```
C    THIS FORTRAN CODE IS FOR SOLVING ORDINARY MINIMUM COST NETWORK FLOW
C    PROBLEM WITH SEPARABLE CONVEX LINEAR/QUADRATIC ARC COSTS,
C    UPPER BOUNDS ON ARC FLOWS (LOWER BOUNDS ARE ASSUMED TO BE ZERO),
C    AND NODE SUPPLIES/DEMANDS:
C
C      MIN   A'*X + X'*C*X  S.T.  E*X = S,  0 <= X <= U,
C
C    WHERE C IS A DIAGONAL MATRIX WITH NONNEGATIVE DIAGONAL ENTRIES,
C    E IS THE NODE-ARC INCIDENCE MATRIX,
C    A IS THE VECTOR OF LINEAR COST COEFFICIENTS,
C    S IS THE VECTOR OF NODE SUPPLIES/DEMANDS,
C    U IS THE VECTOR OF ARC UPPER BOUNDS.
C    THESE ARE SUPPLIED BY THE USER.
C    X IS THE DECISION VECTOR OF ARC FLOWS.
C
C    THIS CODE IMPLEMENTS THE EPSILON-RELAXATION METHOD.
C    THE METHOD AND NUMERICAL TEST RESULTS ARE DESCRIBED IN THE PAPERS:
C      D.P. Bertsekas, L.C. Polymenakos, and P. Tseng,
C      "Epsilon-Relaxation Method for Separable Convex
C      Cost Network Flow Problems", SIAM Journal on Optimization,
C      Vol. 7 (1997), 853--870.
C
C      D.P. Bertsekas, L.C. Polymenakos, and P. Tseng,
C      "Epsilon-Relaxation Method and Auction Methods for Separable
C      Convex Cost Network Flow Problems,''
C      in Proceedings of International Conference on Network Optimization,
C      edited by P. M. Pardalos, D. W. Hearn, and W. W. Hager,
C      Springer-Verlag, Berlin, 1997, 103--126.
C
C    FOR ANY QUESTIONS, CONTACT   DIMITRI P. BERTSEKAS (dimitrib@mit.edu)
C                         OR   PAUL TSENG (tseng@math.washington.edu).
C    THIS CODE IS PLACED IN THE PUBLIC DOMAIN TO BE USED FOR
C    RESEARCH PURPOSES.  FOR ANY OTHER PURPOSES, PLEASE CONTACT
C    DIMITRI P. BERTSEKAS OR PAUL TSENG.
C
C    THIS VERSION MAINTAINS SEPARATE LISTS FOR QUADRATIC AND
C    LINEAR COST ARCS. FLOW IS FIRST PUSHED ON LINEAR ARCS
C    THEN ON QUADRATIC.
C    Modified to push flow on arcs violating EPS/2-CS on March 5, 1995.
C
C    Version 1: November, 2005.
C    ************************************************
C
      PARAMETER (MAXNODES=5000, MAXARCS=20000)
      IMPLICIT INTEGER (A-Z)

      INTEGER NXTQUEUE(MAXNODES)
      INTEGER STARTN(MAXARCS)
      INTEGER ENDN(MAXARCS)
      INTEGER FIN(MAXNODES)
      INTEGER FOUT(MAXNODES)
      INTEGER NXTIN(MAXARCS)
      INTEGER NXTOU(MAXARCS)
      INTEGER FPUSHF(MAXNODES)
      INTEGER NXTPUSHF(MAXARCS)
      INTEGER FPUSHB(MAXNODES)
      INTEGER NXTPUSHB(MAXARCS)
      INTEGER QFPUSHF(MAXNODES),QNXTPUSHF(MAXARCS)
      INTEGER QFPUSHB(MAXNODES),QNXTPUSHB(MAXARCS)
      INTEGER DUMMY, DUM1, DUM2
```

```
      REAL*8 A(MAXARCS),C(MAXARCS),C2(MAXARCS),DUM3
      REAL*8 U(MAXARCS)
      REAL*8 MAXCOST
      REAL*8 COST(MAXARCS)
      REAL*8 F(MAXARCS)
      REAL*8 SURPLUS(MAXNODES)
      REAL*8 P(MAXNODES)
      REAL*8 FACTOR,EPS,THRESHSURPLUS,SFACTOR,MAXSURPLUS,LARGE
      real*8 ref_thr

      REAL*8 TT,TIMER
      DOUBLE PRECISION TCOST,PRODUCT

      COMMON /SCALARS/  N,NA,LARGE
      COMMON /SC2/ FACTOR,EPS,THRESHSURPLUS,SFACTOR,MAXSURPLUS
      COMMON /STATS/    NCYC,NITER,TOTALITER
      COMMON /BLK1/     STARTN
      COMMON /BLK2/     ENDN
      COMMON /UBOUND/   U
      COMMON /FLOW/     F
      COMMON /PRICES/   P
      COMMON /BLK3/     SURPLUS
      COMMON /BLK4/     FIN
      COMMON /BLK5/     FOUT
      COMMON /BLK6/     NXTIN
      COMMON /BLK7/     NXTOU
      COMMON /BLK11/    FPUSHF
      COMMON /BLK12/    NXTPUSHF
      COMMON /BLK13/    FPUSHB
      COMMON /BLK14/    NXTPUSHB
      COMMON /COST/     COST
      COMMON /QUEUE/    NXTQUEUE
      COMMON /QCO1/     C
      COMMON /QCO2/     C2
      COMMON /LCOS/     A
     COMMON /QBU1/    QFPUSHF
      COMMON /QBU2/    QNXTPUSHF
      COMMON /QBU3/    QFPUSHB
      COMMON /QBU4/    QNXTPUSHB
      real*8 SMAL
      REAL*8 MAXVIOL
      real etime, tarray1(2)
      external etime

      LARGE=2000000000
      SMAL = 0.0
C     Print*,'DO you want to have ill conditioning?(1)'
C     read(*,*),dummy
C     if (dummy .eq.1) then
C        print*, 'Will substitute zero quad. coeff with
C    $   the small coefficient you will enter now'
C        READ(*,*) SMAL
C        PRINT*, SMAL
C     end if
      PRINT*,'E-RELAXATION METHOD FOR MIN COST FLOW'
      PRINT*,'************************************'
      PRINT *,'READING PROBLEM DATA'
CC     OPEN(13,FILE='QUAD013.DAT',STATUS='OLD')
      OPEN(13,FILE='NL013.DAT',STATUS='OLD')
```

```
      REWIND(13)

C     READ NUMBER OF NODES AND ARCS

CC      READ(13,1010) N,NA
      READ(13,1010) N,NA,DUMMY
      PRINT*,' #NODES=',N,' #ARCS=',NA

C     READ START, END, COST, AND CAPACITY OF EACH ARC
C     AND GENERATE MAX COST VALUE

      MAXCOST=0.0

      DO 5 I=1,NA
CC        READ(13,1020) STARTN(I),ENDN(I),DUM1,DUM2,U(I)
CC        A(I) = DBLE(DUM1)
        READ(13,1020) STARTN(I),ENDN(I),A(I),C(I),U(I),DUM3
 COST(I) = A(I)
        if ((dum2 .eq.0).and.(smal.gt.1.0e-9)) then
          C(I) = SMAL
          C2(I) = 2.0*C(I)
c           print*,'executed it'
        ELSE
CC           C(I) = DBLE(DUM2)
          C2(I) = 2.0*C(I)
        END IF
        MAXCOST=MAX( MAXCOST, ABS(COST(I)+C2(I)*U(I)) )
5     CONTINUE

C     READ SUPPLY OF EACH NODE

      DO 8 I=1,N
CC        READ(13,1000) DUMMY
CC        SURPLUS(I) = DBLE(DUMMY)
        READ(13,1000) SURPLUS(I)
8     CONTINUE

      REWIND(13)
      CLOSE(13)

      PRINT*,'END OF READING'

CC1000  FORMAT(1I8)
CC1010  FORMAT(2I8)
CC1020  FORMAT(5I8)
1000  FORMAT(E15.9)
1010  FORMAT(2I8,I3)
1020  FORMAT(2I8,8E15.9)

C     INPUT NUMBER OF DIGITS OF ACCURACY DESIRED IN SOLUTION
C
      PRINT*,' INPUT #DIGITS OF ACCURACY FOR SOLUTION'
      READ(*,*)ACCUR


C     INITIALIZE PRICES & PUSH LISTS

      DO 10 NODE=1,N
 FPUSHF(NODE)=0
 FPUSHB(NODE)=0
```

```
QFPUSHF(NODE)=0
QFPUSHB(NODE)=0
C        P(NODE)=-(LARGE/10)
         P(NODE)=0.0
10    CONTINUE

C     IN THE FOLLOWING STATEMENTS, THE E-SCALING PARAMETERS ARE SET.
C     THE FOLLOWING RANGES ARE RECOMMENDED:
C     STARTING EPSILON: BETWEEN MAXCOST/20 TO MAXCOST/2
C     SCALING FACTOR: BETWEEN FOUR AND TEN
c      WRITE(*,*)'ENTER STARTING EPSILON'
c      WRITE(*,*)'SHOULD BE BETWEEN 1 & ',MAXCOST
c      READ*,EPS
c      IF (EPS.LT.1) GO TO 25
      EPS = MAXCOST/5.0
      IF (EPS.LT.(1.0/(N+1.0))) EPS = 1.0/(N+1.0)
      FACTOR = 5.0
      WRITE(*,*) 'EPS, FACTOR:',EPS,FACTOR
c      WRITE(*,*)'ENTER SCALING FACTOR'
c30    CONTINUE
c      READ*,FACTOR
c      IF ((EPS.GT.1).AND.(FACTOR.LE.1)) THEN
c        PRINT*,'ENTER SCALING FACTOR; SHOULD BE GREATER THAN 1'
c        GO TO 30
c      END IF
C
MAXSURPLUS=1.0
      DO 920 NODE=1,N
IF (SURPLUS(NODE).GT.MAXSURPLUS) THEN
  MAXSURPLUS=SURPLUS(NODE)
END IF
920   CONTINUE

C     SET THE PARAMETER SSCALE TO 1 TO ALLOW SURPLUS SCALING
C     A HEURISTIC SCHEME IS USED TO SET THE SFACTOR AND
C     THRESHSURPLUS PARAMETERS THAT CONTROL SURPLUS SCALING

      REF_THR = 1.0E-10
      SSCALE=1

      IF (SSCALE.EQ.1) THEN
SFACTOR=2.0+INT(FACTOR*MAXSURPLUS/MAXCOST)
IF (SFACTOR.LT.4.0) SFACTOR=4.0
THRESHSURPLUS=MAX(INT(MAXSURPLUS/SFACTOR),REF_THR)
CIF (EPS.LE.(1.0/(N+1.0))) THRESHSURPLUS=REF_THR
      ELSE
        THRESHSURPLUS=REF_THR
      END IF

      PRINT*,'INITIALIZING DATA STRUCTURES'
      CALL INIDAT

      PRINT *,'**********************************'
      PRINT *,'CALLING E-RELAX FOR MCF (+ SURPLUS NODES ITERATED ONLY)'
      TIMER = etime(tarray1)
      TIMER = TARRAY1(1)

      CALL EPS_RELAX(ACCUR)

      TT = etime(tarray1)
```

```
      TT = TARRAY1(1)-TIMER
      PRINT*, 'TOTAL TIME = ',TT, ' secs.'
      open(17,FILE='FORT9',STATUS='unknown')
      rewind(17)


      TCOST=0
      DO 330 I=1,NA
C     COST(I)=COST(I)
C     PRODUCT=F(I)*F(I)*(C(I))+F(I)*(A(I))
        TCOST=TCOST+PRODUCT
330   CONTINUE

c     WRITE(9,1100) TCOST
c1100  FORMAT(' ','OPTIMAL COST        =',F14.2)

      PRINT *,'**********************************'
      WRITE(*,*) 'OPTIMAL COST    = ', TCOST
      PRINT *,'# OF ITERATIONS = ',TOTALITER
      PRINT *,'# OF S. N. PRICE RISES = ',NITER
      PRINT *,'**********************************'

C     CHECK CORRECTNESS OF THE ANSWER

      MAXVIOL = 0.0
      DO 80 NODE=1,N
        MAXVIOL = MAX(MAXVIOL,ABS(SURPLUS(NODE)))
80    CONTINUE
C     PRINT*,'FLOW CONSERVATION VIOLATION= ',MAXVIOL
      NUM_POS_VI = 0
      NUM_NEG_VI = 0
      MAXVIOL = 0.0
      DO 90 ARC=1,NA
        COST(ARC)=COST(ARC)
        IF (F(ARC).GT.REF_THR) THEN
C     IF (P(STARTN(ARC))-P(ENDN(ARC)).LT.-EPS+COST(ARC)
     $         -REF_THR) THEN
C     PRINT*,'E-CS VIOLATED AT ARC ',ARC,
C     $   P(STARTN(ARC))-P(ENDN(ARC))+EPS-COST(ARC)
          NUM_POS_VI =  NUM_POS_VI + 1
          MAXVIOL = MAX(MAXVIOL,
     $     ABS(P(STARTN(ARC))-P(ENDN(ARC))+EPS-COST(ARC)))
C     ENDIF
C     ENDIF
        IF (F(ARC).LT.(U(ARC)-REF_THR)) THEN
C     IF (P(STARTN(ARC))-P(ENDN(ARC)).GT.EPS+COST(ARC)
     $       +REF_THR) THEN
c     PRINT*,'E-CS VIOLATED AT ARC ',ARC,
c     $     P(STARTN(ARC))-P(ENDN(ARC))-EPS-COST(ARC)
          NUM_NEG_VI =  NUM_NEG_VI + 1
          MAXVIOL = MAX(MAXVIOL,
     $     ABS(P(STARTN(ARC))-P(ENDN(ARC))-EPS-COST(ARC)))
C     ENDIF
C     ENDIF
90    CONTINUE
      print*, 'NUMBER OF E-CS VIOLATIONS: ',NUM_POS_VI,' ',NUM_NEG_VI
      PRINT*, 'MAXIMUM VIOLATION: ', MAXVIOL
C      PRINT *, 'PROGRAM ENDED; PRESS <CR>'
C      PAUSE
      STOP
```

```
      END


C     ************************************************
      SUBROUTINE INIDAT
C     THIS SUBROUTINE USES THE DATA ARRAYS STARTN AND ENDN
C     TO CONSTRUCT AUXILIARY DATA ARRAYS FOUT, NXTOU, FIN, AND
C     NXTIN THAT ARE REQUIRED BY E-RELAX.  IN THIS SUBROUTINE WE
C     ARBITRARILY ORDER THE ARCS LEAVING EACH NODE AND STORE
C     THIS INFORMATION IN FOUT AND NXTOU.  SIMILARLY, WE ARBITRA-
C     RILLY ORDER THE ARCS ENTERING EACH NODE AND STORE THIS
C     INFORMATION IN FIN AND NXTIN.  AT THE COMPLETION OF THE
C     CONSTRUCTION, WE HAVE THAT
C
C        FOUT(I)    = FIRST ARC LEAVING NODE I.
C        NXTOU(J)   = NEXT ARC LEAVING THE HEAD NODE OF ARC J.
C        FIN(I)     = FIRST ARC ENTERING NODE I.
C        NXTIN(J)   = NEXT ARC ENTERING THE TAIL NODE OF ARC J.

      PARAMETER (MAXNODES=5000, MAXARCS=20000)
      IMPLICIT INTEGER (A-Z)

      INTEGER STARTN(MAXARCS)
      INTEGER ENDN(MAXARCS)
      INTEGER FIN(MAXNODES)
      INTEGER FOUT(MAXNODES)
      INTEGER NXTIN(MAXARCS)
      INTEGER NXTOU(MAXARCS)
      INTEGER FINALIN(MAXNODES),FINALOU(MAXNODES)

      REAL*8 FACTOR,EPS,THRESHSURPLUS,SFACTOR,MAXSURPLUS,LARGE


      COMMON /SCALARS/  N,NA,LARGE
      COMMON /SC2/ FACTOR,EPS,THRESHSURPLUS,SFACTOR,MAXSURPLUS
      COMMON /BLK1/     STARTN
      COMMON /BLK2/     ENDN
      COMMON /BLK4/     FIN
      COMMON /BLK5/     FOUT
      COMMON /BLK6/     NXTIN
      COMMON /BLK7/     NXTOU

      DO 20 NODE=1,N
        FIN(NODE)=0
        FOUT(NODE)=0
        FINALIN(NODE)=0
        FINALOU(NODE)=0
20    CONTINUE

      DO 30 ARC=1,NA
        START=STARTN(ARC)
        END=ENDN(ARC)
        IF (FOUT(START).NE.0) THEN
          NXTOU(FINALOU(START))=ARC
        ELSE
          FOUT(START)=ARC
        END IF
        IF (FIN(END).NE.0) THEN
          NXTIN(FINALIN(END))=ARC
```

```
      ELSE
        FIN(END)=ARC
      END IF
      FINALOU(START)=ARC
      FINALIN(END)=ARC
      NXTIN(ARC)=0
      NXTOU(ARC)=0
30    CONTINUE

      RETURN
      END



C  ***************************************************************
C
C     SCALED E-RELAXATION
C     THIS VERSION USES A QUEUE TO SELECT NODES.
C     NODES JOIN THE QUEUE AT THE BOTTOM.
C
C  ***************************************************************

      SUBROUTINE EPS_RELAX(ACCUR)
      IMPLICIT NONE

      INTEGER ACCUR
      INTEGER N,NA,MAXNODES,MAXARCS,ARC,ARCF,ARCB
      INTEGER NCYC,NITER,NODE
      PARAMETER (MAXNODES=5000, MAXARCS=20000)
      INTEGER I,J,K,LN,PREVNODE,LASTQUEUE
      INTEGER NEXT,START,END
      INTEGER NXTNODE,TOTALITER
      INTEGER NXTQUEUE(MAXNODES)
      INTEGER STARTN(MAXARCS),ENDN(MAXARCS),FIN(MAXNODES),
     $FOUT(MAXNODES)
      INTEGER NXTIN(MAXARCS),NXTOU(MAXARCS)
      INTEGER FPUSHF(MAXNODES),NXTPUSHF(MAXARCS),FPUSHB(MAXNODES),
     $NXTPUSHB(MAXARCS)
      INTEGER QFPUSHF(MAXNODES),QNXTPUSHF(MAXARCS)
      INTEGER QFPUSHB(MAXNODES),QNXTPUSHB(MAXARCS)
      INTEGER QARCF,QARCB

      REAL*8 A(MAXARCS),C(MAXARCS),C2(MAXARCS),U(MAXARCS)
      REAL*8 CAPOUT,CAPIN
      REAL*8 COST(MAXARCS),P(MAXNODES),F(MAXARCS),SURPLUS(MAXNODES)
      REAL*8 THRESHSURPLUS,SFACTOR,MAXSURPLUS,FACTOR,EPS,LARGE
      REAL*8 REFPRICE,PRJ,XP,REFPRJ,PRICEJ,PRICE,PRICEINCR
      REAL*8 PSTART,PEND,DEL,MAXPRICE
      REAL*8 SURPI,FLOW, RESID,UP
      REAL*8 REF_THR, EPS2

      COMMON /SCALARS/  N,NA,LARGE
      COMMON /SC2/ FACTOR,EPS,THRESHSURPLUS,SFACTOR,MAXSURPLUS
      COMMON /STATS/    NCYC,NITER,TOTALITER
      COMMON /BLK1/     STARTN
      COMMON /BLK2/     ENDN
      COMMON /UBOUND/   U
      COMMON /FLOW/     F
      COMMON /PRICES/   P
      COMMON /BLK3/     SURPLUS
```

```
      COMMON /BLK4/     FIN
      COMMON /BLK5/     FOUT
      COMMON /BLK6/     NXTIN
      COMMON /BLK7/     NXTOU
      COMMON /BLK11/    FPUSHF
      COMMON /BLK12/    NXTPUSHF
      COMMON /BLK13/    FPUSHB
      COMMON /BLK14/    NXTPUSHB
      COMMON /COST/     COST
      COMMON /QUEUE/    NXTQUEUE
      COMMON /QCO1/     C
      COMMON /QCO2/     C2

      COMMON /LCOS/     A
      COMMON /QBU1/     QFPUSHF
      COMMON /QBU2/     QNXTPUSHF
      COMMON /QBU3/     QFPUSHB
      COMMON /QBU4/     QNXTPUSHB

C     ADDITIONAL DATA STRUCTURE USED TO COMPUTE DUAL COST

      REAL*8 PCOST,DCOST,RC,F1,DFCT(MAXNODES),DFCT1(MAXNODES)

      DO 1 I=1,N
       DFCT1(I)=-SURPLUS(I)
1     CONTINUE

C     INITIALIZE COUNT OF NUMBER OF UP ITERATIONS PERFORMED

      NITER = 0
      TOTALITER=0
      NCYC=0
      MAXPRICE =LARGE
      REF_THR = 1.0D-10

C     REDUCE ARC CAPACITIES

        DO 40 NODE=1,N
          CAPOUT=0
          ARC=FOUT(NODE)
41          IF (ARC.GT.0) THEN
             CAPOUT=MIN(LARGE,CAPOUT+U(ARC))
             ARC=NXTOU(ARC)
             GO TO 41
          END IF
          CAPOUT=MIN(LARGE,CAPOUT-SURPLUS(NODE))
          IF (CAPOUT.LT.0) GOTO 400
          CAPIN=0
          ARC=FIN(NODE)
43          IF (ARC.GT.0) THEN
             IF (U(ARC) .GT. CAPOUT) THEN
                U(ARC)=CAPOUT
             ENDIF
             CAPIN=MIN(LARGE,CAPIN+U(ARC))
             ARC=NXTIN(ARC)
             GO TO 43
          END IF
          CAPIN=MIN(LARGE,CAPIN+SURPLUS(NODE))
          IF (CAPIN.LT.0) GOTO 400
          ARC=FOUT(NODE)
```

```
45          IF (ARC.GT.0) THEN
             IF (U(ARC) .GT. CAPIN) THEN
               U(ARC)=CAPIN
             ENDIF
             ARC=NXTOU(ARC)
             GO TO 45
           END IF
40        CONTINUE

C   SET ARC FLOWS TO SATISFY E-CS

        EPS2 = EPS/2.0

        DO 49 ARC=1,NA
         START=STARTN(ARC)
  END=ENDN(ARC)
  PSTART=P(START)
  PEND=P(END)
  IF (PSTART.GE.PEND+COST(ARC)+EPS2) THEN
           IF (C(ARC).GT.REF_THR) THEN
            UP = MIN((PSTART-PEND-A(ARC))/C2(ARC),U(ARC))
            COST(ARC) = C2(ARC)*UP+A(ARC)
           ELSE
             UP = U(ARC)
           END IF
     SURPLUS(START)=SURPLUS(START)-UP
     SURPLUS(END)=SURPLUS(END)+UP
     F(ARC)=UP
  ELSE
     F(ARC)=0.0
  END IF
49      CONTINUE

        PRINT*, 'STARTING NEW SCALING PHASE'
C       ********** START OF A NEW SCALING PHASE **********

60        CONTINUE
         EPS2 = EPS/2.0
C       ***** QUEUE INITIALIZATION *****

        DO 82 NODE=1,N-1
           NXTQUEUE(NODE)=NODE+1
82       CONTINUE
        NXTQUEUE(N)=1
        I=1
        PREVNODE=N
        LASTQUEUE=N

C       ********** START A NEW CYCLE OF UP ITERATIONS **********

100     CONTINUE

C       TAKE UP NEXT NODE (NODE I) FOR ITERATION

 SURPI=SURPLUS(I)
        IF (SURPI .GT. THRESHSURPLUS)  THEN

C     ARCF & ARCB ARE THE CURRENT VALUES OF THE STARTING ARCS OF THE
C     PUSH LISTS OF I
```

```
          TOTALITER=TOTALITER+1
          ARCF = FPUSHF(I)
          ARCB = FPUSHB(I)
          QARCF = QFPUSHF(I)
          QARCB = QFPUSHB(I)

        PRICE = P(I)
C         PRINT*,'NODE ',I,'   START PRICE = ',PRICE

115       IF ((ARCF .GT. 0) .OR. (ARCB .GT. 0)) THEN

C         START BY TRYING TO PUSH AWAY FLOW ON ARCS THAT WERE
C         ADMISSIBLE AT THE END OF THE LAST ITERATION (IF ANY)
C         AT THIS NODE.  WE MUST CHECK THAT THEY ARE STILL
C         ADMISSIBLE.


120           IF ((SURPI .GT. REF_THR) .AND. (ARCF .GT. 0)) THEN
          J = ENDN(ARCF)
          IF (PRICE-P(J)-COST(ARCF).GT.EPS2) THEN
C******************TAKING CARE OF LINEAR FORWARD ARCS*****************************
            RESID = U(ARCF) - F(ARCF)
                IF (RESID.GT.REF_THR) THEN
            IF (SURPI .GE. RESID) THEN
          F(ARCF) = U(ARCF)
          SURPI = SURPI - RESID
          SURPLUS(J) = SURPLUS(J) + RESID
          ARCF = NXTPUSHF(ARCF)
            ELSE
          F(ARCF) = F(ARCF) + SURPI
          SURPLUS(J) = SURPLUS(J) + SURPI
          SURPI = 0.0
            ENDIF
            IF (SURPLUS(J).GT.THRESHSURPLUS) THEN
             IF (NXTQUEUE(J).EQ.0) THEN
           NXTQUEUE(PREVNODE)=J
           NXTQUEUE(J)=I
           PREVNODE=J
             END IF
            END IF
                ELSE
            ARCF = NXTPUSHF(ARCF)
          ENDIF
          ELSE
            ARCF = NXTPUSHF(ARCF)
          ENDIF

            GOTO 120
          ENDIF

121           IF ((SURPI .GT. REF_THR) .AND. (ARCB .GT. 0)) THEN
         J = STARTN(ARCB)
         IF (PRICE-P(J)+COST(ARCB).GT.EPS2) THEN
C ************  TAKING CARE OF THE LINEAR COST REVERSE ARCS  ***************
            RESID=F(ARCB)
            IF (RESID.GT.REF_THR) THEN
          IF (SURPI .GE. RESID) THEN
            SURPI = SURPI - RESID
            SURPLUS(J) = SURPLUS(J) + RESID
            F(ARCB) = 0
```

```
      ARCB = NXTPUSHB(ARCB)
    ELSE
      F(ARCB) = F(ARCB) - SURPI
      SURPLUS(J) = SURPLUS(J) + SURPI
      SURPI = 0.0
    ENDIF
    IF (SURPLUS(J).GT.THRESHSURPLUS) THEN
     IF (NXTQUEUE(J).EQ.0) THEN
      NXTQUEUE(PREVNODE)=J
      NXTQUEUE(J)=I
      PREVNODE=J
     END IF
    END IF
              ELSE
    ARCB = NXTPUSHB(ARCB)
         ENDIF
      ELSE
       ARCB = NXTPUSHB(ARCB)
      ENDIF

       GOTO 121
     ENDIF
 ENDIF

 IF ((QARCF+QARCB).GT.0) THEN
122       IF ((SURPI .GT. REF_THR) .AND. (QARCF .GT. 0)) THEN
J = ENDN(QARCF)
IF (PRICE-P(J)-COST(QARCF).GT.EPS2) THEN
C**************TAKING CARE OF NONLINEAR FORWARD ARCS*********************************
  RESID=MIN(U(QARCF),(PRICE-P(J)-A(QARCF))/C2(QARCF))
     $              - F(QARCF)
           IF (RESID.GT.REF_THR) THEN
    IF (SURPI .GE. RESID) THEN
F(QARCF) = F(QARCF)+RESID
SURPI = SURPI - RESID
SURPLUS(J) = SURPLUS(J) + RESID
COST(QARCF) = C2(QARCF)*F(QARCF) + A(QARCF)
QARCF = QNXTPUSHF(QARCF)
    ELSE
F(QARCF) = F(QARCF) + SURPI
COST(QARCF) = C2(QARCF)*F(QARCF) + A(QARCF)
SURPLUS(J) = SURPLUS(J) + SURPI
SURPI = 0.0
    ENDIF
    IF (SURPLUS(J).GT.THRESHSURPLUS) THEN
      IF (NXTQUEUE(J).EQ.0) THEN
   NXTQUEUE(PREVNODE)=J
   NXTQUEUE(J)=I
   PREVNODE=J
      END IF
    END IF
           ELSE
   QARCF = QNXTPUSHF(QARCF)
  ENDIF
     ELSE
      QARCF = QNXTPUSHF(QARCF)
     ENDIF

      GOTO 122
    ENDIF
```

```
123        IF ((SURPI .GT. REF_THR) .AND. (QARCB .GT. 0)) THEN
      J = STARTN(QARCB)
      IF (PRICE-P(J)+COST(QARCB).GT.EPS2) THEN
C ************* TAKING CARE OF ARCS WITH NON LINEAR COST ******************
       RESID=F(QARCB)-MAX(0.0,(P(J)-PRICE-A(QARCB))/
     $           C2(QARCB))
              IF (RESID.GT.REF_THR) THEN
       IF (SURPI .GE. RESID) THEN
  F(QARCB) = F(QARCB)-RESID
  SURPI = SURPI - RESID
  SURPLUS(J) = SURPLUS(J) + RESID
  COST(QARCB) = C2(QARCB)*F(QARCB)+ A(QARCB)
  QARCB = QNXTPUSHB(QARCB)
       ELSE
  F(QARCB) = F(QARCB) - SURPI
  COST(QARCB) = C2(QARCB)*F(QARCB)+ A(QARCB)
  SURPLUS(J) = SURPLUS(J) + SURPI
  SURPI = 0.0
       ENDIF
       IF (SURPLUS(J).GT.THRESHSURPLUS) THEN
         IF (NXTQUEUE(J).EQ.0) THEN
     NXTQUEUE(PREVNODE)=J
     NXTQUEUE(J)=I
     PREVNODE=J
         END IF
       END IF
              ELSE
       QARCB = QNXTPUSHB(QARCB)
  ENDIF
      ELSE
       QARCB = QNXTPUSHB(QARCB)
      ENDIF

       GOTO 123
     ENDIF
   END IF

C   ***** END OF D-PUSHES; CHECK IF PRICE RISE IS NEEDED *****

       IF ((SURPI.GT.THRESHSURPLUS).AND.
     $  (ARCF+ARCB+QARCF+QARCB.EQ.0)) THEN

C    ******************DO A PRICE RISE *********************

REFPRICE=PRICE
PRICE = LARGE
NITER=NITER+1
ARC = FOUT(I)
111             IF (ARC .GT. 0) THEN
                UP = U(ARC)
                IF ((F(ARC)-UP).LT.(-REF_THR)) THEN
     XP = P(ENDN(ARC))+COST(ARC)
     IF ((XP-PRICE).LT.-EPS) THEN
PRICE = XP + EPS
IF (C(ARC).GT.REF_THR) THEN
C************ QUADRATIC FORWARD PUSH LIST***********************
   QARCF = ARC
   QNXTPUSHF(ARC)=0
   ARCF = 0
```

```
□□□ELSE
C********** LINEAR FORWARD PUSH LIST**********************
□□□  ARCF = ARC
□□□  NXTPUSHF(ARC)=0
□□□  QARCF = 0
□□□END IF
□□    ELSE
□□     IF ((XP-PRICE).GE.-EPS2) THEN
□□□ IF (C(ARC).GT.REF_THR) THEN
C********** QUADRATIC FORWARD PUSH LIST**********************
□□□  IF (QARCF.EQ.0) THEN
□□□   QARCF = ARC
□□□   QNXTPUSHF(ARC)=0
□□□  ELSE
□□□   QNXTPUSHF(ARC) = QARCF
□□□   QARCF = ARC
□□□  END IF
□□□ ELSE
C********** LINEAR FORWARD PUSH LIST**********************
               IF (ARCF.EQ.0) THEN
□□□   ARCF = ARC
□□□   NXTPUSHF(ARC)=0
□□□  ELSE
□□□   NXTPUSHF(ARC) = ARCF
□□□   ARCF = ARC
□□□  END IF
□□□ END IF
□□     END IF
□□    ENDIF
□□  ENDIF
□□  ARC = NXTOU(ARC)
□□  GOTO 111
□      ENDIF


□      ARC = FIN(I)
112          IF (ARC .GT. 0) THEN
□□ IF (F(ARC) .GT. REF_THR) THEN
□□   XP = P(STARTN(ARC))-COST(ARC)
□□   IF ((XP-PRICE).LT. -EPS) THEN
□□      PRICE = XP + EPS
□□      ARCF=0
□□□ QARCF = 0
□□□IF (C(ARC).GT.REF_THR) THEN
C********** QUADRATIC REVERSE PUSH LIST**********************
□□□  QARCB = ARC
□□□  QNXTPUSHB(ARC)=0
□□□  ARCB = 0
□□□ELSE
C********** LINEAR REVERSE PUSH LIST**********************
□□□  ARCB = ARC
□□□  NXTPUSHB(ARC)=0
□□□  QARCB = 0
□□□END IF
□□   ELSE
□□    IF ((XP-PRICE).LT. -EPS2) THEN
□□□ IF (C(ARC).GT.REF_THR) THEN
C********** QUADRATIC FORWARD PUSH LIST**********************
□□□  IF (QARCB.EQ.0) THEN
□□□   QARCB = ARC
```

```
□□□   QNXTPUSHB(ARC)=0
□□□  ELSE
□□□   QNXTPUSHB(ARC) = QARCB
□□□   QARCB = ARC
□□□  END IF
□□□ ELSE
C********** LINEAR FORWARD PUSH LIST**********************
               IF (ARCB.EQ.0) THEN
□□□   ARCB = ARC
□□□   NXTPUSHB(ARC)=0
□□□  ELSE
□□□   NXTPUSHB(ARC) = ARCB
□□□   ARCB = ARC
□□□  END IF
□□□ END IF
□□    END IF
□□   ENDIF
□□ ENDIF
□□ ARC = NXTIN(ARC)
□□ GOTO 112
□     ENDIF

C    IF PRICE=LARGE, THE PUSH LIST IS LIKELY EMPTY, SO SET PRICE TO AT LEAST THE
C    LEVEL OF THE BEST ARC

        IF (PRICE.GE.LARGE) THEN
□     ARCF=0
□□ARCB=0
□     QARCF=0
□□QARCB=0
□□
□□ARC = FOUT(I)
211           IF (ARC.GT.0) THEN
□□ XP = P(ENDN(ARC))+COST(ARC)
□□ IF ((XP-PRICE).LT. (-REF_THR)) THEN
□□   PRICE = XP
□□ ELSE
□□   IF (XP.GE.LARGE) THEN
□□    PRINT*,'PRICE OF ',I,'HAS EXCEEDED THE INT. RANGE'
C□□    PAUSE
□□    STOP
□□   END IF
□□ ENDIF
□□ ARC = NXTOU(ARC)
□□ GOTO 211
□      ENDIF
□      ARC = FIN(I)
□
212          IF (ARC.GT.0) THEN
□□ XP = P(STARTN(ARC))-COST(ARC)
□□ IF ((XP-PRICE).LT.(-REF_THR)) THEN
□□    PRICE = XP
□□ ELSE
□□   IF (XP.GE.LARGE) THEN
□□    PRINT*,'PRICE OF ',I,'HAS EXCEEDED THE INT. RANGE'
C□□    PAUSE
□□    STOP
□□   END IF
□□ ENDIF
□□ ARC = NXTIN(ARC)
```

```
      GOTO 212
      ENDIF

      IF (SURPI.GT.REF_THR) GOTO 400
      IF (PRICE.LT.IDINT(LARGE/10)) PRICE=IDINT(LARGE/10)

   END IF


C     SET PRICE OF NODE I


   P(I) = PRICE
   FPUSHF(I) = ARCF
   FPUSHB(I) = ARCB
   QFPUSHF(I) = QARCF
   QFPUSHB(I) = QARCB
C    PRICEINCR=PRICE-REFPRICE
C    IF (PRICEINCR.LE.0) THEN
C      PRINT*,'NONPOSITIVE PRICE INCREMENT=',PRICEINCR
C      PAUSE
C      STOP
C    ENDIF

C           END OF PRICE RISE; IF THERE IS STILL A LOT OF SURPLUS,
C           TRY AND DO SOME PUSHING.

          IF (SURPI.GT.THRESHSURPLUS) GOTO 115

 ELSE

C          IF NO PRICE RISE TOOK PLACE RESET THE START OF THE PUSH LISTS

   FPUSHF(I) = ARCF
   FPUSHB(I) = ARCB
   QFPUSHF(I) = QARCF
   QFPUSHB(I) = QARCB
        ENDIF

C     DO THE FINAL BOOKKEEPING OF THE ITERATION


       SURPLUS(I) = SURPI

C     IF THE PRICE IS TOO HIGH, THEN SOMETHING IS WRONG

C IF (PRICE.GT.MAXPRICE) THEN
C   GO TO 400
C ENDIF

      ENDIF

C      ********** END OF UP ITERATION STARTING AT NODE I *********

C      CHECK FOR THE END OF THE QUEUE. THIS STEP IS NOT NECESSARY FOR
C      THE ALGORITHM; IT IS INCLUDED FOR COLLECTING STATISTICS ABOUT
C      THE ALGORITHM'S BEHAVIOR, E.G. COUNTING THE NUMBER OF CYCLES

C       IF (I.EQ.LASTQUEUE) THEN
```

```
C   LASTQUEUE=PREVNODE
C   NCYC=NCYC+1
C         PRINT*,'CYCLE # ',NCYC,'   # OF PRICE RISES ',NITER
C        END IF


C      CHECK FOR TERMINATION OF SCALING PHASE. IF SCALING PHASE IS
C      NOT FINISHED, ADVANCE THE QUEUE AND RETURN TO TAKE ANOTHER NODE.

       NXTNODE=NXTQUEUE(I)
 IF (I.NE.NXTNODE) THEN
   NXTQUEUE(I)=0
   NXTQUEUE(PREVNODE)=NXTNODE
   I=NXTNODE
   GO TO 100
 END IF

C ************** END OF SUBPROBLEM (SCALING PHASE) **************

      PRINT*,'END OF SCALING PHASE'

C      DO A DIAGNOSTIC CHECK

C      LN=0
C      DO 500 NODE=1,N
C        IF (SURPLUS(NODE).NE.0) THEN
C         LN=LN+1
C ENDIF
500     CONTINUE
C      PRINT*,'NONZERO SURPLUS AT ',LN,' NODES '
C     DO 600 ARC=1,NA
C        IF (F(ARC).GT.0) THEN
C  IF (P(STARTN(ARC))-P(ENDN(ARC)).LT.-EPS+COST(ARC)) THEN
C    PRINT*,'E-CS VIOLATED AT ARC ',ARC
C  ENDIF
C ENDIF
C        IF (F(ARC).LT.U(ARC)) THEN
C  IF (P(STARTN(ARC))-P(ENDN(ARC)).GT.EPS+COST(ARC)) THEN
C    PRINT*,'E-CS VIOLATED AT ARC ',ARC
C  ENDIF
C ENDIF
600     CONTINUE
C       PRINT*,'END OF CHECK OF OLD PHASE'

C ****** IF DUALITY GAP IS WITHIN .1**ACCUR, THEN STOP;
C ELSE REDUCE EPSILON******
C
       PCOST=0.
       DO 330 K=1,NA
330      PCOST=PCOST+C2(K)*F(K)*F(K)/2+F(K)*A(K)
       PRINT*,'PRIMAL COST =',PCOST
       DCOST=0.
       DO 341 I=1,N
341      DFCT(I)=DFCT1(I)
       DO 331 K=1,NA
         RC=A(K)-P(STARTN(K))+P(ENDN(K))
         IF (C2(K).LT.REF_THR) THEN
           F1=0
           IF (RC.LT.0) F1=U(K)
         ELSE
```

```
          F1=DMIN1(-RC/C2(K),U(K))
          IF (F1.LT.0) F1=0
        END IF
        DFCT(STARTN(K))=DFCT(STARTN(K))+F1
        DFCT(ENDN(K))=DFCT(ENDN(K))-F1
        DCOST=DCOST+C2(K)*F1*F1/2+A(K)*F1
331     CONTINUE
      DO 335 I=1,N
335     DCOST=DCOST-P(I)*DFCT(I)
      PRINT*,'DUAL COST =',DCOST
C
CC       IF (PCOST-DCOST.LE.DABS(PCOST)*(.1**ACCUR)) THEN
CC         IF (THRESHSURPLUS.LE.REF_THR) THEN
CC           RETURN
CC         ELSE
CC           THRESHSURPLUS=REF_THR
CC         END IF
CC  ELSE
CC    EPS=(EPS/FACTOR)
CC         THRESHSURPLUS=(THRESHSURPLUS/SFACTOR)
CC  END IF

      IF ( (DABS(PCOST-DCOST).LE.DABS(PCOST)*(.1**ACCUR))
     $  .AND. (THRESHSURPLUS.LE.MAXSURPLUS*(.1**ACCUR)) ) THEN
        RETURN
  ELSE
    EPS=(EPS/FACTOR)
        THRESHSURPLUS=(THRESHSURPLUS/SFACTOR)
    THRESHSURPLUS=MAX(THRESHSURPLUS,EPS*N/(NA+1.0))
        IF ((THRESHSURPLUS.LE.REF_THR).OR.(EPS.LE.REF_THR)) THEN
          PRINT*,'EXIT: THRESHSURPLUS=',THRESHSURPLUS,' EPS=',EPS
          RETURN
        END IF
  END IF

C       IF (EPS.LT. 1.0/(n+1.0)) THRESHSURPLUS=REF_THR

C     RESET THE FLOWS & THE PUSH LISTS; FIND THE MINIMAL PRICE

      XP=LARGE
      DO 800 NODE=1,N
  FPUSHF(NODE)=0
  FPUSHB(NODE)=0
  QFPUSHF(NODE)=0
  QFPUSHB(NODE)=0
  IF (P(NODE).LT.XP) XP=P(NODE)
800    CONTINUE

C     REDUCE ALL PRICES TO REDUCE DANGER OF OVERFLOW

C       DEL=XP+(LARGE/10)
C       IF (DEL.LT.0.0) DEL=0.0
C       DO 810 NODE=1,N
C         P(NODE)=P(NODE)-DEL
C810    CONTINUE

C     PRINT*,'MODIFYING ARC FLOWS TO SATISFY E-CS'

      DO 900 ARC=1,NA
        START=STARTN(ARC)
```

```
      END=ENDN(ARC)
      PSTART=P(START)
      PEND=P(END)
      IF ((PSTART-PEND-EPS2-COST(ARC)).GE.0.0) THEN
        IF (C(ARC).GT.REF_THR) THEN
          UP = MIN(((pstart-pend-a(arc))/
     $           C2(ARC)),U(ARC))
          COST(ARC) = C2(ARC)*UP+A(ARC)
        ELSE
          UP = U(ARC)
        END IF
            RESID=UP-F(ARC)
C       IF (RESID.GT.REF_THR) THEN
        SURPLUS(START)=SURPLUS(START)-RESID
        SURPLUS(END)=SURPLUS(END)+RESID
        F(ARC)=UP
C       END IF
      ELSE
       IF ((-PSTART+PEND-EPS2+COST(ARC)).GE.0.0) THEN
        IF (C(ARC).GT.REF_THR) THEN
         FLOW =
     $ DMAX1(((pstart-pend-a(arc))/C2(ARC)),0.D0)
          COST(ARC) = C2(ARC)*FLOW+A(ARC)
        ELSE
          FLOW =0.0
        END IF
       FLOW=F(ARC)-FLOW
C       IF (FLOW.GT.REF_THR) THEN
         SURPLUS(START)=SURPLUS(START)+FLOW
         SURPLUS(END)=SURPLUS(END)-FLOW
         F(ARC)=F(ARC)-FLOW
C       END IF
       END IF
      END IF
900     CONTINUE


C     RETURN FOR ANOTHER PHASE

C     PRINT*,'CHECKING E-CS BEFORE STARTING NEW PHASE'
C      DO 700 ARC=1,NA
C        IF (F(ARC).GT.0) THEN
C       IF (P(STARTN(ARC))-P(ENDN(ARC)).LT.-EPS+COST(ARC)) THEN
C        PRINT*,'E-CS VIOLATED AT ARC ',ARC
C       ENDIF
C      ENDIF
C        IF (F(ARC).LT.U(ARC)) THEN
C       IF (P(STARTN(ARC))-P(ENDN(ARC)).GT.EPS+COST(ARC)) THEN
C        PRINT*,'E-CS VIOLATED AT ARC ',ARC
C       ENDIF
C      ENDIF
700    CONTINUE
       GO TO 60

400    PRINT*,'PROBLEM IS INFEASIBLE'
C       PAUSE
       STOP

       END
```