รายการอ้างอิง

1. C.K. Pang and H.C. Chen , " Optimal Short-term Thermal Unit Commitment " IEEE Transactions on Power Apparatus and Systems , Vol. PAS-95 , no.4 , pp. 1336-1346 , July/August 1976
2. T.S. Dillon , K.W. Edwin , H.D. Kochs and R.J. Taud , " Integer Programming Approach to the Problem of Optimal Unit Commitment with Probabilistic Reserve determination " IEEE Transactions on Power Apparatus and Systems , Vol. PAS-97 , No. 6 , pp. 2154-2166 , November/December 1978
3. John A. Muckstadt , Sherri A. Koenig , " An Application of Lagrangian Relaxation to Scheduling in Power - Generation system " Operations Research , Vol. 25 , No. 3 , pp. 387- 403 , May-June 1977
4. G.S. Lauer , N.R. Sandell , D.R. Bertsekas and T.A. Posbergh , " Solution of Large-Scale Optimal Unit Commitment Problems " IEEE Transactions on Power Apparatus and Systems , Vol. PAS-101, pp. 79-86 , January 1982
5. M.L. Fisher , " The Lagrangian Relaxation Method for Solving Integer Programming Problems " , Management Science , Vol. 27 , No.1 , pp. 1-18 , January 1981
6. A. Merlin and P. Sandrin , " A New Method for Unit Commitment at Electricite de France " IEEE Transactions on Power Apparatus and Systems , Vol. PAS-102 , pp. 1218-1225 , May 1983
7. Arthur I. Cohen and Vahid R. Sherkat ," Optimization - Based Methods for Operations Scheduling " Proceeding of the IEEE , Vol. 75 , pp. 1574-1591 , December 1987
8. I. Cohen , S.H. Wan , " A Method for Solving the Fuel Constrained Unit Commitment Problem " IEEE Transactions on Power Systems , Vol. PWRS-2 , pp. 608-614 , August 1987
9. K. Aoki , T. Satoh , M. Itoh , T. Ichimori and K. Masegi , " Unit Commitment in a Large- Scale Power System Including Fuel Constrained Thermal and Pumped Storage Hydro " IEEE Transactions on Power Systems , Vol. PWRS-2 , pp. 1077-1084 , November 1987

10. Fulin Zhuang and F.D. Galiana , " Towards a More Rigorous and Practical Unit Commitment by Lagrangian Relaxation " IEEE Transactions on Power Systems , Vol. 3 , No. 2 , pp. 763-773 , May 1988

11. Jonathan F. Bard , " Short Term Scheduling of Thermal Electric Generators using Lagrangian Relaxation " Operations Research , Vol. 36 , No. 5 , pp. 756-766 , September - October 1988

12. S.M. Shahidehpour and S.K. Tong , " An Overview of Power Generation Scheduling in The Optimal Operation of a Large Scale Power System " Electric Machines and Power Systems , pp. 731-762 , Hemisphere Publishing Corporation , 1991

13. S.K. Tong and S.M. Shahidehpour , " A Combination of Lagrangian Relaxation and Linear Programming Approaches for Fuel Constrained Unit Commitment Problem" IEE Proceedings , Vol. 136 , Part C , No. 3 , pp. 162-174 , May 1989

14. S.K. Tong and S.M. Shahidehpour , " An Innovative Approach to Generation Scheduling in Large-Scale Hydro-Thermal Power Systems with Fuel Constrained Units " IEEE Transactions on Power Systems Vol. 5 , pp. 665-673 , May 1990

15. D.P. Bertsekas , G.S. Lauer , N.R. Sandell and T.A. Posbergh , " Optimal Short Term Scheduling of Large Scale Power Systems " IEEE Transactions on Automatic Control , Vol. AC-28 , No. 1 , pp. 1-11 , January 1983

16. A.J. Wood and B.F. Wollenberg , Power Generation , Operation and Control , John Wiley & Sons , New York , 1984

17. M.S. Bazaraa and C.M. Shetty , Nonlinear Programming - Theory and Algorithms , John Wiley & Sons , Singapore , 1990

18. J.K. Delson ,S.M. shahidehpour, " Linear Programming Applications to Power System Economics , Planning and Operations " IEEE Transactions on Power Systems , Vol 7 , No. 3 , pp 1155-1163 , August 1992

19. H.W. Zaininger, A.J. Wood, H.K. Clark, T.F. Laskowski, J.D. Burns, " Synthetic Electric Utility Systems for Evaluating Advanced Technologies " EPRI Report EM-285 , Electric Power Research Institute , Palo Alto , CA , February 1977

20. Jasbir S. Arora , Introduction to Optimum Design , McGraw - Hill Book Company , Singapore , 1989

ภาคผนวก

ภาคผนวก ก.

โปรแกรมที่ใช้ในการคำนวณค่ายูนิตคอมมิตเมนต์ โดยใช้วิธีรีแล็กเซชันแบบลากรองจ์

```
UNIT    INPUTDT;
INTERFACE
USES    CRT;
CONST   MaxUnit  =  200;
        MaxPeriod  =  24;
TYPE    CoefFuelCost  =  RECORD
                         a,b,c  :  Real;
                         END;
        CoefStartCost  =  RECORD
                         b1,b2  :  Real;
                         d      :  Real;
                         END;
        ThermalUnit  =  RECORD
                         number  :  Byte;
                         cfuel    :  CoefFuelCost;
                         cstart   :  CoefStartCost;
                         pmin     :  Real;
                         pmax     :  Real;
                         minup    :  ShortInt;
                         mindown:  ShortInt;
                         status   :  ShortInt;
                         END;
        Requirement  =  RECORD
                         period  :  Byte;
```

```
                    demand : Word;
                    reserve : Word;
                    END;
    FiTunit     = FILE OF ThermalUnit;
    FiReq       = FILE OF Requirement;
    PtTunit     = ^TunitNode;
    TunitNode   = RECORD
                    TunitRec : ThermalUnit;
                    LinkT    : PtTunit
                    END;
    PtReq       = ^ReqNode;
    ReqNode     = RECORD
                    ReqRec   : Requirement;
                    LinkR    : PtReq
                    END;
    PowerPt     = ^PowerNode;
    PowerNode   = RECORD
                    Pr       : Real;
                    LinkP    : PowerPt
                    END;
    CommitPt    = ^CommitNode;
    CommitNode  = RECORD
                    Ur       : Byte;
                    LinkU    : CommitPt
                    END;
    Power       = ARRAY [1..MaxUnit] OF PowerPt;
    Commit      = ARRAY [1..MaxUnit] OF CommitPt;
VAR TunitF      : FiTunit;
    ReqF        : FiReq;
    Tup         : PtTunit;
```

```
        Rqp      : PtReq;
        nunit    : Byte;
        nperiod  : Byte;
        P        : Power;
        U        : Commit;
        TxFi     : Text;
PROCEDURE DisplayBoard(NLine,NCol : Integer);
PROCEDURE DisplayFrame(NLine,NCol : Integer);
PROCEDURE InputDataT(VAR TunitF : FiTunit);
PROCEDURE InputDataR(VAR ReqF   : FiReq );
PROCEDURE PtrTunitFi(VAR TunitF : FiTunit;
                     VAR Tup    : PtTunit;
                     VAR nunit  : Byte );
PROCEDURE PtrReqFi( VAR ReqF    : FiReq;
                    VAR Rqp     : PtReq;
                    VAR nperiod : Byte );
PROCEDURE InitPUptr( VAR P      : Power;
                     VAR U      : Commit;
                     nunit,nperiod : Byte );
IMPLEMENTATION
{=================================================================}
PROCEDURE DisplayBoard(NLine,NCol : Integer);
VAR LeftCol,TopLine,Col,Line : Integer;
BEGIN
  IF Nline > 22 THEN Nline := 22;
  IF NCol > 75 THEN NCol := 75;
  TopLine := 12 - Nline DIV 2 - 3;
  IF TopLine < 1 THEN TopLine := 1;
  LeftCol := 40 - NCol DIV 2 - 3;
  IF LeftCol < 1 THEN LeftCol := 1;
```

```
    GotoXY(LeftCol,TopLine);    Write(',#201);
    FOR Col := 1 TO NCol+2 DO
    Write(#205);    Write(#187,');
    FOR Line := TopLine+1 TO TopLine+NLine+1 DO
        BEGIN
          GotoXY(LeftCol,line); Write(',#186,');
          GotoXY(LeftCol+NCol+3,Line);
          WriteLn(',#186,');
        END;
    GotoXY(LeftCol,TopLine+NLine+2); Write(',#200);
    FOR Col := 1 TO Ncol+2 DO
     Write(#205);    Write(#188,');
END;   { Display Board }

PROCEDURE DisplayFrame(NLine,NCol : Integer);
VAR  LeftCol,TopLine,Col,Line : Integer;
BEGIN
   IF NLine > 20 THEN Nline := 20;
   IF NCol  > 73 THEN NCol := 73;
   TopLine := 12 - Nline DIV 2 - 3;
   IF TopLine < 1 THEN TopLine := 1;
   LeftCol := 40 - NCol DIV 2 - 3;
   IF LeftCol < 1 THEN LeftCol := 1;
   GotoXY(LeftCol,TopLine);    Write(',#201);
   FOR Col := 1 TO Ncol+2 DO
    Write(#205);    Write(#187,');
   GotoXY(LeftCol,TopLine+1); Write(',#186,');
   GotoXY(LeftCol+NCol+3,TopLine+1); Write(',#186,');
   GotoXY(LeftCol,TopLine+2); Write(',#204,');
   FOR Col := 1 TO NCol+2 DO
```

```
    Write(#205);  Write(',#185,');
  FOR  Line  :=  TopLine+3  TO  TopLine+NLine+1  DO
      BEGIN
        GotoXY(LeftCol,line);  Write(',#186,');
        GotoXY(LeftCol+NCol+3,Line);
        WriteLn(',#186,');
      END;
  GotoXY(LeftCol,TopLine+NLine+2);  Write(',#204,');
  FOR  Col  :=  1  TO  Ncol+2  DO
    Write(#205);   Write(',#185,');
  GotoXY(LeftCol,TopLine+NLine+3);  Write(',#186,');
  GotoXY(LeftCol+NCol+3,TopLine+NLine+3);  Write(',#186,');
  GotoXY(LeftCol,TopLine+NLine+4);  Write(',#200);
  FOR  Col  :=  1  TO  Ncol+2  DO
    Write(#205);   Write(#188,');
END;  {  Display  Frame  }

PROCEDURE  InputDataT(VAR  TunitF  :  FiTunit);
VAR   TunitRec  :  ThermalUnit;
      option    :  char;
  PROCEDURE  PrepareT(VAR  TunitF  :  FiTunit);
  VAR   finame  :  String[10];
  BEGIN
      DisplayBoard(22,72);  GotoXY(12,8);
      Write('Thermal Unit Data - File Name : ');
      ReadLn(finame);
      Assign(TunitF,finame);
      {$I-}  Reset(TunitF);  {$I+}
      IF  Ioresult  <>  0  THEN
        BEGIN
```

```
        GotoXY(12,10);
        Write(' Create a NEW data file (Y/N) ? ');
        IF UpCase(ReadKey) = 'Y' THEN
            BEGIN
              GotoXY(12,12);
              WriteLn('Create =>> NEW data file ...');
              Rewrite(TunitF);
              Close(TunitF);
            END
        ELSE Halt;
      END; { IF IO }
END; { PrepareT }

FUNCTION GetDataT(VAR TunitRec : ThermalUnit) : Boolean;
VAR nber,k : Byte;
BEGIN
    Clrscr; DisplayBoard (22,72);
    k := 7; GotoXY(7,k);
    Write('Unit number : '); ReadLn(nber);
    IF nber = 0 THEN GetDataT := False
    ELSE
    WITH TunitRec DO
    BEGIN { TunitRec }
       number := nber; GotoXY(7,k+2);
       WriteLn(' Fuel Cost = a + bP + cP**2 ');
       WITH cfuel DO
            BEGIN
       GotoXY(7,k+3 ); Write('==> a = '); ReadLn(a);
       GotoXY(27,k+3); Write(' b = '); ReadLn(b);
       GotoXY(47,k+3); Write(' c = '); ReadLn(c);
```

```
            END;  { cfuel }
        GotoXY(7,k+5);
        WriteLn('Start Up Cost = b1( 1 - exp(-x/T) ) + b2 ');
        WITH cstart DO
            BEGIN
        GotoXY(7,k+6 ); Write('==> b1 =  ');  ReadLn(b1);
        GotoXY(27,k+6); Write('     b2 =  ');  ReadLn(b2);
        GotoXY(47,k+6); Write('     T  =  ');  ReadLn(d);
            END;  { cstart }
        GotoXY(7,k+8);
        Write('Minimum  power      > Pmin (mw) = ');
        ReadLn(pmin);    GotoXY(7,k+9);
        Write('Maximum  power      > Pmax (mw) = ');
        ReadLn(pmax);    GotoXY(7,k+10);
        Write('Minimum  Up  Time   > MinUp(hr) = ');
        ReadLn(minup);   GotoXY(7,k+11);
        Write('Minimum Down Time  > MinDn(hr) = ');
        ReadLn(mindown); GotoXY(7,k+12);
        Write('Status at t = 0    > X(0)      = ');
        ReadLn(status);
        GetDataT  :=  True;
    END;  {  TunitRec  }
END;  {  GetDataT  }

PROCEDURE  DataInputT(VAR  TunitF  :  FiTunit);
VAR   TunitRec  :  ThermalUnit;
BEGIN
    Reset(TunitF);
    Seek(TunitF,Filesize(TunitF));
    WHILE  GetDataT(TunitRec)  DO
```

```
    Write(TunitF,TunitRec);
    Close(TunitF);
END; { DataInputT }

PROCEDURE DispRecT(TunitRec : ThermalUnit;k : Byte);
BEGIN
 WITH TunitRec DO
  BEGIN
    GotoXY(7,k); WriteLn('Unit no.  ',number);
    WITH cfuel DO
     BEGIN
       GotoXY(7,k+1); Write('Fuel Cost    = ');
       WriteLn(a:9:2,' + ',b:9:4,' P  + ',c:9:6,' P**2');
     END;
    WITH cstart DO
     BEGIN
       GotoXY(7,k+2); Write('Start up Cost = ');
       WriteLn(b1:9:2,'( 1 - exp(-x/',d:4:1,')) + ',b2:7:2);
     END;
    GotoXY(7,k+3);
    Write('Min. Power    = ',pmin:6:1,' mw.');
    WriteLn('  Max. Power    = ',pmax:6:1,' mw.');
    GotoXY(7,k+4);
    Write('Min. Up Time  = ',minup:3,'  hr.');
    WriteLn('  Min. Down Time = ',mindown:3,'  hr.');
    GotoXY(7,k+5);
    WriteLn('Initial Status  x(0) = ',status:3);
  END; { TunitRec }
END; { DispRecT }
```

```
PROCEDURE DispDataT(VAR TunitF : FiTunit);
VAR TunitRec  : ThermalUnit;
     ch : Char;   cntr,k : byte;
BEGIN
 Clrscr;  DisplayBoard(22,72);
 Reset(TunitF);  cntr := 0; k := 3;
 WHILE NOT EOF(TunitF) DO
 BEGIN
    cntr := cntr + 1;
    Read(TunitF,TunitRec);
    GotoXY(6,k);  DispRecT(TunitRec,k);
    k := k + 7;
    IF cntr MOD 3 = 0 THEN
    Begin
       ch := ReadKey; k := 3;
       Clrscr;  DisplayBoard(22,72);
    End;
 END;
 Close(TunitF);  GotoXY(6,k+4);
 WriteLn(' Total Thermal Units = ',cntr:3,' unit ');
 ch := ReadKey;
END; { DispDataT }

FUNCTION GetDTch(VAR TunitRec : ThermalUnit) : Boolean;
VAR    nber,k : Byte;
BEGIN
    k := 10;  GotoXY(7,k);
    Write('Unit number : '); ReadLn(nber);
    IF nber = 0 THEN GetDTch := False
    ELSE
```

```
WITH TunitRec DO
BEGIN { TunitRec }
  number := nber;  GotoXY(7,k+1);
  Write('Change => Fuel Cost (Y/N) ? ');
  IF UpCase(ReadKey) = 'Y' THEN
   BEGIN { cfuel cost }
       GotoXY(7,k+2);
       WriteLn('Fuel Cost   = a + bP + cP**2 ');
       WITH cfuel DO
       BEGIN
         GotoXY(7,k+3 ); Write('==> a = '); ReadLn(a);
         GotoXY(27,k+3); Write('     b = '); ReadLn(b);
         GotoXY(47,k+3); Write('     c = '); ReadLn(c);
       END;
   END; { cfuel cost }
  GotoXY(7,k+4);
  Write(' Change => Start Up Cost (Y/N) ? ');
  IF UpCase(ReadKey) = 'Y' THEN
  BEGIN { cstart cost }
       GotoXY(7,k+5);
       WriteLn('Start Up Cost = b1( 1 - exp(-x/T) ) + b2 ');
       WITH cstart DO
       BEGIN
         GotoXY(7,k+6 ); Write(' ==> b1 = '); ReadLn(b1);
         GotoXY(27,k+6); Write('     b2 = '); ReadLn(b2);
         GotoXY(47,k+6); Write('     T  = '); ReadLn(d);
       END; { cstart }
  END; { cstart cost }
  GotoXY(7,k+8);
  Write(' Change => Min. & Max. Power (Y/N) ? ');
```

```
        IF UpCase(ReadKey) = 'Y' THEN
        BEGIN { min & max power }
            GotoXY(7,k+9); Write(' ==> Pmin (mw) = ');
            ReadLn(pmin); GotoXY(32,k+9);
            Write(' Pmax (mw) = '); ReadLn(pmax);
        END; { min & max power }
        GotoXY(7,k+10);
        Write(' Change => Min. Up & Down Time (Y/N) ? ');
        IF UpCase(ReadKey) = 'Y' THEN
        BEGIN { up & down time }
            GotoXY(7,k+11); Write('==> Min. Up(hr) = ');
            ReadLn(minup); GotoXY(32,k+11);
            Write(' Min. Dn(hr) = '); ReadLn(mindown);
        END; { up & down time }
        GotoXY(7,k+12);
        Write(' Change => Initial Status (Y/N) ? ');
        IF UpCase(ReadKey) = 'Y' THEN
        BEGIN
            GotoXY(7,k+13); Write(' ==> X(0) = ');
            ReadLn(status); END;
        GetDTch := True;
    END; { TunitRec }
END; { GetDataT }

PROCEDURE ChangeDataT(VAR TunitF : FiTunit);
VAR TunitRec : ThermalUnit;
    nber     : Byte;
    done     : Boolean;
BEGIN
 Clrscr; DisplayBoard(23,72);
```

```
  Reset(TunitF);    GotoXY(7,2);
  Write(' Unit number - to be corrected '); ReadLn(nber);
  WHILE nber <> 0 DO
  BEGIN
     done := False;  Seek(TunitF,0);
     WHILE NOT EOF(TunitF) AND NOT done DO
     BEGIN
       Read(TunitF,TunitRec);
       IF TunitRec.number = nber THEN
         BEGIN
            done := True; DispRecT(TunitRec,3);
            IF GetDTch(TunitRec) THEN
            BEGIN
              Seek(TunitF,FilePos(TunitF)-1);
              Write(TunitF,TunitRec);
              DispRecT(TunitRec,3);
            END; { IF Get }
         END; { IF Tunit }
     END; { WHILE NOT }
     GotoXY(7,24);
     Write(' unit number - to be corrected ');
     ReadLn(nber); Clrscr; DisplayBoard(23,72);
  END;  { WHILE }
  Close(TunitF);
END; { ChangeDataT }

PROCEDURE GetOptionT(VAR option : Char);
BEGIN
     Clrscr;  DisplayFrame(5,20);
     GotoXY(30,8);  WriteLn(' Thermal Unit Data ');
```

```
        GotoXY(30,10);  WriteLn('   I - Input      ');
        GotoXY(30,11);  WriteLn('   D - Display    ');
        GotoXY(30,12);  WriteLn('   C - Change     ');
        GotoXY(30,13);  WriteLn('   E - Exit       ');
        GotoXY(30,15);  WriteLn('   Option ==>>    ');
        REPEAT
          GotoXY(43,15);
          option := UpCase(ReadKey)
        UNTIL option in [' I ',' D ',' C ',' E' ];
        WriteLn(option);
    END; { GetOptionT }

BEGIN
    Clrscr;
    PrepareT(TunitF);
    REPEAT
        GetOptionT(option);
        Case option OF
         'I' : DataInputT(TunitF);
         'D' : DispDataT(TunitF);
         'C' : ChangeDataT(TunitF);
         'E' : WriteLn;
        END {case}
    UNTIL option = 'E';
END; { InputDataT }
{=============================================================}
PROCEDURE InputDataR(VAR ReqF : FiReq);
VAR   ReqRec  : Requirement;
        option    : char;
    PROCEDURE DataIputR(VAR ReqF : FiReq);
```

```
VAR   ReqRec  :  Requirement;
      ct      :  Byte;
    PROCEDURE Head;
    VAR     n  :  Byte;
    BEGIN
      DisplayBoard(15,50);
      GotoXY(25,1);   Write(' INPUT  Requirement  Data');
      GotoXY(17,3);   Write(' Period  [Hr] ');
      GotoXY(33,3);   Write(' Demand  [MW]');
      GotoXY(49,3);   Write(' Reserve  [MW]');
      GotoXY(13,4);   FOR  n  :=  1  TO  13  DO
                          Write('====');
    END; { Head }

BEGIN
  Clrscr;  Reset(ReqF);  Head;
  FOR  ct  :=  1  TO  24  DO
    BEGIN
      Seek(ReqF,Filesize(ReqF));
      WITH  ReqRec  DO
      BEGIN  { ReqRec }
            period  :=  ct;
            IF  ct  <  13  THEN
            Begin
              GotoXY(20,ct+5);  Write(period : 3);
              GotoXY(35,ct+5);  ReadLn(demand);
              GotoXY(53,ct+5);  ReadLn(reserve)
            End
            ELSE    Begin
                  GotoXY(20,ct-7);  Write(period : 3);
```

```
                    GotoXY(35,ct-7);  ReadLn(demand);
                    GotoXY(53,ct-7);  ReadLn(reserve);
                     End;
              IF  ct  =  12  THEN
              Begin
                Clrscr;   Head;
              End;
        END;  { ReqRec }
        Write(ReqF,ReqRec);
      END;
  Close(ReqF);
END;  { DataInputR }

PROCEDURE PrepareR(VAR ReqF : FiReq);
VAR   finame  :  String[10];
BEGIN
  DisplayBoard(22,72);   GotoXY(12,8);
  Write(' Demand & Reserve Requirement - File Name : ');
  ReadLn(finame);
  Assign(ReqF,finame);
  {$I-}  Reset(ReqF);  {$I+}
  IF   ioresult  <>  0  THEN
  BEGIN
    GotoXY(12,10);
    Write(' Create  a  NEW  data  File  (Y/N)  ?  ');
    IF  UpCase(ReadKey)  =  'Y'  THEN
    BEGIN
      GotoXY(12,12);
      WriteLn('  Create  >>>  NEW  data  file  ...');
      Rewrite(ReqF);  Close(ReqF);
```

```
        DataIputR(ReqF);
      END       ELSE  Halt;
    END; { IF IO }
  END; { PrepareR }

  PROCEDURE DispRecR(ReqRec : Requirement);
  BEGIN
    WITH ReqRec DO
    BEGIN
      Write(period:8);
      Write('|' : 8, demand : 9,'|' : 8);
      WriteLn(reserve : 9);
    END;
  END; { DispRecR }

  PROCEDURE DispRecRc(ReqRec : Requirement; k : Byte);
  BEGIN
    WITH ReqRec DO
    BEGIN
      GotoXY(8,k);     Write(' Period ', period : 4);
      GotoXY(8,k+1);  Write(' Demand  = ', demand : 6,' MW.');
      GotoXY(36,k+1); Write(' Reserve  = ', reserve : 6,' MW.');
    END;
  END; { DispRecRc }

  PROCEDURE DispDataR(VAR ReqF : FiReq);
  VAR    ReqRec  : Requirement;
         ch : Char;  cntr : byte;
    PROCEDURE Head;
    VAR  n : Byte;
```

```
  BEGIN
    DisplayBoard(15,50);
    GotoXY(17,3);    Write(' Period  [Hr]');
    GotoXY(33,3);    Write(' Demand  [MW]');
    GotoXY(49,3);    Write(' Reserve  [MW]');
    GotoXY(13,4);    FOR n := 1 TO 13 DO
                          Write('====');
  END; { Head }

BEGIN
    Clrscr;  Reset(ReqF);
    Head;    cntr := 0;
    WHILE NOT EOF(ReqF) DO
    BEGIN
       cntr := cntr + 1;
       Read(ReqF,ReqRec);
       GotoXY(14,5+cntr);
       DispRecR(ReqRec);
       IF cntr MOD 12 = 0 THEN
       Begin
            ch := ReadKey;
            Clrscr;  Head;
            cntr := 0;
       End;
    END;
    Close(ReqF);
END; { DispDataR }

PROCEDURE ChangeDataR(VAR ReqF : FiReq);
VAR   ReqRec  : Requirement;
```

```
    priod,k : Byte;   done : Boolean;
BEGIN
   Clrscr;   DisplayBoard(22,70);
   Reset(ReqF);   GotoXY(7,4);
   Write(' Time ( period ) - to be corrected ');
   ReadLn(priod);  k := 5;
   WHILE priod <> 0 DO
   BEGIN
      done := False;  Seek(ReqF,0);
      WHILE NOT EOF(ReqF) AND NOT done DO
      BEGIN
         Read(ReqF,ReqRec);
         IF ReqRec.period = priod THEN
         BEGIN
            done := True; DispRecRc(ReqRec,k);
            WITH ReqRec DO
            BEGIN
               k := k + 2;   GotoXY(8,k);
               Write(' Period  ', period : 4);
               GotoXY(8,k+1);  Write(' Demand [ MW ] = ');
               ReadLn(demand); GotoXY(36,k+1);
               Write('Reserve [ MW ] =  '); ReadLn(reserve);
            END;
            Seek(ReqF, FilePos(ReqF) - 1);
            Write(ReqF,ReqRec);
         END;
      END; { WHILE eof }
      IF k > 16 THEN
       Begin
         Clrscr;  DisplayBoard(22,70);
```

```
                GotoXY(7,4);   k := 5;
            End   ELSE    Begin
                              GotoXY(7,k+3);  k := k+4;
                          End;
          Write(' Time ( period ) - to be corrected ');
          ReadLn(priod);
        END;  { WHILE priod }
        Close(ReqF);
    END;  { ChangeDataR }

    PROCEDURE GetOptionR(VAR option : Char);
    BEGIN
        Clrscr;  DisplayFrame(5,20);
        GotoXY(30,8);  WriteLn(' Requirement Data ');
        GotoXY(30,10); WriteLn('    D - Display   ');
        GotoXY(30,11); WriteLn('    C - Change    ');
        GotoXY(30,13); WriteLn('    E - Exit      ');
        GotoXY(30,15); WriteLn('  Option ==>>     ');
          REPEAT
              GotoXY(42,15);
              option := UpCase(ReadKey)
          UNTIL option in ['D','C','E'];
        WriteLn(option);
    END;  { GetOptionR }

BEGIN  { InputDataR }
    Clrscr;
    PrepareR(ReqF);
    REPEAT
        GetOptionR(option);
```

```
        Case option OF
          'D' : DispDataR(ReqF);
          'C' : ChangeDataR(ReqF);
          'E' : WriteLn;
        END {case}
    UNTIL option = 'E';
END; { InputDataR }
{=============================================================}
PROCEDURE PtrTunitFi( VAR TunitF : FiTunit; VAR Tup : PtTunit;
                      VAR nunit : Byte);
VAR     tem,dum : PtTunit;
BEGIN
        Reset(TunitF);
        Tup := nil;   nunit := 0;
        WHILE NOT EOF(TunitF) DO
          Begin
              New(dum);  nunit := nunit + 1;
              IF Tup = nil THEN Tup := dum
              ELSE tem^.LinkT := dum;
              tem := dum;
              Read(TunitF,tem^.TunitRec);
          End;
        tem^.LinkT := nil;
        Close(TunitF);
END; { Pointer - Thermal Unit File }

PROCEDURE PtrReqFi( VAR ReqF : FiReq; VAR Rqp : PtReq;
            VAR nperiod : Byte);
VAR     tem,dum : PtReq;
BEGIN
```

```
        Reset(ReqF);
        Rqp := nil; nperiod := 0;
        WHILE NOT EOF(ReqF) DO
          Begin
              New(dum); nperiod := nperiod + 1;
              IF Rqp = nil THEN Rqp := dum
              ELSE tem^.LinkR := dum;
              tem := dum;
              Read(ReqF,tem^.ReqRec);
          End;
        tem^.LinkR := nil;
        Close(ReqF);
END; { Pointer - Requirement File }
{=============================================================}
PROCEDURE InitPUptr( VAR P : Power; VAR U : Commit;
                     nunit,nperiod : Byte);
VAR   p1,p2 : PowerPt;    i, t : Byte;
      u1,u2 : CommitPt;
BEGIN
  FOR i := 1 TO nunit DO
    Begin { Power }
      P[i] := nil;
      FOR t := 1 TO nperiod DO
        Begin
            New(p1);
            IF P[i] = nil THEN P[i] := p1
            ELSE p2^.LinkP := p1;
            p2 := p1;
            p2^.Pr := 0.0;
        End;
```

```
        p2^.LinkP  :=  nil;
      End;  { Power }
    FOR i := 1 TO nunit DO
      Begin { Commitment }
        U[i]  :=  nil;
        FOR  t  :=  1  TO  nperiod  DO
          Begin
              New(u1);
              IF  U[i]  =  nil  THEN  U[i]  :=  u1
              ELSE   u2^.LinkU  :=  u1;
              u2  :=  u1;
              u2^.Ur  :=  5;
          End;
        u2^.LinkU  :=  nil;
      End;  {  Commitment  }
END;  {  Initial  Power  &  Commitment  }
{=============================================================}
END.
```

```
UNIT    TSUBPROP;
INTERFACE
USES    CRT,INPUTDT;

TYPE    PathT       = ARRAY [ 1..MaxPeriod,-36..36 ] OF ShortInt;
        ValueT      = ARRAY [ -36..36,0..MaxPeriod ] OF Real;
        UnitState   = ARRAY [ 0..MaxPeriod,1..50 ] OF ShortInt;
        StateVector = ARRAY [ 0..MaxPeriod ] OF Byte;
        LgrMtiplier = ARRAY [ 1..MaxPeriod ] OF Real;
        TunitArry   = ARRAY [1..MaxUnit] OF Real;
VAR     lmd,mue     : LgrMtiplier;
        Vsub        : TunitArry;
PROCEDURE  CostMup ( VAR  tmp : PtTunit; VAR  Vxt,Pxt : ValueT;
                          lmd,mue   : LgrMtiplier; t,n1  : Byte;
                          VAR  Ustate : UnitState;VAR  Path : PathT );
PROCEDURE  CostMdn ( VAR  Vxt,Pxt  : ValueT; t,n1  : Byte;
                          VAR  Ustate : UnitState;VAR  Path : PathT );
PROCEDURE  CostUpDn ( VAR  Vxt,Pxt  : ValueT; t,n1  : Byte;
                           Ustate  : UnitState; VAR  Path  : PathT;
                           VAR  OneM  : StateVector );
PROCEDURE  CostDnUp ( VAR  tmp  : PtTunit; lmd,mue  : LgrMtiplier;
                           VAR  Vxt,Pxt  : ValueT; t,n1  : Byte;
                           Ustate  : UnitState; VAR  Path  : PathT;
                           VAR  OneP  : StateVector );
PROCEDURE  StateOfUnit ( VAR  tmp   : PtTunit; nperiod   : Byte;
                            VAR  Path  : PathT; lmd,mue  : LgrMtiplier;
                            VAR  Vxt,Pxt  : ValueT );
PROCEDURE  RecoveryPath ( VAR  pt  : PowerPt; VAR  ut  : CommitPt;
                             VAR  Vsub  : TunitArry; i,nperiod  : Byte;
                             Path  : PathT;  Vxt,Pxt  : ValueT );
```

```
PROCEDURE  TsubProblem ( VAR  P    : Power; VAR  U   : Commit;
                          VAR  Vsub : TunitArry; VAR  Tup  : PtTunit;
                          lmd,mue  : LgrMtiplier;  nperiod  :  Byte );
IMPLEMENTATION
{=============================================================}
PROCEDURE  CostMup ( VAR  tmp   : PtTunit; VAR  Vxt,Pxt  :  ValueT;
                     lmd,mue    : LgrMtiplier;  t,n1   :  Byte;
                     VAR  Ustate  :  UnitState; VAR  Path  :  PathT );
VAR     Vmn,Vmx,c1,c2  :  Real;
        xt1,xt   :  ShortInt;  t1   :  Byte;
        Pmn,Pmx          :  Real;
   FUNCTION  FuelCost1 ( VAR  tmp  :  PtTunit ) : Real;
   VAR  k1,k2,k3,p  :  Real;
   BEGIN
        k1  :=  tmp^.TunitRec.cfuel.a;
        k2  :=  tmp^.TunitRec.cfuel.b;
        k3  :=  tmp^.TunitRec.cfuel.c;
        p   :=  tmp^.TunitRec.pmin;
        FuelCost1  :=  k1 + ( k2*p ) + ( k3*p*p )
   END;  {  FuelCostPmin  }
   FUNCTION  FuelCost2 ( VAR  tmp  :  PtTunit ) : Real;
   VAR  k1,k2,k3,p  :  Real;
   BEGIN
        k1  :=  tmp^.TunitRec.cfuel.a;
        k2  :=  tmp^.TunitRec.cfuel.b;
        k3  :=  tmp^.TunitRec.cfuel.c;
        p   :=  tmp^.TunitRec.pmax;
        FuelCost2  :=  k1 + ( k2*p ) + ( k3*p*p )
   END;  {  FuelCostPmax  }
BEGIN
```

```
    t1    := t - 1;
    Ustate [ t,n1 ]  :=  Ustate [ t1,n1 ] + 1;
    c1     :=  FuelCost1 (tmp);
    c2     :=  FuelCost2 (tmp);
    Pmn  :=  tmp^.TunitRec.pmin;
    Pmx  :=  tmp^.TunitRec.pmax;
    xt1    :=  Ustate [t1,n1];
    xt     :=  Ustate [t,n1];
    Path[t,xt]  :=  xt1;
    Vmn  :=  c1 - ( lmd [t]*Pmn ) - ( mue [t]*Pmx );
    Vmx  :=  c2 - ( lmd [t]*Pmx ) - ( mue [t]*Pmx );
    IF  Vmn  <  Vmx
    THEN  Begin
                Vxt [xt,t]  :=  Vmn + Vxt [xt1,t1];
                Pxt [xt,t]  :=  Pmn
          End
    ELSE  Begin
                Vxt [xt,t]  :=  Vmx + Vxt [xt1,t1];
                Pxt [xt,t]  :=  Pmx
          End;
END;  {  CostMustUp  }

PROCEDURE   CostMdn ( VAR  Vxt,Pxt   :  ValueT; t,n1   :  Byte;
                           VAR Ustate : UnitState; VAR Path : PathT );
VAR   xt1,xt  :  ShortInt;   t1     :   Byte;
BEGIN
    t1  :=  t - 1;
    Ustate [t,n1]  :=  Ustate [t1,n1] - 1;
    xt1  :=  Ustate[t1,n1];
    xt    :=  Ustate[t,n1];
```

```
    Vxt [xt,t]  :=  Vxt [xt1,t1];
    Pxt [xt,t]  :=  0;
    Path [t,xt]  :=  xt1;
END;  {  CostMustDown  }

PROCEDURE   CostUpDn ( VAR  Vxt,Pxt   :  ValueT;  t,n1   :  Byte;
                            Ustate  :  UnitState; VAR  Path  :  PathT;
                            VAR  OneM  :  StateVector );
VAR    xt,xt1,k  :  ShortInt;
       t1        :  Byte;
BEGIN
   t1    :=  t - 1;
   xt1   :=  Ustate[t1,n1];
   xt    :=  -1;
   OneM [t]  :=  OneM [t] + 1;
   IF   OneM[t] = 1
   THEN    Begin
              Vxt [xt,t]  :=  Vxt [xt1,t1];
              Pxt [xt,t]  :=  0;
              Path [t,xt]  :=  xt1
           End
   ELSE    Begin
              IF  Vxt [xt1,t1]  <  Vxt [xt,t]
              THEN  Begin
                       Vxt [xt,t]   :=  Vxt [xt1,t1];
                       Path [t,xt]  :=  xt1;
                    End;
           End;
END;  {  CostUpToDown  }
```

```
PROCEDURE  CostDnUp ( VAR  tmp  :  PtTunit; lmd,mue  :  LgrMtiplier;
                           VAR  Vxt,Pxt  :  ValueT;  t,n1  :  Byte;
                           Ustate  :  UnitState;  VAR  Path  :  PathT;
                           VAR  OneP  :  StateVector );
VAR   Vmn,Vmx,Vm   :  Real;
      c1,c2,St,Temp   :  Real;
      xt1,xt,k        :  ShortInt;
      Pmn,Pmx         :  Real;
      t1              :  Byte;
  FUNCTION  FuelCost1 ( VAR  tmp  :  PtTunit ) : Real;
  VAR   k1,k2,k3,p  :  Real;
  BEGIN
      k1  :=  tmp^.TunitRec.cfuel.a;
      k2  :=  tmp^.TunitRec.cfuel.b;
      k3  :=  tmp^.TunitRec.cfuel.c;
      p   :=  tmp^.TunitRec.pmin;
      FuelCost1  :=  k1 + ( k2*p ) + ( k3*p*p )
  END;  { FuelCostPmin }
  FUNCTION  FuelCost2 ( VAR  tmp  :  PtTunit ) : Real;
  VAR   k1,k2,k3,p  :  Real;
  BEGIN
      k1  :=  tmp^.TunitRec.cfuel.a;
      k2  :=  tmp^.TunitRec.cfuel.b;
      k3  :=  tmp^.TunitRec.cfuel.c;
      p   :=  tmp^.TunitRec.pmax;
      FuelCost2  :=  k1 + ( k2*p ) + ( k3*p*p )
  END;  { FuelCostPmax }
  FUNCTION  StartCost ( VAR  tmp  :  PtTunit; Ustate  :  UnitState;
                           t,n1  :  Byte)  :  Real;
  VAR   k1,k2,k3,s  :  Real;
```

```
        x : ShortInt; t1 : Byte;
    BEGIN
        t1  := t - 1;
        k1  := tmp^.TunitRec.cstart.b1;
        k2  := tmp^.TunitRec.cstart.b2;
        s   := tmp^.TunitRec.cstart.d;
        x   := Ustate [t1,n1];
        k3  := 1 - Exp ( x/s );
        StartCost := ( k1*k3 ) + k2
    END; { StartUpCost }
BEGIN
    c1   := FuelCost1 (tmp);
    c2   := FuelCost2 (tmp);
    St   := StartCost (tmp,Ustate,t,n1);
    t1   := t - 1;
    xt1  := Ustate [t1,n1];
    xt   := 1;
    Pmn  := tmp^.TunitRec.pmin;
    Pmx  := tmp^.TunitRec.pmax;
    OneP [t] := OneP [t] + 1;
    Vmn  := c1 - ( lmd [t]*Pmn ) - ( mue [t]*Pmx );
    Vmx  := c2 - ( lmd [t]*Pmx ) - ( mue [t]*Pmx );
    IF  Vmn < Vmx
    THEN  Begin
              Vm  := Vmn;
              Pxt [xt,t] := Pmn
          End
    ELSE  Begin
              Vm  := Vmx;
              Pxt [xt,t] := Pmx
```

```
            End;
    IF  OneP [t]  =  1
    THEN    Begin
                Vxt [xt,t]  :=  Vm + St + Vxt [xt1,t1];
                Path [t,xt]  :=  xt1
            End
    ELSE    Begin
                Temp  :=  Vm + St + Vxt [xt1,t1];
                IF  Temp  <  Vxt [xt,t]
                THEN   Begin
                            Vxt [xt,t]  :=  Temp;
                            Path [t,xt]  :=  xt1;
                       End;
            End;
END; { CostDownToUp }

PROCEDURE   StateOfUnit ( VAR  tmp    : PtTunit; nperiod     : Byte;
                          VAR  Path  : PathT; lmd,mue  :  LgrMtiplier;
                          VAR  Vxt,Pxt  :  ValueT );
VAR    Nstate,OneP,OneM  :  StateVector;
       Ustate     :  UnitState;
       mu,md     :  ShortInt;
       n1,t,t1     :  Byte;
   PROCEDURE  PathInitial ( VAR  Path  :  PathT; nperiod  :  Byte);
   VAR   t  :  Byte;  xt  :  ShortInt;
   BEGIN
       FOR  t  :=  1  TO  nperiod  DO
       FOR  xt  :=  -36  TO  36  DO
       BEGIN
              Path[t,xt]  :=  0;
```

```
    END;
END; { PathInitialize }
PROCEDURE VxtInitial ( VAR Vxt : ValueT; nperiod : Byte);
VAR  t : Byte; xt : ShortInt;
BEGIN
    FOR xt := -12 TO 12 DO
        Vxt [xt,0] := 0.0;
    t := nperiod;
    FOR xt := -36 TO 36 DO
    Begin
        Vxt [xt,t] := 9.0E10;
    End;
END; { VxtInitialize }
PROCEDURE OnePMinit ( VAR OneP,OneM : StateVector;
                      nperiod : Byte);
VAR  t : Byte;
BEGIN
    FOR t := 1 TO nperiod DO
    BEGIN
        OneP [t] := 0;
        OneM [t] := 0
    END;
END; { OnePMinitialize }
PROCEDURE OnePMstate ( VAR Ustate : UnitState; t,n1 : Byte;
                       VAR Nstate,OneP,OneM : StateVector );
BEGIN
    IF OneP [t] >= 1
    THEN Begin
        n1 := n1 + 1;
        Ustate [t,n1] := 1
```

```
            End;
        IF  OneM [t]  >=  1
        THEN   Begin
                    n1  :=  n1 + 1;
                    Ustate [t,n1]  :=  -1
                End;
        Nstate [t]  :=  n1;
    END; { OnePMstate }
BEGIN
    PathInitial ( Path,nperiod );
    VxtInitial ( Vxt,nperiod );
    OnePMinit ( OneP,OneM,nperiod );
    Nstate [0]  :=  1;
    Ustate [0,1]  :=  tmp^.TunitRec.status;
    mu  :=  tmp^.TunitRec.minup;
    md  :=  -tmp^.TunitRec.mindown;
    FOR  t  :=  1  TO  nperiod  DO
    BEGIN  { Period }
        t1  :=  t - 1;
        FOR  n1  :=  1  TO  Nstate [t1]  DO
        BEGIN  { State }
            IF  ( Ustate [t1,n1]  >=  1 )  AND  ( Ustate [t1,n1]  <  mu )
            THEN  CostMup ( tmp,Vxt,Pxt,lmd,mue,t,n1,Ustate,Path )
            ELSE
                IF  ( Ustate [t1,n1]  <=  -1 )  AND  ( Ustate [t1,n1]  >  md )
                THEN  CostMdn ( Vxt,Pxt,t,n1,Ustate,Path )
                ELSE
                    IF  Ustate [t1,n1]  >=  1
                    THEN  Begin
                              CostUpDn ( Vxt,Pxt,t,n1,Ustate,Path,OneM );
```

```
                    CostMup ( tmp,Vxt,Pxt,lmd,mue,t,n1,Ustate,Path )
                  End
              ELSE Begin
                    CostDnUp ( tmp,lmd,mue,Vxt,Pxt,t,n1,
                    Ustate,Path,OneP );
                    CostMdn ( Vxt,Pxt,t,n1,Ustate,Path )
                  End;
      END; { State }
      OnePMstate ( Ustate,t,n1,Nstate,OneP,OneM );
  END; { Period }
END; { StateOfUnit }
{=============================================================}
PROCEDURE RecoveryPath ( VAR pt : PowerPt; VAR ut : CommitPt;
                         VAR Vsub : TunitArry; i,nperiod : Byte;
                         Path : PathT; Vxt,Pxt : ValueT );
VAR   ptem : LgrMtiplier; utem : StateVector;
      Xop,xt : ShortInt;   t : Byte;
BEGIN
  Vsub [I] := 9.0E10;
  t := nperiod;
  FOR xt := -36 TO 36 DO
  Begin
      IF Vx t[xt,t] < Vsub [i]
      THEN Begin
              Vsub [i] := Vxt[xt,t];
              Xop      := xt;
           End;
  End;
  FOR t := nperiod DOWNTO 1 DO
  Begin
```

```
        xt := Path [t,Xop];
        ptem [t] := Pxt [Xop,t];
        IF Xop > xt
        THEN utem [t] := 1
        ELSE utem [t] := 0;
        Xop := xt;
    End;
    t := 0;
    WHILE pt <> nil DO
    Begin
        t := t + 1;
        pt^.Pr := ptem [t];
        pt := pt^.LinkP;
        ut^.Ur := utem [t];
        ut := ut^.LinkU;
    End;
END; { Recovery Path - Save P,U }
{=============================================================}
PROCEDURE TsubProblem ( VAR P : Power; VAR U : Commit;
                        VAR Vsub : TunitArry; VAR Tup : PtTunit;
                        lmd,mue : LgrMtiplier; nperiod : Byte );
VAR Path : PathT; pt : PowerPt;
    Vxt,Pxt : ValueT; ut : CommitPt;
    i : Byte; tmp : PtTunit;
BEGIN
    tmp := Tup; i := 0;
    WHILE tmp <> nil DO
    Begin
        i := i + 1; GotoXY(28,15); Writeln(' Unit ',i:5 );
        pt := P [i];
```

```
        ut := U [i];
        StateOfUnit ( tmp,nperiod,Path,lmd,mue,Vxt,Pxt );
        RecoveryPath ( pt,ut,Vsub,i,nperiod,Path,Vxt,Pxt );
        tmp := tmp^.LinkT;
    end;
END;
{=============================================================}
END.
{*************************************************************}
UNIT   ECONDP;
INTERFACE
USES   CRT,INPUTDT,TSUBPROP;
TYPE   CostPt    =  ^CostNode;
       CostNode  =  RECORD
                      Cr     : Real;
                      LinkC  : CostPt
                      END;
VAR    Cts,Ctl   :  CostPt;
       Ect,Sct   :  Real;
PROCEDURE  EconDispat( VAR  Tup  :  PtTunit; VAR  Rqp  :  PtReq;
                  VAR  P  :  Power; U  :  Commit; lmd  :  LgrMtiplier );
PROCEDURE  CostEcon( VAR  Ect,Sct  :  Real; Tup  :  PtTunit;
                        VAR  Cts,Ctl  :  CostPt; P  :  Power );
PROCEDURE  OutputUC( U  :  Commit; P  :  Power; Cts,Ctl  :  CostPt;
                        Ect,Sct  :  Real; nunit  :  Byte );
IMPLEMENTATION
{=============================================================}
PROCEDURE  EconDispat( VAR  Tup  :  PtTunit; VAR  Rqp  :  PtReq;
                  VAR  P  :  Power; U  :  Commit; lmd  :  LgrMtiplier );
VAR    tmT  :  PtTunit;   pt  :  Power;
```

```
    tmR : PtReq;     ut : Commit;
    t,i : Byte; k : Word; ch : char;
    Psum,Dm,Ed,Err : Real;
PROCEDURE PowerComput( VAR tmT : PtTunit; VAR pt : Power;
                       lmd : LgrMtiplier; i,t : Byte );
VAR k1,k2,tem,pm,px : Real;
BEGIN
    k1 := tmT^.TunitRec.cfuel.b;
    k2 := tmT^.TunitRec.cfuel.c;
    tem := (lmd[t] - k1)/(2*k2);
    pt[i]^.Pr := tem;
    pm := tmT^.TunitRec.pmin;
    px := tmT^.TunitRec.pmax;
    IF tem < pm
    THEN pt[i]^.Pr := pm;
    IF tem > px
    THEN pt[i]^.Pr := px
END; { Power Compute }
PROCEDURE UpdatLmd( VAR lmd : LgrMtiplier;
            Dm,Psum : Real; t : Byte; k : Word);
VAR tem,r : Real;
    a,b   : Integer;
BEGIN
    a := 100; b := 10;
    r := 1/(a + (b*k));
    tem := Dm - Psum;
    lmd[t] := lmd[t] + (r*tem);
END; { Update Lampda }

BEGIN
```

```
Err := 0.25;
pt := P;  ut := U;
tmR := Rqp;  t := 0;
WHILE tmR <> nil DO
 Begin
     k := 0; t := t + 1;
     REPEAT
       k := k + 1; Psum := 0;
       tmT := Tup;   i := 0;
       WHILE tmT <> nil DO
       Begin
            i := i + 1;
            IF ut[i]^.Ur = 1
            THEN Begin
                      PowerComput(tmT,pt,lmd,i,t);
                      Psum := Psum + pt[i]^.Pr;
                 End;
            tmT := tmT^.LinkT;
       End; { nunit }
       Dm := tmR^.ReqRec.demand;
       Ed := Abs(Psum - Dm);
       IF Ed > Err
       THEN UpdatLmd(lmd,Dm,Psum,t,k);
     UNTIL ( Ed <= Err ) OR ( k >= 2000 );
     tmR := tmR^.LinkR;
     tmT := Tup; i := 0;
     WHILE tmT <> nil DO
     Begin
       i := i + 1;
       pt[i] := pt[i]^.LinkP;
```

```
            ut[i] := ut[i]^.LinkU;
            tmT := tmT^.LinkT;
          End;
    End; { nperiod }
END; { Economic Dispatch }
{=============================================================}
PROCEDURE CostEcon( VAR Ect,Sct : Real; Tup : PtTunit;
                        VAR Cts,Ctl : CostPt; P : Power );
VAR   c1,c2,c3,st,ct : Real;
      r1,r2,r3,r4,po : Real;
      s1,s2,t1,t2    : CostPt;
      Ecost,Scost    : TunitArry;
      tmp   : PtTunit;   x : ShortInt;
      pt    : PowerPt;   i : Byte;
BEGIN
   Ect := 0;  Sct := 0;
   tmp := Tup;  i := 0;
   WHILE tmp <> nil DO
   Begin  { while tmp }
      i  := i + 1;  Scost [i] := 0;
      pt := P[i];   Ecost [i] := 0;
      x  := tmp^.TunitRec.status;
      c1 := tmp^.TunitRec.cfuel.a;
      c2 := tmp^.TunitRec.cfuel.b;
      c3 := tmp^.TunitRec.cfuel.c;
      r1 := tmp^.TunitRec.cstart.b1;
      r2 := tmp^.TunitRec.cstart.b2;
      r3 := tmp^.TunitRec.cstart.d;
      WHILE pt <> nil DO
      Begin { while pt }
```

```
        IF  pt^.Pr  >  0
        THEN    Begin  { if }
                    IF  x  <  0
                    THEN   Begin
                               po  :=  pt^.Pr;
                               r4  :=  1 - Exp(x/r3);
                               st  :=  (r1*r4)  +  r2;
                               ct  :=  c1 + (c2*po) + (c3*po*po);
                               x  :=  1;
                               Scost[i]  :=  Scost[i]  +  st;
                               Ecost[i]  :=  Ecost[i]  +  ct  +  st;
                           End
                    ELSE   Begin
                               po  :=  pt^.Pr;
                               ct  :=  c1 + (c2*po) + (c3*po*po);
                               x  :=  x  +  1;
                               Ecost[i]  :=  Ecost[i]  +  ct;
                           End
                End  { if }
        ELSE   Begin
                    IF  x  <  0
                    THEN   x  :=  x - 1
                    ELSE   x := -1
                End;
        pt  :=  pt^.LinkP;
    End;   { while pt }
    Sct  :=  Sct  +  Scost[i];
    Ect  :=  Ect  +  Ecost[i];
    tmp  :=  tmp^.LinkT;
End;   { while tmp }
```

```
    tmp := Tup;    i := 0;
    Ctl := nil; Cts := nil;
    WHILE tmp <> nil DO
     Begin
        i := i + 1;
        New(s2);
        IF  Cts = nil THEN Cts := s2
        ELSE  s1^.LinkC := s2;
        s1 := s2;
        s1^.Cr := Scost[i];
        New(t2);
        IF  Ctl = nil THEN Ctl := t2
        ELSE  t1^.LinkC := t2;
        t1 := t2;
        t1^.Cr := Ecost[i];
        tmp := tmp^.LinkT;
     End;
   s1^.LinkC := nil;
   t1^.LinkC := nil;
END; {  Total Cost => Economic Dispatch }
{=============================================================}
PROCEDURE  OputCost ( Cts,Ctl : CostPt; Ect,Sct : Real );
VAR  cs,ct   : CostPt;  t,i,k : Byte;
       Cf,Fct : Real;    ch   : Char;
   PROCEDURE HeadCost;
   VAR  t : Byte;
   BEGIN
       GotoXY(36,3);  WriteLn('C O S T');
       GotoXY(9,5);   FOR t := 1 TO 15 DO
                          Write('****');  WriteLn;
```

```
        GotoXY(13,6);   Write('Plant');
        GotoXY(25,6);   Write('Fuel  Cost');
        GotoXY(40,6);   Write('StartUp Cost');
        GotoXY(57,6);   Write('Total  Cost');
        GotoXY(9,7);    FOR  t  :=  1  TO  15  DO
                                Write('----');  WriteLn;
    END; { Head Cost }

BEGIN
    Clrscr;   DisplayBoard(21,60);
    HeadCost;   k  :=  7;   i  :=  0;
    cs  :=  Cts;    ct  :=  Ctl;
    WHILE   cs  <>  nil  DO
      Begin
        k  :=  k + 1;   i  :=  i  +  1;
        GotoXY(12,k);   Write('Unit',i:3);
        cf  :=  ct^.Cr  -  cs^.Cr;
        GotoXY(25,k);    Write(cf:10:0);
        GotoXY(40,k);    Write(cs^.Cr:10:0);
        GotoXY(57,k);    Write(ct^.Cr:10:0);
        IF  i  MOD  12  =  0  THEN
          Begin
              ch  :=  ReadKey;  Clrscr;
              DisplayBoard(21,60);
              HeadCost;   k  :=  7;
          End;
        cs  :=  cs^.LinkC;   ct  :=  ct^.LinkC;
      End;
    k  :=  k + 2;      Fct  :=  Ect  -  Sct;
    GotoXY(9,k);      FOR  t  :=  1  TO  15  DO
```

```
                    Write('=====');
  GotoXY(13,k+1);   Write('TOTAL');
  GotoXY(25,k+1);   Write(Fct:10:0);
  GotoXY(40,k+1);   Write(Sct:10:0);
  GotoXY(57,k+1);   Write(Ect:10:0);
  GotoXY(9,k+2);    FOR t := 1 TO 15 DO
                          Write('=====');
  ch := ReadKey;
END; { Output Cost }
PROCEDURE OutputUC( U : Commit; P : Power; Cts,Ctl : CostPt;
                                  Ect,Sct : Real; nunit : Byte );
VAR   op : Char;
  PROCEDURE OputCommit ( U : commit; nunit : Byte );
  VAR  k,i,t : Byte; ch : Char;
       ut    : CommitPt;
    PROCEDURE HeadUcom;
    VAR  t : Byte;
    BEGIN
        GotoXY(32,3); WriteLn('UNIT COMMITMENT');
        GotoXY(16,5); Write('**********');
        GotoXY(26,5); FOR t := 1 TO 12 DO
                          Write('***'); WriteLn;
        GotoXY(17,6); Write('Period');
        GotoXY(25,6); FOR t := 1 TO 12 DO
                          Write(t:3);  WriteLn;
        GotoXY(25,7); FOR t := 13 TO 24 DO
                          Write(t:3);  WriteLn;
        GotoXY(16,8);     Write('--------');
        GotoXY(26,8); FOR t := 1 TO 12 DO
                          Write('---');  WriteLn;
```

```
    END; { Head Unit Commitment }
BEGIN
    Clrscr; DisplayBoard(21,48);
    HeadUcom;  k := 7;
    FOR i := 1 TO nunit DO
      Begin
          k := k + 2;
          ut := U[i];
          GotoXY(17,k);  Write('Unit',i:3);
          GotoXY(25,k);  FOR t := 1 TO 12 DO
                          Begin
                              Write(ut^.Ur:3);
                              ut := ut^.LinkU;
                          End;  WriteLn;
          GotoXY(25,k+1);  FOR t := 13 TO 24 DO
                          Begin
                              Write(ut^.Ur:3);
                              ut := ut^.LinkU;
                          End;  WriteLn;
          IF i MOD 7 = 0 THEN
            Begin
                ch := ReadKey; Clrscr; k := 7;
                DisplayBoard(21,48); HeadUcom;
            End;
      End; { nunit }
    k := k + 2;
    GotoXY(16,k); Write('**********');
    GotoXY(26,k); FOR t := 1 TO 12 DO
                          Write('***'); WriteLn;
                          ch := ReadKey;
```

```
END;  { Output Commitment }
PROCEDURE  OputPower ( P  :  Power; nunit  :  Byte );
VAR    pt    :  PowerPt;
       k,i,t  :  Byte;     ch  :  Char;
    PROCEDURE  HeadPower;
    VAR   t  :  Byte;
    BEGIN
          GotoXY(28,3);  WriteLn(' ECONOMIC DISPATCH');
          GotoXY(4,5);   Write('**********');
          GotoXY(14,5);  FOR  t  :=  1  TO  12  DO
                              Write('*****');  WriteLn;
          GotoXY(5,6);    Write('Period');
          GotoXY(13,6);  FOR  t  :=  1  TO  12  DO
                              Write(t:5);   WriteLn;
          GotoXY(13,7);  FOR  t  :=  13  TO  24  DO
                              Write(t:5);   WriteLn;
          GotoXY(4,8);  Write('----------');
          GotoXY(14,8);  FOR  t  :=  1  TO  12  DO
                              Write('-----');   WriteLn;
    END;  { Head Power }
BEGIN
    Clrscr;   DisplayBoard(22,70);
    HeadPower;   k  :=  7;
    FOR  i  :=  1  TO  nunit  DO
       Begin
          k  :=  k  +  2;
          pt  :=  P[i];
          GotoXY(5,k);    Write('Unit',i:3);
          GotoXY(13,k);   FOR  t  :=  1  TO  12  DO
                              Begin
```

```
                                Write(pt^.Pr:5:0);
                                pt := pt^.LinkP;
                            End; WriteLn;
            GotoXY(13,k+1); FOR t := 13 TO 24 DO
                            Begin
                                Write(pt^.Pr:5:0);
                                pt := pt^.LinkP;
                            End; WriteLn;
          IF i MOD 7 = 0 THEN
            Begin
                ch := ReadKey; Clrscr; k := 7;
                DisplayBoard(22,70); HeadPower;
            End;
        End; { nunit }
      k := k + 3;
      GotoXY(4,k);   Write('**********');
      GotoXY(14,k); FOR t := 1 TO 12 DO
                            Write('*****');
      ch := ReadKey;
END; { Output Power }
PROCEDURE GetOutput ( VAR op : Char );
BEGIN
      Clrscr;   DisplayFrame(5,22);
      GotoXY(31,8);   WriteLn('UNIT COMMITMENT');
      GotoXY(31,10);  WriteLn('U - UC Schedule');
      GotoXY(31,11);  WriteLn('P - Econ. Dispatch');
      GotoXY(31,12);  WriteLn('C - Cost ');
      GotoXY(31,13);  WriteLn('E - Exit ');
      GotoXY(32,15);  WriteLn('Option ==>>');
      REPEAT
```

```
            GotoXY(45,15);
            op := Upcase(ReadKey)
          UNTIL op in ['U','P','C','E'];
          WriteLn(op);
      END; { GetOutput }

BEGIN  { Output UC }
    REPEAT
          GetOutput(op);
          Case op OF
            'U' :  OputCommit(U,nunit);
            'P' :  OputPower(P,nunit);
            'C' :  OputCost(Cts,Ctl,Ect,Sct);
            'E' :  WriteLn;
          END { case }
    UNTIL op = 'E';
END; { OutPut UC }
{=============================================================}
END.
```

ภาคผนวก ข

วิธีการใช้โปรแกรมยูนิตคอมมิตเมนต์

โปรแกรมคำนวณยูนิตคอมมิตเมนต์ โดยใช้วิธีรีแล็กชันแบบลากรองจ์นี้ ได้พัฒนาโดยใช้ภาษา Turbo Pascal ver. 7.0 วิธีการใช้โปรแกรมอาจแบ่งตามลักษณะการทำงานของโปรแกรมได้เป็น 3 ส่วน ดังนี้

ข1. การป้อนข้อมูล

เริ่มต้นโดยใส่แผ่นดิสก์ลงในไดรว์ A: แล้วป้อนชื่อโปรแกรม ดังรูปที่ ข1

```
A : \ > UCLR
```

รูปที่ ข1. แสดงการเรียกโปรแกรมยูนิตคอมมิตเมนต์

ข1.1 ข้อมูลเครื่องกำเนิดไฟฟ้า

เมื่อ ENTER แล้วจะได้ข้อความดังแสดงในรูปที่ ข2 ซึ่งเป็นการให้ป้อนชื่อแฟ้มข้อมูลสำหรับข้อมูลของเครื่องกำเนิดไฟฟ้า ซึ่งอาจจะเป็นแฟ้มข้อมูลที่มีอยู่หรือเปิดแฟ้มข้อมูลใหม่

```
Thermal Unit Data - File name : _
```

รูปที่ ข2. ป้อนชื่อแฟ้มข้อมูลของเครื่องกำเนิดไฟฟ้า

เมื่อป้อนชื่อที่ไม่ตรงกับชื่อแฟ้มข้อมูลที่มีอยู่ โปรแกรมจะถือว่าต้องการเปิดแฟ้มข้อมูลใหม่ และจะถามย้ำอีกครั้งเพื่อเป็นการป้องกันในกรณีที่อาจป้อนชื่อแฟ้มข้อมูลผิด ซึ่งถ้าต้องการเปิดแฟ้มข้อมูลใหม่ให้ป้อน Y ดังรูปที่ ข3

```
Create a NEW data file (Y/N) ? Y
Create ==>> NEW data file .....
```

รูปที่ ข3. เปิดแฟ้มข้อมูลใหม่สำหรับข้อมูลเครื่องกำเนิดไฟฟ้า

ส่วนกรณีต้องการใช้แฟ้มข้อมูลที่ใช้ในการวิจัยครั้งนี้ ให้เลือกชื่อแฟ้มข้อมูลดังนี้ คือ " DATA10 "," DATA53 "," DATA110 " ซึ่งเป็นชื่อแฟ้มข้อมูลของเครื่องกำเนิดไฟฟ้าสำหรับ ระบบไฟฟ้ากำลังขนาด 10 ยูนิต ขนาด 53 ยูนิต และขนาด 110 ยูนิต ตามลำดับ เมื่อป้อนชื่อแฟ้มข้อมูลแล้ว ENTER จะได้เมนูดังรูปที่ ข4

```
Thermal Unit Data
I - Input
D - Display
C - Change
E - Exit
Option ==>> _
```

รูปที่ ข4 แสดงเมนูของข้อมูลเครื่องกำเนิดไฟฟ้า

ในเมนูสำหรับข้อมูลของเครื่องกำเนิดไฟฟ้า มีหัวข้อย่อยให้เลือกดังนี้

"I" สำหรับการป้อนข้อมูลในแฟ้มข้อมูลใหม่ หรือเป็นการเพิ่มเติมข้อมูลในแฟ้มข้อมูลเดิม

"D" สำหรับการแสดงข้อมูล ที่มีอยู่ในแฟ้มข้อมูล

"C" สำหรับการแก้ไขข้อมูลเดิมที่มีอยู่

"E" สำหรับการออกจากเมนูของข้อมูลเครื่องกำเนิดไฟฟ้า เพื่อไปยังส่วนข้อมูลค่าความต้องการและกำลังผลิตสำรองในระบบ

เมื่อเลือกแต่ละตัวเลือก จะมีรายละเอียดดังนี้

ตัวเลือก "I"

เป็นการป้อนข้อมูลของเครื่องกำเนิดไฟฟ้า ข้อมูลที่ป้อนจะประกอบด้วย 4 ส่วน คือ

1) Unit number

หมายเลขยูนิตให้ป้อนค่าเป็นตัวเลขจำนวนเต็มบวก ENTER แล้วได้ดังรูปที่ ข5(1)

- กรณีเป็นแฟ้มข้อมูลใหม่ให้ป้อนค่าเรียงลำดับ ตั้งแต่ 1,2,3,.....
- กรณีเป็นแฟ้มข้อมูลเดิมให้ป้อนค่าเรียงลำดับ ต่อจากจำนวนยูนิตเดิม
- กรณีเสร็จสิ้นการป้อนข้อมูล ให้ป้อนเลข "0" แล้ว ENTER

```
Unit number : 1
Fuel Cost   =  a + bP + cP**2
==> a =
```

รูปที่ ข5(1) ป้อนข้อมูลเครื่องกำเนิดไฟฟ้า

2) Fuel cost function

เป็นการป้อนค่าสัมประสิทธ์ของ Fuel cost function ซึ่งเป็นสมการกำลังสอง ป้อนค่าเป็นเลขจำนวนจริงแล้ว ENTER จะได้ดังรูปที่ ข5(2)

```
Unit number : 1
Fuel Cost       =   a + bP + cP**2
==>   a = 820        b = 9.023        c = 0.00113
Start Up Cost   =   b1 ( 1 - exp (-x/T) ) + b2
==>   b1 =
```

รูปที่ ข5(2) ป้อนข้อมูลเครื่องกำเนิดไฟฟ้า

3) Start up cost function

เป็นการป้อนค่าสัมประสิทธ์ของ Start up cost function ซึ่งเป็นสมการแบบเอ็กโปเนนเชียล ป้อนค่าเป็นเลขจำนวนจริงแล้ว ENTER ดังรูปที่ ข5(3)

```
Unit number : 1
Fuel Cost      =  a + bP + cP**2
==> a = 820        b = 9.023        c = 0.00113
Start Up Cost  =  b1( 1 - exp (-x/T) ) + b2
==> b1 = 2050          b2 = 825          T = 4
Minimum power    > Pmin(mw) =
```

รูปที่ ข5(3) ป้อนข้อมูลเครื่องกำเนิดไฟฟ้า

4) ขีดจำกัดของเครื่องกำเนิดไฟฟ้า

- Minimum & Maximum Power

  ป้อนค่าขีดจำกัดของกำลังการผลิต เป็นเลขจำนวนจริง

- Minimum Up & Down Time

  ป้อนค่าขีดจำกัดเวลาในการเดินเครื่องและหยุดเดินเครื่อง เป็นเลขจำนวนเต็ม

- Status เริ่มต้น

  ป้อนค่าสถานะของครื่องกำเนิดไฟฟ้าในเวลาเริ่มต้น เป็นเลขจำนวนเต็ม

ป้อนข้อมูลแล้ว ENTER จะได้ดังรูปที่ ข5(4) เมื่อแล้วเสร็จจะกลับไปหมายเลขยูนิต ใน 1) อีกเป็นวงรอบ

```
Unit number : 1
Fuel Cost      =  a + bP + cP**2
==> a = 820        b = 9.023        c = 0.00113
Start Up Cost  =  b1 ( 1 - exp (-x/T) ) + b2
==> b1 = 2050      b2 = 825      T = 4
Minimum power        >  Pmin(mw)  =  300
Maximum power        >  Pmax(mw)  =  1000
Minimum Up time      >  MinUp(hr) =  5
Minimum Down time    >  MinDn(hr) =  4
Status at  t = 0     >  x (0)     =  -4
```

รูปที่ ข5(4) ป้อนข้อมูลเครื่องกำเนิดไฟฟ้า

ตัวเลือก "D"

เป็นการแสดงข้อมูลของเครื่องกำเนิดไฟฟ้าที่มีอยู่ในแฟ้มข้อมูล จะแสดงข้อมูลครั้งละ 3 ยูนิต เมื่อ Enter จะแสดงข้อมูลชุดต่อไป ดังรูปที่ ข6

```
Unit no.  1
Fuel cost      =  820.00 + 9.0230 P + 0.001130 P**2
Start up cost  =  2050.00( 1 - exp(-x / 4.0) ) + 825.00
Min Power      =  300.0 mw.  Max Power      =  1000.0 mw
Min Up Time    =   5  hr.    Min Down Time  =   4  hr
Initial Status    x(0) = 4
Unit no.  2
Fuel cost      =  400.00 + 7.6540 P + 0.001600 P**2
Start up cost  =  1460.00( 1 - exp(-x / 3.0) ) + 650.00
Min Power      =  130.0 mw.  Max Power      =  400.0 mw
Min Up Time    =   3  hr.    Min Down Time  =   2  hr
Initial Status    x(0) = 5
Unit no.  3
Fuel cost      =  600.00 + 8.7520 P + 0.001470 P**2
Start up cost  =  2100.00( 1 - exp(-x / 4.0) ) + 950.00
Min Power      =  165.0 mw.  Max Power      =  600.0 mw
Min Up Time    =   2  hr.    Min Down Time  =   4  hr
Initial Status    x(0) = 1
```

รูปที่ ข6 แสดงข้อมูลของเครื่องกำเนิดไฟฟ้าในระบบ

ตัวเลือก "C"

เป็นการแก้ไขข้อมูลของเครื่องกำเนิดไฟฟ้า ให้ป้อนหมายเลขยูนิตที่ต้องการแก้ไข เช่น หมายเลข 8 ENTER โปรแกรมจะแสดงข้อมูลของ Unit no. 8 เพื่อตรวจสอบอีกครั้ง ดังรูปที่ ข7(1)

```
Unit number  -  to be corrected   8
Unit no.  8
Fuel cost        =   400.00 + 7.7620P +  0.001710 P**2
Start Up Cost  =  1370.00( 1  -  exp (-x/3.0) )  +  550.00
Min  Power    =  110  mw     Max  Power        =  375.0   mw
Min Up Time  =   1    hr       Min Down Time  =    3    hr
Initial  Status     x(0)   =  -1

Unit number  :
```

รูปที่ ข7(1) แสดงการแก้ไขข้อมูลเครื่องกำเนิดไฟฟ้า

ถ้าไม่ต้องการแก้ไขข้อมูล ป้อนเลข "0" ที่ Unit number แล้ว ENTER ถ้าต้องการแก้ไขข้อมูลป้อนหมายเลขยูนิต ENTER แล้วเลือกแก้ไขข้อมูลที่ต้องการด้วยการกด "Y" จึงจะป้อนข้อมูลดังรูปที่ ข7(2)

```
Unit  number  -  to be corrected   8
Unit no   8
Fuel  Cost     =  400.00  +  7.7620 P  +  0.001710 P**2
Start Up Cost  =  1370.00( 1  -  exp (-x/3.0) )  +  550.00
Min  Power    =  110 mw    Max  Power        =  375.0  mw
Min Up Time  =   1   hr      Min Down Time  =    3    hr
Initial  Status     x(0)   =  -1
Unit number  :  8
Change  =>  Fuel  Cost  (Y/N)   ?  N

Change  =>  Start  Up  Cost  (Y/N)   ?  Y
=>  b1  =
```

รูปที่ ข7(2) แสดงการแก้ไขข้อมูลเครื่องกำเนิดไฟฟ้า

ตัวเลือก "E"

เป็นการออกจากเมนูของข้อมูลเครื่องกำเนิดไฟฟ้า เพื่อไปยังส่วนข้อมูลค่าความต้องการ และกำลังผลิตสำรองในระบบ

ข1.2 ข้อมูลความต้องการและกำลังผลิตสำรองในระบบ

เมื่อออกจากเมนูของข้อมูลเครื่องกำเนิดไฟฟ้า จะได้ข้อความดังรูปที่ ข8 จะเป็นการให้ป้อนชื่อแฟ้มข้อมูลที่มีอยู่หรือเปิดแฟ้มข้อมูลใหม่

Demand & Reserve Requirement_File name :_

รูปที่ ข8 ป้อนชื่อแฟ้มข้อมูลของความต้องการและกำลังผลิตสำรอง

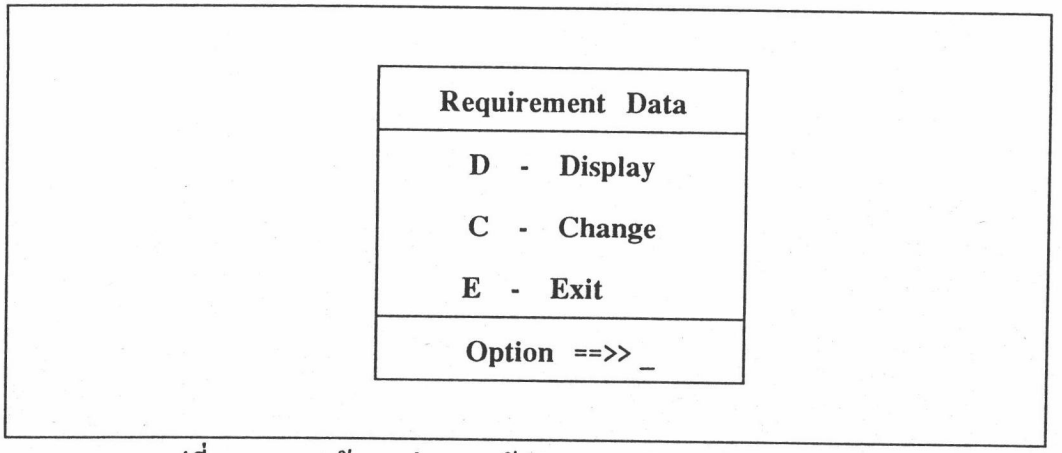
ทำนองเดียวกับการป้อนชื่อแฟ้มข้อมูลของเครื่องกำเนิดไฟฟ้า ในกรณีเปิดแฟ้มข้อมูลใหม่จะได้ดังรูปที่ ข3 เมื่อได้เปิดแฟ้มข้อมูลใหม่แล้ว การป้อนข้อมูลจะป้อนเฉพาะค่าความต้องการ (Demand) และกำลังผลิตสำรอง (Reserve) เป็นเลขจำนวนเต็มบวก ดังรูปที่ ข9

INPUT Requirement Data

Period [hr]	Demand [mw]	Reserve [mw]
1	1025	85
2	1000	85
3		

รูปที่ ข9 ป้อนข้อมูลความต้องการและกำลังผลิตสำรอง

ส่วนกรณีที่ต้องการใช้แฟ้มข้อมูลที่ใช้ในการวิจัยครั้งนี้ ให้เลือกชื่อแฟ้มข้อมูล ดังนี้ คือ "DR10" , "DR53" , "DR110" ซึ่งเป็นชื่อแฟ้มข้อมูลความต้องการและกำลังผลิตสำรองสำหรับระบบขนาด 10 ยูนิต 53 ยูนิต และ 110 ยูนิตตามลำดับ เมื่อป้อนชื่อแล้ว ENTER จะได้เมนูดังรูปที่ ข10



รูปที่ ข10 เมนูข้อมูลค่าความต้องการและกำลังผลิตสำรอง

ในเมนูสำหรับข้อมูลของความต้องการและกำลังผลิตสำรอง มีหัวข้อย่อยให้เลือกดังนี้

" D " สำหรับการแสดงข้อมูล ที่มีอยู่ในแฟ้มข้อมูล

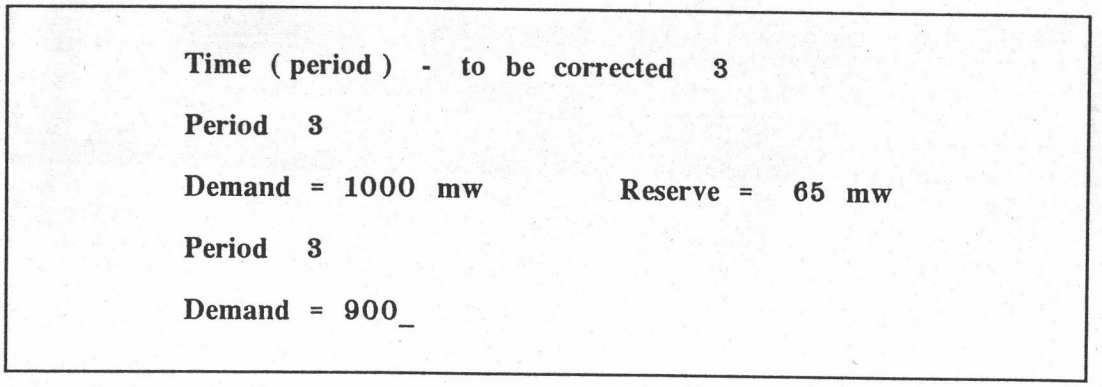
" C " สำหรับการแก้ไขข้อมูลเดิมที่มีอยู่

" E " สำหรับการออกจากเมนูของข้อมูลค่าความต้องการ เป็นการเสร็จสิ้นในส่วนการป้อนข้อมูล จะไปยังส่วนการรันโปรแกรมต่อไป

เมื่อเลือกแต่ละตัวเลือก จะมีรายละเอียดดังนี้

ตัวเลือก " C "

ให้ป้อนคาบเวลาที่ต้องการแก้ไขข้อมูล เช่น คาบเวลาที่ 3 โปรแกรมจะแสดงข้อมูลของคาบเวลาที่ 3 เพื่อการตรวจสอบอีกครั้ง ให้ป้อนค่าที่ต้องการแก้ไขเป็นเลขจำนวนเต็มบวก ดังรูปที่ ข11 (1)



รูปที่ ข11(1) แสดงการแก้ไขข้อมูลความต้องการและกำลังผลิตสำรอง

เมื่อไม่ต้องการแก้ไขข้อมูลแล้ว ป้อนเลข "0" ENTER ดังรูปที่ ข11(2)

```
Time ( period ) - to be corrected      0_
```

รูปที่ ข11(2) แสดงการสิ้นสุดการแก้ไขข้อมูลความต้องการ

ตัวเลือก "D"

จะแสดงข้อมูลที่มีอยู่ในแฟ้มข้อมูล โดยแสดงครั้งละ 12 ชั่วโมง เมื่อ ENTER จะแสดงข้อมูลชุดต่อไป ดังรูปที่ ข12

Period [Hr]	Demand [mw]	Reserve [mw]
1	1025	85
2	1000	85
3	900	65
4	850	55
5	1025	85
6	1400	110
7	1970	165
8	2400	190
9	2850	210
10	3150	230
11	3300	250
12	3400	275

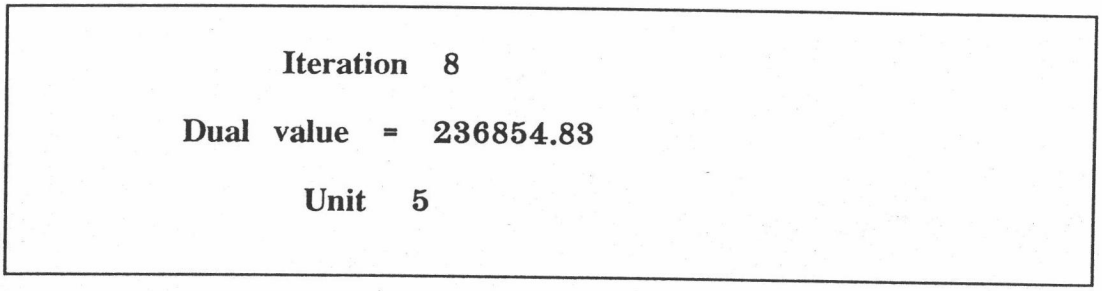
รูปที่ ข12 แสดงข้อมูลความต้องการและกำลังผลิตสำรองในระบบ

ข2 <u>การรันโปรแกรม</u>

การคำนวณของโปรแกรมจะเป็น 2 ขั้นตอน คือ

ข2.1 <u>คำนวณปัญหาดูอัล</u>

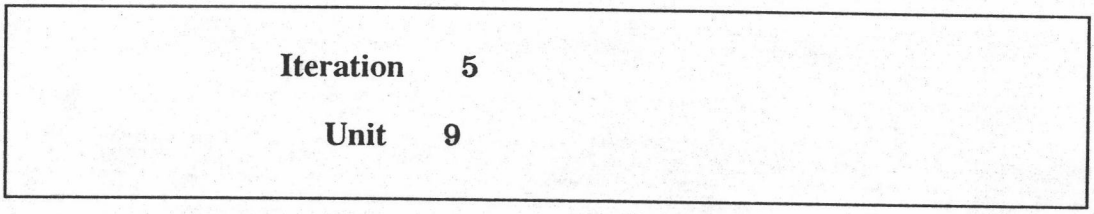
ในระหว่างการคำนวณปัญหาดูอัล จะแสดงผลค่าดูอัลในแต่ละ Iteration และจะแสดงด้วยว่า ขณะนั้นกำลังคำนวณปัญหาย่อยของยูนิตใด ดังรูปที่ ข13



รูปที่ ข13 แสดงระหว่างการคำนวณปัญหาดูอัล

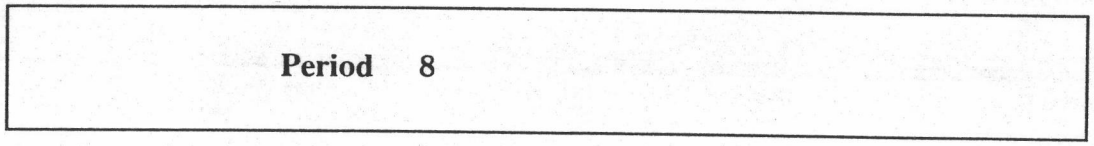
ข2.2 <u>คำนวณผลลัพธ์ปัญหาพรีมัล</u>

เมื่อเสร็จสิ้นการคำนวณปัญหาดูอัล จะเป็นการตรวจสอบเงื่อนไขบังคับกำลังผลิตสำรอง ซึ่งถ้าผลลัพธ์ไม่สอดคล้องกับเงื่อนไขดังกล่าว โปรแกรมจะปรับค่า multiplier แล้วคำนวณปัญหาย่อย ในระหว่างการคำนวณจะแสดงจำนวน Iteration และยูนิตที่กำลังคำนวณปัญหาย่อย ดังรูปที่ ข14



รูปที่ ข14 ระหว่างการปรับค่าให้สอดคล้องกับเงื่อนไขบังคับร่วมกัน

เมื่อผลลัพธ์สอดคล้องกับเงื่อนไขบังคับกำลังผลิตสำรอง จะเป็นการคำนวณการจ่ายโหลดอย่างประหยัด จะแสดงคาบเวลาที่คำนวณ ดังรูปที่ ข15



รูปที่ ข15 คำนวณการจ่ายโหลดอย่างประหยัด

ข3 <u>การแสดงผลลัพธ์</u>

เมื่อเสร็จสิ้นการคำนวณ จะได้เมนูดังรูปที่ ข16

UNIT COMMITMENT
U - UC Schedule
P - Econ. Dispatch
C - Cost
E - Exit
Option ==> _

รูปที่ ข16 เมนูสำหรับแสดงผลลัพธ์ปัญหายูนิตคอมมิตเมนต์

เมื่อเลือกแต่ละตัวเลือก จะแสดงผลลัพธ์ดังนี้

ตัวเลือก "U"

เป็นการแสดงสถานะการคอมมิตเมนต์ของเครื่องกำเนิดไฟฟ้า โดย"1" หมายถึง เดินเครื่อง ส่วน "0" หมายถึง หยุดเดินเครื่อง ดังรูปที่ ข17

UNIT COMMITMENT

Period	1	2	3	4	5	6	7	8	9	10	11	12
	13	14	15	16	17	18	19	20	21	22	23	24
Unit 1	0	0	0	0	0	0	0	0	0	1	1	1
	1	1	1	1	1	1	1	1	0	0	0	0
Unit 2	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1
Unit 3	1	0	0	0	0	0	0	1	1	1	1	1
	1	1	1	1	1	1	1	1	0	0	0	0
Unit 4	0	0	0	0	0	0	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	0	0	0	0
Unit 5	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0

รูปที่ ข17 แสดงสถานะการคอมมิตเมนต์ของเครื่องกำเนิดไฟฟ้า

ตัวเลือก "P"

เป็นการแสดงปริมาณการจ่ายโหลดของเครื่องกำเนิดไฟฟ้าในระบบ ดังรูปที่ ข18

ECONOMIC DISPATCH

Period	1	2	3	4	5	6	7	8	9	10	11	12
	13	14	15	16	17	18	19	20	21	22	23	24
Unit 1	0	0	0	0	0	0	0	0	0	424	475	510
	467	355	300	300	300	441	476	338	0	0	0	0
Unit 2	400	400	400	387	400	400	400	400	400	400	400	400
	400	400	400	400	400	400	400	400	400	400	400	400
Unit 3	184	0	0	0	0	0	0	296	468	418	458	484
	451	365	296	252	303	431	458	352	0	0	0	0
Unit 4	0	0	0	0	0	0	410	397	420	420	420	420
	420	420	397	354	404	420	420	420	0	0	0	0

รูปที่ ข18 แสดงปริมาณการจ่ายโหลดของเครื่องกำเนิดไฟฟ้า

ตัวเลือก "C"

เป็นการแสดงค่าใช้จ่ายในการผลิต ซึ่งประกอบด้วยค่าใช้จ่ายของเชื้อเพลิง ค่าใช้จ่ายเริ่มเดินเครื่อง และค่าใช้จ่ายรวมในการผลิตไฟฟ้าของแต่ละยูนิต และผลรวมทั้งระบบ แสดงดังรูปที่ ข19

COST

****

Plant	Fuel Cost	StartUp Cost	Total Cost
Unit 1	50641	2796	53437
Unit 2	89104	0	89104
Unit 3	57060	2581	59641
Unit 4	57818	1930	59748
Unit 5	0	0	0
Unit 6	40887	1610	42497
Unit 7	0	0	0
Unit 8	67473	1735	69208
Unit 9	118874	0	118874
Unit 10	34540	1804	36344
TOTAL	516399	12455	528854

รูปที่ ข19 แสดงต้นทุนค่าใช้จ่ายในการผลิตไฟฟ้า

ตัวเลือก "E"

เป็นการออกจากการใช้งานโปรแกรมยูนิตคอมมิตเมนต์

ภาคผนวก ค

ข้อมูลระบบไฟฟ้ากำลัง ขนาด 10 ยูนิต

ข้อมูลที่ใช้ในการคำนวณค่ายูนิตคอมมิตเมนต์ ของระบบไฟฟ้ากำลัง ขนาด 10 ยูนิต ประกอบด้วย

ตาราง ค1 ข้อมูลเงื่อนไขของ เครื่องกำเนิดไฟฟ้า
ตาราง ค2 ข้อมูลฟังก์ชันค่าเชื้อเพลิง ในการเดินเครื่อง
ตาราง ค3 ข้อมูลฟังก์ชันค่าใช้จ่าย ในการเริ่มเดินเครื่อง
ตาราง ค4 ข้อมูลความต้องการกำลังไฟฟ้า และ กำลังผลิตสำรอง ที่เดินเครื่องอยู่ในระบบ

ตาราง ค1 ข้อมูลเงื่อนไขของ เครื่องกำเนิดไฟฟ้า

	Pmin (MW)	Pmax (MW)	Min. Up Time (hr)	Min. Down Time (hr)	Initial Status (hr)
Unit 1	300	1000	5	4	-4
Unit 2	130	400	3	2	5
Unit 3	165	600	2	4	1
Unit 4	130	420	1	3	-2
Unit 5	225	700	4	5	-8
Unit 6	50	200	2	2	-1
Unit 7	250	750	3	4	6
Unit 8	110	375	1	3	-1
Unit 9	275	850	4	3	2
Unit 10	75	250	2	1	-7

ตาราง ค2 ข้อมูลฟังก์ชันค่าเชื้อเพลิง ในการเดินเครื่อง

$[\ C(P) = a + bP + cP^2\ ]$

	a	b	c
Unit 1	820.0	9.023	0.00113
Unit 2	400.0	7.654	0.00160
Unit 3	600.0	8.752	0.00147
Unit 4	420.0	8.431	0.00150
Unit 5	540.0	9.223	0.00234
Unit 6	175.0	7.054	0.00515
Unit 7	600.0	9.121	0.00131
Unit 8	400.0	7.762	0.00171
Unit 9	725.0	8.162	0.00128
Unit 10	200.0	8.149	0.00452

ตาราง ค3 ข้อมูลฟังก์ชันค่าใช้จ่าย ในการเริ่มเดินเครื่อง

$\{\ S(X) = b1\ [\ 1 - \exp[\ -x / T\ ]\ ] + b2\ \}$

	b1	b2	T
Unit 1	2050.0	825.0	4.0
Unit 2	1460.0	650.0	3.0
Unit 3	2100.0	950.0	4.0
Unit 4	1480.0	650.0	4.0
Unit 5	2100.0	900.0	3.0
Unit 6	1360.0	750.0	2.0
Unit 7	2300.0	950.0	4.0
Unit 8	1370.0	550.0	3.0
Unit 9	2200.0	950.0	4.0
Unit 10	1180.0	625.0	2.0

ตาราง ค4 ข้อมูลความต้องการกำลังไฟฟ้า และ
กำลังผลิตสำรองที่เดินเครื่องอยู่ในระบบ

Time (hr)	Demand (MW)	Reserve (MW)	Time (hr)	Demand (MW)	Reserve (MW)
1	1025	85	13	3275	240
2	1000	85	14	2950	210
3	900	65	15	2700	200
4	850	55	16	2550	195
5	1025	85	17	2725	200
6	1400	110	18	3200	220
7	1970	165	19	3300	250
8	2400	190	20	2900	210
9	2850	210	21	2125	170
10	3150	230	22	1650	130
11	3300	250	23	1300	100
12	3400	275	24	1150	90

ภาคผนวก ง

ข้อมูลระบบไฟฟ้ากำลัง ขนาด 53 ยูนิต

ข้อมูลที่ใช้ในการคำนวณค่ายูนิตคอมมิตเมนต์ ของระบบไฟฟ้ากำลัง ขนาด 53 ยูนิต ได้มาจากตัวอย่าง ข้อมูลกำลังการผลิตของ Scenario A ในรายงาน EPRI [19] โดยได้ปรับปรุงเพิ่มเติมข้อมูลบางส่วนให้เหมาะสมกับการคำนวณค่ายูนิตคอมมิตเมนต์ ข้อมูลประกอบด้วย

ตาราง ง1 ข้อมูลเงื่อนไขของ เครื่องกำเนิดไฟฟ้า
ตาราง ง2 ข้อมูลฟังก์ชันค่าเชื้อเพลิง ในการเดินเครื่อง
ตาราง ง3 ข้อมูลฟังก์ชันค่าใช้จ่าย ในการเริ่มเดินเครื่อง
ตาราง ง4 ข้อมูลความต้องการกำลังไฟฟ้า และ กำลังผลิตสำรอง ที่เดินเครื่องอยู่ในระบบ

ตาราง ง1 ข้อมูลเงื่อนไขของ เครื่องกำเนิดไฟฟ้า

	Capacity (MW)	Pmin (MW)	Pmax (MW)	Min. Up Time (hr)	Min. Down Time (hr)
Unit 1	1200	600	1200	12	12
Unit 2	1000	500	1000	12	12
Unit 3	800	320	800	12	12
Unit 4-7	600	240	600	9	9
Unit 8-12	400	160	400	7	7
Unit 13-29	200	80	200	4	4
Unit 30-53	50	20	50	2	2

ตาราง ง2 ข้อมูลฟังก์ชันค่าเชื้อเพลิง ในการเดินเครื่อง

$[\ C(P) = a + bP + cP^2\ ]$

	a	b	c
Unit 1	756.70	3.2842	0.000638
Uint 2	630.45	3.2846	0.000765
Unit 3	916.42	8.2409	0.001392
Unit 4	648.94	8.7805	0.001363
Unit 5	648.94	8.8805	0.001363
Unit 6	648.94	8.9805	0.001363
Unit 7	648.94	9.0805	0.001363
Unit 8	414.62	8.7377	0.002564
Unit 9	414.62	8.8377	0.002564
Unit 10	414.62	8.9377	0.002564
Unit 11	722.08	15.2084	0.004466
Unit 12	722.08	15.3084	0.004466
Unit 13	249.29	9.0515	0.005510
Unit 14	249.29	9.8515	0.005510
Unit 15	249.29	9.1515	0.005510
Unit 16	249.29	9.9515	0.005510
Unit 17	249.29	9.2515	0.005510
Unit 18	249.29	10.0515	0.005510
Unit 19	249.29	9.3515	0.005510
Unit 20	249.29	10.1515	0.005510
Unit 21	249.29	9.4515	0.005510
Unit 22	249.29	10.2515	0.005510
Unit 23	249.29	9.5515	0.005510
Unit 24	249.29	10.3515	0.005510
Unit 25	249.29	9.6515	0.005510
Unit 26	249.29	10.4515	0.005510

ตาราง ง2 (ต่อ)

	a	b	c
Unit 27	249.29	9.7515	0.005510
Uint 28	433.08	15.7178	0.009584
Unit 29	433.08	15.8178	0.009584
Unit 30	72.16	10.4808	0.025520
Unit 31	72.16	11.6808	0.025520
Unit 32	72.16	10.5808	0.025520
Unit 33	72.16	11.7808	0.025520
Unit 34	72.16	10.6808	0.025520
Unit 35	72.16	11.8808	0.025520
Unit 36	72.16	10.7808	0.025520
Unit 37	72.16	11.9808	0.025520
Unit 38	72.16	10.8808	0.025520
Unit 39	72.16	12.0808	0.025520
Unit 40	72.16	10.9808	0.025520
Unit 41	72.16	12.1808	0.025520
Unit 42	72.16	11.0808	0.025520
Unit 43	72.16	12.2808	0.025520
Unit 44	72.16	11.1808	0.025520
Unit 45	72.16	12.3808	0.025520
Unit 46	72.16	11.2808	0.025520
Unit 47	72.16	12.4808	0.025520
Unit 48	72.16	11.3808	0.025520
Unit 49	72.16	12.5808	0.025520
Unit 50	72.16	11.4808	0.025520
Unit 51	72.16	12.6808	0.025520
Unit 52	72.16	11.5808	0.025520
Unit 53	72.16	12.7808	0.025520

ตาราง ง3 ข้อมูลฟังก์ชันค่าใช้จ่าย ในการเริ่มเดินเครื่อง

$$\{ S(X) = b1 [1 - \exp [- x / T]] + b2 \}$$

	b1	b2	T
Unit 1	2000	1000	10
Unit 2	1800	1000	10
Unit 3	1500	950	8
Unit 4-7	1200	900	8
Unit 8-10	840	650	7
Unit 11-12	1400	650	7
Unit 13-27	780	700	5
Unit 28-29	1300	700	5
Unit 30-53	660	450	5

ตาราง ง4 ข้อมูลความต้องการกำลังไฟฟ้า และ
กำลังผลิตสำรองที่เดินเครื่องอยู่ในระบบ

Time (hr)	Demand (mw)	Reserve (mw)
1	6277	502
2	6123	490
3	6100	488
4	6015	480
5	6086	487
6	6570	525
7	7388	590
8	8806	704
9	9739	779
10	10016	800
11	10200	816
12	10060	804

Time (hr)	Demand (mw)	Reserve (mw)
13	10030	802
14	9563	765
15	9898	792
16	9767	780
17	9365	749
18	9345	747
19	9034	722
20	9444	755
21	9748	780
22	9412	753
23	8730	698
24	7985	638

ภาคผนวก จ

ข้อมูลระบบไฟฟ้ากำลัง ขนาด 110 ยูนิต

ข้อมูลที่ใช้ในการคำนวณค่ายูนิตคอมมิตเมนต์ ของระบบไฟฟ้ากำลังขนาด 110 ยูนิต ได้มาจากตัวอย่าง ข้อมูลกำลังการผลิตของ Scenario A ในรายงาน EPRI [19] โดยได้ปรับปรุงเพิ่มเติมข้อมูลบางส่วนให้เหมาะสมกับการคำนวณค่ายูนิตคอมมิตเมนต์ ข้อมูลประกอบด้วย

ตาราง จ1 ข้อมูลเงื่อนไขของ เครื่องกำเนิดไฟฟ้า
ตาราง จ2 ข้อมูลฟังก์ชันค่าเชื้อเพลิง ในการเดินเครื่อง
ตาราง จ3 ข้อมูลฟังก์ชันค่าใช้จ่าย ในการเริ่มเดินเครื่อง
ตาราง จ4 ข้อมูลความต้องการกำลังไฟฟ้า และ กำลังผลิตสำรอง ที่เดินเครื่องอยู่ในระบบ

ตาราง จ1 ข้อมูลเงื่อนไขของ เครื่องกำเนิดไฟฟ้า

	Capacity (MW)	Pmin (MW)	Pmax (MW)	Min. Up Time (hr)	Min. Down Time (hr)
Unit 1-2	1200	600	1200	12	12
Unit 3-5	1000	500	1000	12	12
Unit 6-7	800	320	800	12	12
Unit 8-19	600	240	600	9	9
Unit 20-34	400	160	400	7	7
Unit 35-86	200	80	200	4	4
Unit 87-110	50	20	50	2	2

ตาราง จ2 ข้อมูลฟังก์ชันค่าเชื้อเพลิง ในการเดินเครื่อง

$[\ C(P) = a + bP + cP^2\ ]$

	a	b	c
Unit 1	756.70	3.2842	0.000638
Uint 2	756.70	3.3842	0.000638
Unit 3	630.45	3.2846	0.000765
Unit 4	630.45	3.3846	0.000765
Unit 5	630.45	3.4846	0.000765
Unit 6	916.42	8.2409	0.001392
Unit 7	916.42	8.3409	0.001392
Unit 8	648.94	8.7805	0.001363
Unit 9	648.94	8.8805	0.001363
Unit 10	648.94	8.9805	0.001363
Unit 11	648.94	9.0805	0.001363
Unit 12	648.94	9.1805	0.001363
Unit 13	648.94	9.2805	0.001363
Unit 14	648.94	9.3805	0.001363
Unit 15	648.94	9.4805	0.001363
Unit 16	648.94	9.5805	0.001363
Unit 17	648.94	9.6805	0.001363
Unit 18	648.94	9.7805	0.001363
Unit 19	648.94	9.8805	0.001363
Unit 20	414.62	8.7377	0.002564
Unit 21	414.62	8.8377	0.002564
Unit 22	414.62	8.9377	0.002564
Unit 23	414.62	9.0377	0.002564
Unit 24	414.62	9.1377	0.002564
Unit 25	414.62	9.2377	0.002564
Unit 26	414.62	9.3377	0.002564

ตาราง จ2 (ต่อ)

	a	b	c
Unit 27	414.62	9.4377	0.002564
Uint 28	414.62	9.5377	0.002564
Unit 29	414.62	9.6377	0.002564
Unit 30	722.08	15.2084	0.004466
Unit 31	722.08	15.1084	0.004466
Unit 32	722.08	15.0084	0.004466
Unit 33	722.08	14.9084	0.004466
Unit 34	722.08	14.8084	0.004466
Unit 35	249.29	9.0515	0.005510
Unit 36	249.29	9.1515	0.005510
Unit 37	249.29	9.2515	0.005510
Unit 38	249.29	9.3515	0.005510
Unit 39	249.29	9.4514	0.005510
Unit 40	249.29	9.5515	0.005510
Unit 41	249.29	9.6515	0.005510
Unit 42	249.29	9.7515	0.005510
Unit 43	249.29	9.8515	0.005510
Unit 44	249.29	9.9515	0.005510
Unit 45	249.29	10.0515	0.005510
Unit 46	249.29	10.1515	0.005510
Unit 47	249.29	10.2515	0.005510
Unit 48	249.29	10.3515	0.005510
Unit 49	249.29	10.4515	0.005510
Unit 50	249.29	10.5515	0.005510
Unit 51	249.29	10.6515	0.005510
Unit 52	249.29	10.7515	0.005510
Unit 53	249.29	10.8515	0.005510
Unit 54	249.29	10.9515	0.005510

ตาราง จ2 (ต่อ)

	a	b	c
Unit 55	249.29	11.0515	0.005510
Uint 56	249.29	11.1515	0.005510
Unit 57	249.29	11.2515	0.005510
Unit 58	249.29	11.3515	0.005510
Unit 59	249.29	11.4515	0.005510
Unit 60	249.29	11.5515	0.005510
Unit 61	249.29	11.6515	0.005510
Unit 62	249.29	11.7515	0.005510
Unit 63	249.29	11.8515	0.005510
Unit 64	249.29	11.9515	0.005510
Unit 65	249.29	12.0515	0.005510
Unit 66	249.29	12.1515	0.005510
Unit 67	249.29	12.2515	0.005510
Unit 68	249.29	12.3515	0.005510
Unit 69	249.29	12.4515	0.005510
Unit 70	249.29	12.5515	0.005510
Unit 71	249.29	12.6515	0.005510
Unit 72	249.29	12.7515	0.005510
Unit 73	249.29	12.8515	0.005510
Unit 74	249.29	12.9515	0.005510
Unit 75	249.29	13.0515	0.005510
Unit 76	249.29	13.1515	0.005510
Unit 77	249.29	13.2515	0.005510
Unit 78	249.29	13.3515	0.005510
Unit 79	249.29	13.4515	0.005510
Unit 80	249.29	13.5515	0.005510
Unit 81	433.08	15.7178	0.009584
Unit 82	433.08	15.8178	0.009584

ตาราง จ2 (ต่อ)

	a	b	c
Unit 83	433.08	15.9178	0.009584
Uint 84	433.08	15.6178	0.009584
Unit 85	433.08	15.5178	0.009584
Unit 86	433.08	15.4178	0.009584
Unit 87	72.16	10.4808	0.025520
Unit 88	72.16	10.5808	0.025520
Unit 89	72.16	10.6808	0.025520
Unit 90	72.16	10.7808	0.025520
Unit 91	72.16	10.8808	0.025520
Unit 92	72.16	10.9808	0.025520
Unit 93	72.16	11.0808	0.025520
Unit 94	72.16	11.1808	0.025520
Unit 95	72.16	11.2808	0.025520
Unit 96	72.16	11.3808	0.025520
Unit 97	72.16	11.4808	0.025520
Unit 98	72.16	11.5808	0.025520
Unit 99	72.16	11.6808	0.025520
Unit 100	72.16	11.7808	0.025520
Unit 101	72.16	11.8808	0.025520
Unit 102	72.16	11.9808	0.025520
Unit 103	72.16	12.0808	0.025520
Unit 104	72.16	12.1808	0.025520
Unit 105	72.16	12.2808	0.025520
Unit 106	72.16	12.3808	0.025520
Unit 107	72.16	12.4808	0.025520
Unit 108	72.16	12.5808	0.025520
Unit 109	72.16	12.6808	0.025520
Unit 110	72.16	12.7808	0.025520

ตาราง จ3 ข้อมูลฟังก์ชันค่าใช้จ่าย ในการเริ่มเดินเครื่อง

{ S (X) = b1 [1 - exp [- x / T]] + b2 }

	b1	b2	T
Unit 1-2	2000	1000	10
Unit 3-5	1800	1000	10
Unit 6-7	1500	950	8
Unit 8-19	1200	900	8
Unit 20-29	840	650	7
Unit 30-34	1400	650	7
Unit 35-80	780	700	5
Unit 81-86	1300	700	5
Unit 87-110	660	450	5

ตาราง จ4 ข้อมูลความต้องการกำลังไฟฟ้า และ
กำลังผลิตสำรองที่เดินเครื่องอยู่ในระบบ

Time (hr)	Demand (MW)	Reserve (MW)	Time (hr)	Demand (MW)	Reserve (MW)
1	17218	1034	13	27513	1650
2	16796	1008	14	26234	1574
3	16732	1004	15	27152	1629
4	16500	990	16	26793	1607
5	16695	1002	17	25691	1541
6	18022	1081	18	25635	1538
7	20266	1216	19	24782	1487
8	24158	1449	20	25906	1554
9	26715	1603	21	26740	1604
10	27476	1648	22	25820	1549
11	27980	1679	23	23948	1437
12	27596	1656	24	21906	1314

ประวัติผู้เขียน

นาย คิดชาย อุณหศิริกุล เกิดวันที่ 18 สิงหาคม พ.ศ. 2504 ที่ อำเภอเมือง จังหวัดพังงา สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ในปีการศึกษา 2526 ปัจจุบันทำงานที่การไฟฟ้าฝ่ายผลิตแห่งประเทศไทย อำเภอบางกรวย จังหวัดนนทบุรี และได้รับการสนับสนุนจากหน่วยงานต้นสังกัด ให้เข้าศึกษาต่อปริญญาโท ในภาควิชาวิศวรรมไฟฟ้า ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2535