j++)
      {
```

```
      A[i][j]=alpha3[i][j]/alp3[i];
      B[i][j]=((-alpha3[i][j]*bet3[i])/alp3[i])+beta3[i][j];
/*   printf("\n Aij`s=%f",A[i][j]);
      printf("\n Bij`s=%f",B[i][j]);
      getch();                          */
     }
 }

 if (o!=1 && o!=2)
  {
 for (i=1;i<=m;i++)
   for (j=1;j<=n;j++)
      {
      A[i][j]=alpha1[i][j]/alp1[i];
      B[i][j]=((-alpha1[i][j]*bet1[i])/alp1[i])+beta1[i][j];
/*   printf("\n Aij`s =%f",A[i][j]);
      printf("\n B[i][j]=%f",B[i][j]);
      getch();  */
      }
  }

 for (i=1;i<=m;i++)
  for (j=1;j<=n;j++)
    {
   X[i][j]=(a[i][j]*A[i][j]*A[i][j]);
   Y[i][j]=(2*a[i][j]*A[i][j]*B[i][j])+(b[i][j]*A[i][j]);
   Z[i][j]=(a[i][j]*B[i][j]*B[i][j])+(b[i][j]*B[i][j])+c[i][j];
    }


 for (i=1;i<=m;i++)
   {
   x1[i]=0;
   y1[i]=0;
   z1[i]=0;
   for (j=1;j<=n;j++)
      {
      x1[i]=x1[i]+X[i][j];
      y1[i]=y1[i]+Y[i][j];
```

```
        z1[i]=z1[i]+Z[i][j];
        }
    }

for (i=1;i<=m;i++)
  {
 for (j=1;j<=n;j++)
    {
 F1[i][j]=X[i][j]*pt[i]+Y[i][j]*pt[i]+Z[i][j];
  printf("\n F1ij`s = %f",F1[i][j]);
  getch();
  }
  }

for (i=1;i<=m;i++)
  {
 ft1[i]=0;
 for (j=1;j<=n;j++)
    {
  ft1[i]=ft1[i]+F1[i][j];

    }
   printf("\n ft1`s =%f",ft1[i]);

  }


 FT1=0;
for (i=1;i<=m;i++)
  {
  FT1=FT1+ft1[i];
  }
  printf("\n total gen cost=%f",FT1);
  getch();

/* calculating lambda */
  for(i=1;i<=m;i++)
    {
      lamda1[i] = ((2*a[i][1]*p[i][1]) +b[i][1]);
```

```
      printf("\n lamda1's = %f",lamda1[i]);
    getch();
    }
/* INTERCHANGE EVALUATION */
 printf("\n Enter the demand values-interchange evaluation ");
 for(i=1;i<=m;i++)
   {
   scanf("\n %f", &pg[i]);
   }
   for(i=1;i<=m;i++)
    {
    for (j=1;j<=n;j++)
     {
    alpha1[i][j]=alpha[i][j];
    alpha2[i][j]=alpha[i][j];
    alpha3[i][j]=alpha[i][j];
     beta1[i][j]=beta[i][j];
     beta2[i][j]=beta[i][j];
     beta3[i][j]=beta[i][j];
      }
      alp1[i]=alp[i];
      alp2[i]=alp[i];
       bet1[i] = bet[i];
       bet2[i] = bet[i];
     }

/*  temporary pgi in pti */
   for (i=1;i<=m;i++)
     {
     pt[i] =pg[i];
     pt1[i] = pg[i];
     pt2[i] = pg[i];
      }

/*  calculating pij */
    for (i=1;i<=m;i++)
     {
     for (j=1;j<=n;j++)
      {
```

```
    if (pt[i]==0)
     {
    alpha1[i][j]=0;
    beta1[i][j]=0;
     }
    p[i][j] = alpha1[i][j] * ((pt[i]-bet1[i])/alp1[i]) + beta1[i][j];

    }
   }

/* printing pij's */
   for (i=1;i<=m;i++)
    for (j=1;j<=n;j++)
       {
       printf("\n11pij = %f",p[i][j]);
       getch();
       }
   for (i=1;i<=m;i++)
    {
    for (j=1;j<=n;j++)
     {
    if (p[i][j] > pijma[i][j])
     o=1;
     }
    }
   for (i=1;i<=m;i++)
    {
    for (j=1;j<=n;j++)
     {
    if (p[i][j] < pijmi[i][j])
     o=2;
     }
    }

   if (o==1)
    {
    do
     {
```

```
 g1=0;
 for (i=1;i<=m;i++)
  {
  for (j=1;j<=n;j++)
     {
     if (p[i][j] > pijma[i][j])
      {
      p[i][j]=pijma[i][j];
      printf("\n 22pij = %f ",p[i][j]);
      getch();
      pt[i]=pt[i]-pijma[i][j];
      printf("\n pt[i] =%f",pt[i]);
      getch();

      alpha1[i][j]=0;
      beta1[i][j]=0;
      alpha2[i][j]=0;
      beta2[i][j]=pijma[i][j];
      }
    }
  alp1[i]=0;
  bet1[i]=0;

for (j=1;j<=n;j++)
 {

 alp1[i]=alp1[i]+alpha1[i][j];
 bet1[i]=bet1[i]+beta1[i][j];

/*  printf("\n alp1[i]= %f",alp1[i]);
      getch();
      printf("\n bet1[i]= %f",bet1[i]);
      getch(); */
  }
for (j=1;j<=n;j++)
 {
p[i][j] = alpha2[i][j] * ((pt[i]-bet1[i]) / alp1[i]) + beta2[i][j];
```

```
 printf("\n 22pij= %f",p[i][j]);
  getch();
  }
}
for (i=1;i<=m;i++)
  {
  for (j=1;j<=n;j++)
    {
  if (p[i][j] > pijma[i][j])
     {
     g1=1;
     }
    }
  }
}while(g1==1);
}

 if (o==2)
  {
 for (i=1;i<=m;i++)
   {
   u=0;
   v=0;

   if (pg[i]==0)
     {
    for (j=1;j<=n;j++)
      {
    alpha3[i][j]=0;
     beta3[i][j]=0;
      u=1;
      v=1;
      }
    }
 for (j=1;j<=n;j++)
  {
 if (p[i][j] < pijmi[i][j])
    {
    alpha3[i][j]=0;
```

```
   beta3[i][j]=0;
  }
 }

alp3[i]=0+u;
bet3[i]=0+v;


for (j=1;j<=n;j++)
 {
alp3[i]=alp3[i]+alpha3[i][j];
bet3[i]=bet3[i]+beta3[i][j];

 }
for (j=1;j<=n;j++)
 {
p[i][j] = alpha3[i][j] * ((pt1[i] - bet3[i]) / alp3[i]) + beta3[i][j];
 printf("\n 33pij`s =%f",p[i][j]);
  getch();
 }
}
}

if (o==1)
 {
for (i=1;i<=m;i++)
 for (j=1;j<=n;j++)
  {
 A[i][j]=alpha1[i][j]/alp1[i];
 B[i][j]=((-alpha1[i][j]*bet1[i])/alp1[i])+beta1[i][j];
/*  printf("\n Aij`s=%f",A[i][j]);
      printf("\n Bij`s=%f",B[i][j]);
      getch();                    */
  }
 }

if(o==2)
 {
for (i=1;i<=m;i++)
```

```
 for (j=1;j<=n;j++)
    {
    A[i][j]=alpha3[i][j]/alp3[i];
    B[i][j]=((-alpha3[i][j]*bet3[i])/alp3[i])+beta3[i][j];
    /*  printf("\n Aij`s=%f",A[i][j]);
       printf("\n Bij`s=%f",B[i][j]);
        getch();                        */
    }
 }

if (o!=1 && o!=2)
 {
 for (i=1;i<=m;i++)
  for (j=1;j<=n;j++)
   {
    A[i][j]=alpha1[i][j]/alp1[i];
    B[i][j]=((-alpha1[i][j]*bet1[i])/alp1[i])+beta1[i][j];
    /*  printf("\n Aij`s =%f",A[i][j]);
       printf("\n B[i][j]=%f",B[i][j]);
        getch();        */
   }
  }

for (i=1;i<=m;i++)
  {
  for (j=1;j<=n;j++)
   {
  X[i][j]=(a[i][j]*A[i][j]*A[i][j]);
  Y[i][j]=(2*a[i][j]*A[i][j]*B[i][j])+(b[i][j]*A[i][j]);
  Z[i][j]=(a[i][j]*B[i][j]*B[i][j])+(b[i][j]*B[i][j])+c[i][j];
   }
  }

for (i=1;i<=m;i++)
 {
 x1[i]=0;
 y1[i]=0;
 z1[i]=0;
 for (j=1;j<=n;j++)
```

```
 {
 x1[i]=x1[i]+X[i][j];
 y1[i]=y1[i]+Y[i][j];
 z1[i]=z1[i]+Z[i][j];
  }
 }

for (i=1;i<=m;i++)
 {
 for (j=1;j<=n;j++)
  {
 F2[i][j]=X[i][j]*pt[i]+Y[i][j]*pt[i]+Z[i][j];
  printf("\n F2ij`s = %f",F2[i][j]);
   getch();
  }
 }

for (i=1;i<=m;i++)
 {
 ft2[i]=
 0;
 for (j=1;j<=n;j++)
  {
   ft2[i]=ft2[i]+
   F2[i][j];
  }
   printf("\n ft2`s =%f",ft2[i]);
 }


 FT2=0;
for (i=1;i<=m;i++)
{
FT2=FT2+ft2[i];
}
printf("\n 2total gen cost=%f",FT2);
printf("\n 1total gen cost=%f",FT1);
getch();
/* calculating lamda2's */
```

```
for(i=1;i<=m;i++)
 {
lamda2[i]=(2*a[i][1]*p[i][1] + b[i][1]):
printf("\n lamda2's = %f",lamda2[i]):
 getch();
 }
 }
```

The system selected for the optimum economic despatch is assumed to have 3 plants each having 3 units. The cost function along with the inequality capacity constraints are given below. And the optimum generation schedule for the plants as well as for the units are obtained.

The cost functions of units in each areas are given as

Plant 1 : $(20<P_1<450)$

$F_{11}= .009\ P_{11}^2 + 2.00\ P_{11} + 80.0;\ 10 < P_{11} < 200$

$F_{12}= .010P_{12}^2 + 2.2\ P_{12} + 95.0;\ 10 < P_{12} < 150$

$F_{13}= .015\ P_{13}^2 + 2.4\ P_{13} + 105.0;\ 10 < P_{13} < 100$

Plant 2 : $(25<P_2<250)$

$F_{21}= .020P_{21}^2 + 1.45\ P_{21} + 85.0;\ 10 < P_{21} < 100$

$F_{22}= .03P_{22}^2 + 1.4\ P_{22} + 100.0;\ 5 < P_{22} < 75$

$F_{23}= .035P_{23}^2 + 1.9\ P_{23} + 120.0;\ 10 < P_{23} < 75$

Plant 3 : ($25<P_3<225$)

$F_{31} = .015P_{31}^2 + 1.95\ P_{31} + 60.0;\ 10 < P_{31} < 102$

$F_{32} = .02P_{32}^2 + 2.05\ P_{32} + 80.0;\ 10 < P_{32} < 73$

$F_{33} = .025P_{33}^2 + 2.109\ P_{33} + 100.0;\ 5 < P_{33} < 50$

The Program is simulated and the results are tabulated.

```
 Enter the number of rows m & no of columns n 3 3

 Enter the elements of matrix a 0.009000 0.010000
0.015000 0.020000 0.030000 0.035000 0.015000 0.020000
0.025000

 Enter the elements for matrix b 2.000000 2.200000
2.400000 1.450000 1.400000 1.900000 1.950000 2.050000
2.109000

 Enter the elements for matrix c 80.000000 95.000000
105.000000 85.000000 100.000000 120.000000 60.000000
80.000000 100.000000

 Enter the Power Demand Pd 800

 Enter the Pgi minimum 20.000000 25.000000 25.000000

 Enter the Pgi maximum 450.000000 250.000000 225.000000

 Enter the Pij minimum 10.000000 10.000000 0.000000
10.000000 5.000000 10.000000 10.000000 10.000000
5.000000

 Enter the Pij maximum 200.000000 150.000000 100.000000
100.000000 75.000000 75.000000 102.000000 73.000000
50.000000

 pgi's 389.102722
 pgi's 187.526642
 pgi's 223.370590

 new pgi 389.102722
 new pgi 187.526642
 new pgi 223.370590

 lpij = 155.641083
 lpij = 130.076981
 lpij = 80.051308
 lpij = 83.788498
 lpij = 56.692333
 lpij = 41.450569
 lpij = 95.051315
 lpij = 68.788490
 lpij = 53.850788
```

```
2pij= 164.974426
2pij= 138.476974
2pij= 85.651314
2pij= 86.288498
2pij= 58.359001
2pij= 42.879139
2pij = 50.000000
pt[i] =173.370590
2pij= 100.497482
2pij= 72.873108
2pij= 50.000000

F1ij`s = 437.440857
F1ij`s = 395.696747
F1ij`s = 290.131134
F1ij`s = 219.370911
F1ij`s = 190.768112
F1ij`s = 186.015518
F1ij`s = 261.095673
F1ij`s = 225.821762
F1ij`s = 100.000000

ft1`s =1123.268677
ft1`s =596.154541
ft1`s =586.917419
total gen cost=2306.340576
```

The selected system is assumed to have 2 plants each having 3 units. The cost functions of the units in each plant anong with the inequality capacity constraints are given.

The load on the individual area's given by the user. The optimum generation schedule for the individual units are obtained.

The cost functions of the units in each areas are given :-

Area :

$F11 = 0.003124\ P_{11}^2 + 15.84\ P_{11} + 1122;\ 150\ Mw < P_{11} < 600\ MW$

$F12 = 0.02188 P_{12}^2 + 15.7\ P_{12} + 620\ ;\ 100\ MW < P_{12} < 400\ MW$

$F13 = 0.00564\ P_{13}^2 + 15.94\ P_{13} + 156\ ;\ 50\ MW < P13 < 200MW$

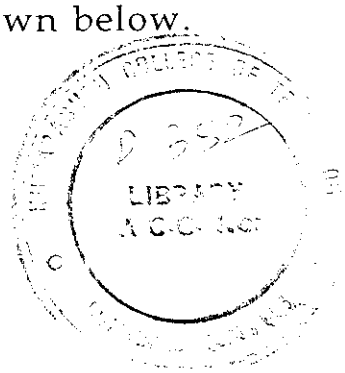
Area 2 :

$F21 = 0.002641\ P_{21}^2 + 13.414 P_{21} + 950;\ 140\ MW < P_{21} < 590\ MW$

$F22 = 0.003496 P_{22}^2 + 14.174 P_{22} + 560.5\ ;\ 110\ MW < P_{22} < 440\ MW$

$F23 = 0.003496 P_{23}^2 + 14.174 P_{23} + 560.5\ ;\ 110\ MW < P_{23} < 440\ MW$

The program simulated and the results are as shown below.



```
 Enter the number of rows m & no of columns n 3 3
 Enter the elements of matrix a 0.003124 0.021840
0.005640 0.002641 0.003496 0.003496
 Enter the elements for matrix b 15.340000 15.700000
15.940000 13.414000 14.174000 14.174000
 Enter the elements for matrix c 1122.000000 620.000000
156.000000 950.000000 560.500000 560.500000
 Enter the Power Demand Pd 1800
 Enter the Pgi minimum 300.000000 360.000000
 Enter the Pgi maximum 1200.000000 1470.000000
 Enter the Pij minimum 150.000000 100.000000 50.000000
140.000000 110.000000 110.000000
 Enter the Pij maximum 600.000000 400.000000 200.000000
590.000000 440.000000 440.000000

 pgi's 235.144196
 pgi's 1564.855835

 new pgi 0.000000
 new pgi 1800.000000

 pgi values are
 result= 330.000000
 result= 1470.000000

 lpij = 194.497543
 lpij = 30.969400
 lpij = 98.867134
 lpij = 585.454590
 lpij = 333.577087
 lpij = 333.577087

 3pij`s =218.073898
 3pij`s =0.000000
 3pij`s =111.926094
 3pij`s =672.034668
 3pij`s =398.982697
 3pij`s =398.982697

 F1ij`s = 4584.389160
 F1ij`s = 620.000000
 F1ij`s = 1932.951904
 F1ij`s = 10252.823242
 F1ij`s = 6088.828613
 F1ij`s = 6088.828613
```

```
 ft1`s =7137.340820
 ft1`s =22430.480469
 total gen cost=29567.820312

 lamda1's = 17.202526
 lamda1's = 16.963686


 Enter the demand values-interchange evaluation
700.000000
1100.000000

11pij = 412.570557
11pij = 62.105598
11pij = 219.657928
11pij = 438.095245
11pij = 222.256729
11pij = 222.256729

 33pij`s =456.184357
 33pij`s =0.000000
 33pij`s =243.815643
 33pij`s =524.675354
 33pij`s =287.662354
 33pij`s =287.662354

 F2ij`s = 8365.024414
 F2ij`s = 620.000000
 F2ij`s = 4027.048828
 F2ij`s = 8208.600586
 F2ij`s = 4544.551758
 F2ij`s = 4544.551758

 ft2`s =13012.073242
 ft2`s =17297.703125
 2total gen cost=30309.777344
 1total gen cost=29567.820312

 lamda2's = 18.690241
 lamda2's = 16.185335
```

Result analysis of interchange evaluation:

The power generations are computed for all the plants. And these generations are distributed among the units in each plant.

For a power demand of 800, the power generations of the plants are found to be 389.103, 187.526 & 223.37. In this case, none of the plant violates both the minimum and maximum limits. Hence the power generation remains the same.

These plant generations are distributed among its units, and it was found that the third unit of the third plant violates the maximum limit. Therefore its unit generation is set to P_{ij}= P_{ijmax}= 50. And the remaining is shared between the other units of that plant.

In case of interchange evaluations we have chosen the no. of plants to be 2 and each plant has 3 units. The total power demand is 1800. Initially the total power demand of the system is divided as 235.15 and 1564.85 for first and second plant respectively. Since the first plant violate the minimum condition its generation is fixed to be zero. And the second plant supplies the whole demand. Then it is checked for its maximum inequality constraints where in it was found that the generation of the second plant exceeds the maximum limit. Hence the

generation of the second plant was fixed to its max. value and the remaining was supplied by the first unit.

Then these generations are divided among its units and the unit generations are calculated.

The total operating cost are calculated for both cases. i.e. By considering the whole system to be 1 unit (inter connected), as well as by considering each plants supplying their own units themselves. When the above 2 cases are compared, we can find that the operating cost is less in the case of interconnected system.

Chapter 4

Conclusion

The economy aspect of interchange evaluation is considered in two project work. The method is developed by making use of the equivalent generation cost function of the interconnected system. When any of the units in the system is shut down due to fault of maintenance schedule the resultant equivalent cost function of the plant can be found out very easily by making its respective cost coefficient equal to zero. Likewise when any of the new unit is put into service, its effect on the equivalent cost function of the interconnected system can be a acertianed by including the incoming units cost co-efficient while valuating the cost co-efficient of the inter connected system.

The proposed method reduces the number of iterative dispatch calculations in interchange evaluation. It also takes care of the inequality capacity constraints of individual units./ The unit and area participation factors are to be calculated only one for a given system. However these area to be recalculated whenever the inequality capacity constriants are violated which seldom occurs. Hence the proposed method considerably reduces the time factor involved in interchange evaluation.

It is hoped that the method suggested can be extended by including an interchange energy brokerage system to match buyers with sellers through or central energy exchange. The energy broker can then decide the energy exchange between interconnected utility with a view to maximise the profit of all participating utility. Thus an overall energy management strategy for the entire interconnected system can be developed.

References

1. N. Ramaraj, "Adirect method to evolve the equivalent cost functions of plants with multiple generating units, " Journal of the institution of engineers (India), Vol 73, Pl El 5, December 1992, P 258 - 260.

2. Dr. N.Ramaraj and Dr. R.Rajaram, "A direct method of interchange evaluation among power utilities, "Journal of institution of Engineers (India) vol. 76, Feb 1996, Page 241 to 244.

3. C.L. Wadhwa, "Electrical Power systems" Newage international (P) limited, Publishers; India. April 1997.

4. Allen .J. Wood, Bruce F. Wollenburg " Power Generation, Operation and control". John Wiley L& sons, Newyork,I 1984.