Appendix E

MOL Program

Program Description

Input:

- The generating unit system data which consists of the heat input-output curves, ramp rate limits, unit operating limits, fuel cost of each unit, initial power outputs at time t=0, forecast load demands, and transmission loss B-matrix coefficients.

Output:

- The result of the program consists of the power output of each generating units and their highest and lowest possible power output, the power loss, the generator fuel cost and the CPU time of each time period.

```
/*-------------------------------------------------------------*/
/* Program : Merit Order Loading for Constrained ED problem
/* Compile : gcc mol.c -o MOL -lm
/* Execute : MOL
/*-------------------------------------------------------------*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "os.h"
#include "coops.c"

#define INF   pow(10,14);
#define MaxUnit 80
#define MaxHour 500
char    datafile[20], rsfile[20];
FILE    *OUTP;
FILE    *Inp;
typedef unsigned short  Word;
Word    piece[MaxUnit];
Word    Step,MaxStep,ref,NUnit,Member;

/* Cost Coefficient */
double  Fuel_Cost[MaxUnit],F[MaxUnit];
double  Ah[MaxUnit][10],Bh[MaxUnit][10],Ch[MaxUnit][10];
double  aC[MaxUnit][10],bC[MaxUnit][10],cC[MaxUnit][10];
double  P_int[MaxUnit][10];
/* B Matrix */
double  B00;
double  B0[MaxUnit];
double  B[MaxUnit][MaxUnit];

/* Based Power */
double  P_Base;

/* Power output variables */
double  DemandMin,DemandMax,Demand_Actual;
unsigned long combination;
double  SumPrmax,SumPrmin,PowPlant,SumPow;
double  Pmin[MaxUnit],Pmax[MaxUnit],UR[MaxUnit],DR[MaxUnit];
double  Prmin[MaxUnit],Prmax[MaxUnit],P[MaxUnit],P_ini[MaxUnit];
double  Pd[MaxHour];
double  TotalCost,MolCost,P_Loss;
int     List[MaxUnit];
double  Sub[MaxUnit],IC[MaxUnit];

/*-------- Time ----------*/
struct   timeval  start_time,stop_time,diff_time;
struct   timezone timz;

void TitleReport(void)
{       Word i;
        fprintf(OUTP,"\nData: %s  Result: %s\n",datafile,rsfile);
        fprintf(OUTP,"Step\tPowDemand\tPowPlant\tP_Loss\tSumPow\t");
        for (i=0; i < NUnit; i++) fprintf(OUTP,"Prmax[%d]\t",i);
        for (i=0; i < NUnit; i++) fprintf(OUTP,"Prmin[%d]\t",i);
        fprintf(OUTP,"SumPrmax\tSumPrmin\t",SumPrmax);
        for (i=0; i < NUnit; i++) fprintf(OUTP,"IC[%d]\t",i);
        for (i=0; i < NUnit; i++) fprintf(OUTP,"P[%d]\t",i);
        fprintf(OUTP,"MolCost\n");
}

void Arrange(int *List)
{
        int i,flag,ltemp; double temp;
        double CostList[MaxUnit];

        for(i=0;i<NUnit;i++){CostList[i]=UnitCost(i,Pmax[i])/Pmax[i];List[i] = i;}

        do{     flag = 0;
                for(i=0; i<NUnit-1; i++)
                if(CostList[i] > CostList[i+1])
                {       temp = CostList[i+1];
                        CostList[i+1] = CostList[i];
                        CostList[i] = temp;
                        ltemp = List[i+1];
```

```
                                List[i+1] = List[i];
                                List[i] = ltemp;
                                flag ++;
                        }
        }while(flag != 0);

        for (i=0; i < NUnit; i++)fprintf(OUTP,"List[%d] = %d\n",i,List[i]);
}

double MOL_Pmax(double PowDemand,double DemandMin,double *P,double *P_Loss,double *SumPow)
{
          int     i,loop;
          double  TempLess,AccumLess,Err;
      double  PowPlant;
      double  cost;
          loop = 0;
          AccumLess = PowDemand-DemandMin;
          TempLess  = AccumLess;
          for (i=0; i < NUnit; i++) Sub[i]=Prmax[i]-Prmin[i];

          do
          {   PowPlant=0; loop++;
               for (i=0; i < NUnit; i++)
               {
                         if (Sub[List[i]]>= TempLess)
                         {     P[List[i]] = Prmin[List[i]]+TempLess;
                               TempLess = 0;
                         }
                         else
                         {     P[List[i]] = Prmax[List[i]];
                               TempLess-= Sub[List[i]];
                         }
                         PowPlant += P[List[i]];
               }
               *P_Loss = B_Matrix(P);
               *SumPow = PowPlant  - *P_Loss;
               Err     = PowDemand - *SumPow;
               AccumLess +=  Err;
               TempLess   =  AccumLess;

        }  while ((Err > 0.0) && (loop <= 50));

        cost=CostCal(P);
        return cost;
}

void Report(void)
{  Word i;
        fprintf(OUTP,"%i\t",Step);
        fprintf(OUTP,"%8.4f\t",Pd[Step]);
        fprintf(OUTP,"%8.4f\t",PowPlant);
        fprintf(OUTP,"%8.4f\t",P_Loss);
        fprintf(OUTP,"%8.4f\t",SumPow);
        for (i=0; i < NUnit; i++) fprintf(OUTP,"%8.4f\t",Prmax[i]);
        for (i=0; i < NUnit; i++) fprintf(OUTP,"%8.4f\t",Prmin[i]);
        fprintf(OUTP,"%8.4f\t",SumPrmax);fprintf(OUTP,"%8.4f\t",SumPrmin);
        for (i=0; i < NUnit; i++) fprintf(OUTP,"%8.4f\t",P[i]);
        fprintf(OUTP,"%lf\n",MolCost);
}


main ()
{
   Word i,j;
   printf("\nData Filename:"); scanf("%s",datafile);
   printf("\n%s",datafile);
   sprintf(rsfile,"MOL%s",datafile);

   if((OUTP    = fopen(rsfile,"w"))==NULL){
               printf("cannot open file output\n");
               exit(0);
               }

  ReadData(); printf("\nIs Data Correct?");
  BuildCoef();
  Arrange(List);
  TitleReport();
```

```
  ClrScr();

/*----- Initialize time, bestcost and power output P ----*/
  diff_time.tv_sec = 0;
  diff_time.tv_usec = 0;
  Start_Time();

  TotalCost = 0;SumPrmin = 0; SumPrmax = 0;
  for (i=0; i < NUnit; i++)
  {     P[i] = Pmin[i];
        SumPrmin += Pmin[i]; SumPrmax += Pmax[i];
  }

  for(Step = 0; Step < MaxStep; Step++)
  {
        Demand_Actual = Pd[Step];
        DemandMin = 0; DemandMax = 0;
        SetRampLimit(P,Prmax,Prmin,&SumPrmax,&SumPrmin);
        DemandMin = SumPrmin-B_Matrix(Prmin);DemandMax=SumPrmax-B_Matrix(Prmax);
        MolCost = MOL_Pmax(Demand_Actual,DemandMin,P,&P_Loss,&SumPow);
        PowPlant = 0;for(i=0;i<NUnit;i++)PowPlant += P[i];
        ICC(P,IC);
        TotalCost += MolCost;
        Report();
   }

   Stop_Time();
   ReadTime(start_time,stop_time,&diff_time);

   /*--- Show Cost ---*/
   TotalCost /= 4;
   printf("\nTotalCost %.4lf",TotalCost);
   fprintf(OUTP,"\nTotal Cost = %lf",TotalCost);

   /*--- Show Time ---*/
   printf("\nTotal Time %6i seconds %6i micro_seconds",diff_time.tv_sec,diff_time.tv_usec);
   fprintf(OUTP,"\nTotal Time %6i seconds %6i micro_seconds",diff_time.tv_sec,
diff_time.tv_usec);
   fclose(OUTP);
   return 0;
}
```