# รายการอ้างอิง


ภาษาไทย

ปราโมทย์ เดชะอำไพ. ไฟไนต์เอลิเมนต์ในงานวิศวกรรม. กรุงเทพมหานคร: สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, 2537.

ปราโมทย์ เดชะอำไพ. ระเบียบวิธีเชิงตัวเลขในงานวิศวกรรม. กรุงเทพมหานคร: สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, 2538.

ภาษาอังกฤษ

Anderson, J. D. Jr. Computational Fluid Dynamics: The Basic with Application. New York: McGraw–Hill, 1995.

Anderson, J. D. Jr. Fundamentals of Aerodynamics. Second Ed. New York: McGraw–Hill, 1991.

Anderson, J. D. Jr. Modern Compressible Flow with Historical Prospective. 2nd ed. New York: McGraw–Hill, 1990.

Dechaumphai, P. Adaptive finite element technique for heat transfer problems. Journal of Energy, Heat & Mass Transfer. 17(1995): 87-94.

Dechaumphai, P., and Janphaisaeng, P. Adaptive finite element technique for high-speed compressible flows. Thammasart International Journal of Science and Technology. 3(1998): 21-30.

Dechaumphai, P., and Weiting, A. R. Coupled fluid-thermal-structural analysis for aerodynamically heated structures. Finite Element Analysis in Fluids. (April 1989): 165-171.

Gnoffo, P. A. Application of program LAURA to three-dimension AOTV flowfields. AIAA Paper86-0565. (January 1986): 1-12.

Hirsch, Ch. Numerical Computation of Internal and External Flows. New York: John Wiley & Sons, 1990.

Hughes, T. J. R. Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier-Stokes equations. International Journal for Numerical Methods in Fluids. 7(1987): 1261-1275.

Jiang, B. N., and Carey, G. F. A stable least-Squares finite element method for non-linear hyperbolic problems. International Journal for Numerical Methods in Fluids. 8(1988): 933-942.

Peraire, J.; Vahjdati, M.; Morgan, K.; and Zienkiewicz, O. C. Adaptive remeshing for compressible flow computation. Journal of Computational Physics. 72(1987): 449-466.

Roe, P. L. Approximate riemann solver, parameter vectors and finite differences. Journal of Computational Physics. 43(1981): 357-372.

Zienkiewicz, O. C., and Taylour, R. L. The Finite Element Method. 4th ed. New York: McGraw-Hill, 1991.

# ภาคผนวก

# ภาคผนวก ก

## รายละเอียดของโปรแกรม FINITE

โปรแกรม FINITE จะมีรายละเอียดเริ่มจากโปรแกรมหลักและตามด้วยโปรแกรมย่อยต่างๆ ทั้งหมดดังนี้

```
C     PROGRAM FINITE
C
C     A FINITE ELEMENT COMPUTER PROGRAM FOR SOLVING THE EULER EQUATIONS
C     OF INVISCID COMPRESSIBLE HIGH SPEED FLOW.
C
C     THE VALUES DECLARED IN THE PARAMETER STATEMENT BELOW SHOULD BE
C     ADJUSTED ACCORDING TO THE SIZE OF THE PROBLEMS :
C        MXPOI = MAXIMUM NUMBER OF NODES IN THE MODEL.
C        MXELE = MAXIMUM NUMBER OF ELEMENTS IN THE MODEL.
C        MXBOU = MAXIMUM NUMBER OF BOUNDARIES IN THE MODEL.
C        MXSID = MAXIMUM NUMBER OF SIDES IN THE MODEL.
C              = (3*NELEM+NBOUN)/2
C
      PARAMETER  (MXPOI = 494, MXELE = 893, MXBOU = 93, MXSID = 1386)
C
      IMPLICIT  REAL*8(A-H,O-Z)
C
      DIMENSION  COORD(MXPOI,2), UNKNP(MXPOI,4), UNKNP1(MXPOI,4)
      DIMENSION  UNKNO(MXELE,4), UNKN1(MXELE,4), RSIDO(MXSID,3)
      DIMENSION  RHSO(MXELE,4) , DELTE(MXELE)  , AREA(MXELE)
      DIMENSION  AMLP(MXPOI)   , SLEN(MXSID,2)
      DIMENSION  SUMSQ(4)
C
      INTEGER  INTMA(MXELE,3), BSIDO(MXBOU,4)
      INTEGER  IDUM1(MXPOI)  , IDUM2(MXPOI)  , IDUM3(4*MXELE)
      INTEGER  ISIDE(MXSID,4), JESID(MXELE,4)
C
      CHARACTER FILNAM*4, CV*2
C
C     INPUT FILE NAME AND VERSION.
C
      WRITE(*,'(/,A,$)') ' PLEASE TYPE INPUT FILE NAME : '
      READ(*,'(A)') FILNAM
      L = NAMLEN(FILNAM)
      IF (L.EQ.0) GO TO 800
      WRITE(*,'(A,$)')   ' PLEASE TYPE CURRENT VERSION : '
      READ(*,'(A)') CV
C
C     OPEN INPUT FILE.
C
      OPEN(UNIT= 5,FILE=FILNAM(1:L)//'.IN'//CV,STATUS='OLD',ERR=800)
C
C     OPEN OUTPUT FILES : *.SO_ = SOLUTION FILE
C                                 *.CO_ = CONVERGENCE FILE
C                                 *.EX_ = EXCEL FILE
C                                 *.JU_ = FEPLOT FILE
C                                 *.X_  = NASTRAN OUTPUT FILE
C                                 *.RN_ = ADAPTIVE FILE
C
      OPEN(UNIT= 7,FILE=FILNAM(1:L)//'.SO'//CV,STATUS='UNKNOWN',ERR=800)
      OPEN(UNIT= 8,FILE=FILNAM(1:L)//'.CO'//CV,STATUS='UNKNOWN',ERR=800)
      OPEN(UNIT=22,FILE=FILNAM(1:L)//'.EX'//CV,STATUS='UNKNOWN',ERR=800)
      OPEN(UNIT=23,FILE=FILNAM(1:L)//'.JU'//CV,STATUS='UNKNOWN',ERR=800)
      OPEN(UNIT=24,FILE=FILNAM(1:L)//'.X' //CV,STATUS='UNKNOWN',ERR=800)
C     OPEN(UNIT=25,FILE=FILNAM(1:L)//'.VO'//CV,STATUS='UNKNOWN',ERR=800)
```

```
      OPEN(UNIT=26,FILE=FILNAM(1:L)//'.RN'//CV,STATUS='UNKNOWN',ERR=800)
      OPEN(UNIT=27,FILE=FILNAM(1:L)//'.SHO'   ,STATUS='UNKNOWN',ERR=800)
C
      NDIMN = 2
      NAMAT = NDIMN + 2
      NNODE = NDIMN + 1
      NBNOR = NDIMN + 1
      NBNOI = NDIMN + 2
C
C     READ INPUT DATA AND ADJUST TO ELEMENT QUANTITIES.
C
      CALL INPUT(NELEM, NPOIN, NBOUN, GAMMA, EPSLAM, NAMAT, NTIME,
     *           IVISC, IMP,   NODEQ, IDISS,  QDOT,  CSAFE, INTMA,
     *           COORD, UNKNP, UNKNP1, UNKNO, BSIDO, MXPOI, MXELE,
     *           MXBOU)
C
      WRITE(6,101)
      WRITE(27,101)
  101 FORMAT(1X, 'SUB. INPUT COMPLETED')
C
C     COMPUTE TOTAL NUMBER OF SIDES.
C
      NSIDE = (3.*NELEM + NBOUN)/2.
C
C     IDENTIFY THE SIDE: DETERMINE ARRAYS ISIDE(NSIDE,4) AND
C     JESID(NELEM,4).
C
      CALL SIDE(NELEM, NPOIN, NSIDE, INTMA, ISIDE, IDUM1, IDUM2, IDUM3,
     *          NBOUN, NBNOI, BSIDO, JESID, COORD, RSIDO)
C
      WRITE(6,221)
      WRITE(27,221)
  221 FORMAT(1X, 'SUB. SIDE COMPLETED')
C
C     COMPUTE ELEMENT AREAS AND ALL ELEMENT MATRICES NEEDED.
C
      CALL GETMAT(NDIMN, NELEM, NPOIN, INTMA, COORD, AMLP, AREA)
C
      WRITE(6,301)
      WRITE(27,301)
  301 FORMAT(1X,'SUB. GETMAT COMPLETED')
C
C     DETERMINE REPRESENTATIVE ELEMENT LENGTHS FOR LOCAL TIME STEP
C     COMPUTATION.
C
      CALL ELELEN(NDIMN, NNODE, NSIDE, NPOIN, NELEM, INTMA, COORD,
     *            ISIDE, JESID,  SLEN)
C
      WRITE(6,401)
      WRITE(27,401)
  401 FORMAT(1X,'SUB. ELELEN COMPLETED')
C
C     PERFORM TIME STEP ITERATIONS :
C
      DO 456  ITIME=1,NTIME
C
C     STORE ELEMENT UNKNOWNS IN  UNKN1(NELEM,NAMAT).
C
      DO 500  IA=1,NAMAT
      DO 500  IE=1,NELEM
      UNKN1(IE,IA) = UNKNO(IE,IA)
  500 CONTINUE
C
      CALL COMPUTE(NAMAT, NSIDE, NBNOR, NPOIN, NELEM, ISIDE, RSIDO,
     *             UNKNO, UNKN1, UNKNP, AREA,  RHSO,  CSAFE, GAMMA,
     *             EPSLAM, DELTE, SLEN)
C
C     COMPUTE RESIDUALS.
C
      DO 550  IA=1,NAMAT
      SUMSQ(IA) = 0.
      DO 600  IE=1,NELEM
      DIFF = UNKNO(IE,IA) - UNKN1(IE,IA)
      SUMSQ(IA) = SUMSQ(IA) + DIFF*DIFF
  600 CONTINUE
      SUMSQ(IA) = SQRT(SUMSQ(IA))
  550 CONTINUE
```

```
C
      WRITE(8,651)   ITIME, (SUMSQ(IA), IA=1,4)
      WRITE(6,651)   ITIME, (SUMSQ(IA), IA=1,4)
      WRITE(27,651)  ITIME, (SUMSQ(IA), IA=1,4)
  651 FORMAT(I6, 4E12.5)
C
      SUML = SUMSQ(1)
      SUMLOG = LOG10(SUML)
      AHALF = ITIME/100.
      IHALF = ITIME/100
      IF(IHALF.EQ.AHALF) THEN
      WRITE(22,661)  ITIME, SUMSQ(1), SUMLOG
  661 FORMAT(I8, 2(2X,E12.5))
      END IF
C
  456 CONTINUE
C
C     CONVERT ELEMENT QUANTITIES TO NODAL QUANTITIES
C     AND PRINT OUT SOLUTIONS.
C
      CALL OUTPUT(NNODE, NAMAT, NBOUN, NPOIN, NELEM, BSIDO, UNKNO,
     *            UNKNP, UNKNP1,COORD, INTMA, IDUM1, GAMMA)
C
      WRITE(6,701)
      WRITE(27,701)
  701 FORMAT(1X, 'SUB. OUTPUT COMPLETED')
C
C     PROGRAM RUN COMPLETELY
C
      WRITE(6,751)
  751 FORMAT(1X, 'PROGRAM COMPLETED')
C
      GOTO 900
  800 WRITE(*,*)' *** FILE NAME NOT EXIST *** '
C
  900 STOP
      END
C*--------------------------------------------------------------------*
C*   [NAMLEN] COUNTS THE NUMBER OF CHARACTERS IN FILNAM.              *
C*--------------------------------------------------------------------*
      INTEGER FUNCTION NAMLEN(FILNAM)
C
      CHARACTER*4 FILNAM
C
      NAMLEN = 0
      DO 10 I = 4,1,-1
      IF (FILNAM(I:I).EQ.' ') GO TO 10
      NAMLEN = I
      GO TO 20
   10 CONTINUE
   20 RETURN
      END
C*--------------------------------------------------------------------*
C*   [SUB. INPUT] READ INPUT DATA AND ADJUST TO ELEMENT QUANTITIES.   *
C*--------------------------------------------------------------------*
      SUBROUTINE INPUT(NELEM, NPOIN, NBOUN, GAMMA, EPSLAM, NAMAT, NTIME,
     *                 IVISC, IMP,   NODEQ, IDISS,  QDOT,  CSAFE, INTMA,
     *                 COORD, UNKNP, UNKNP1, UNKNO, BSIDO, MXPOI,
     *                 MXELE, MXBOU)
C
      IMPLICIT  REAL*8(A-H,O-Z)
C
C     DIMENSION COORD(NPOIN,2), UNKNP(NPOIN,4), UNKNP1(NPOIN,4)
C     DIMENSION UNKNO(NELEM,4)
C
C     INTEGER INTMA(NELEM,3), BSIDO(NBOUN,4)
C
C
      DIMENSION COORD(MXPOI,2), UNKNP(MXPOI,4), UNKNP1(MXPOI,4)
      DIMENSION UNKNO(MXELE,4)
C
      INTEGER INTMA(MXELE,3), BSIDO(MXBOU,4)
C
C
C     READ NUMBER OF ELEMENTS, NODES, AND BOUNDARIES:
C
      READ(5,1)  TEXT
```

```
C     WRITE(6,1) text
    1 FORMAT(10A8)
C     READ(5,3)  NELEM, NPOIN, NBOUN
C     WRITE(6,3) nelem, npoin, nboun
C   3   FORMAT(3I8)
      READ(5,*)  NELEM, NPOIN, NBOUN
C
C     READ  GAMMA, EPSLAM, NTIME :
C
      READ(5,1)  TEXT
C     WRITE(6,1) text
C     READ(5,7)  GAMMA, EPSLAM, NTIME
C     WRITE(6,7) gamma, epslam, ntime
C   7 FORMAT(F12.6,3X,F14.7,1X,I12)
      READ(5,*)  GAMMA, EPSLAM, NTIME
C
      IVISC = 0
      IMP = 0
      NODEQ = 1
      IDISS = 1
      QDOT = 0.
      CSAFE = 0.5
C
C     READ  ELEMENT NODAL CONNECTIONS :
C
      READ(5,1)  TEXT
C     WRITE(6,1) text
      DO 10  I=1,NELEM
C     READ(5,12)  IE, (INTMA(I,J), J=1,3)
C     WRITE(6,12)  ie, (intma(i,j), j=1,3)
C  12 FORMAT(4I8)
      READ(5,*)  IE, (INTMA(I,J), J=1,3)
   10 CONTINUE
C
C     READ ELEMENT NODAL COORDINATES :
C
      READ(5,1)  TEXT
C     WRITE(6,1)  text
      DO 20  I=1,NPOIN
C     READ(5,22)  N, (COORD(I,J), J=1,2)
C     WRITE(6,22)  n, (coord(i,j), j=1,2)
C  22 FORMAT(I8,2E30.14)
      READ(5,*)  N, (COORD(I,J), J=1,2)
   20 CONTINUE
C
C     READ NODAL INITIAL CONDITIONS :
C
      READ(5,1)  TEXT
C     WRITE(6,1) text
      DO 30  I=1,NPOIN
C     READ(5,32)  N, (UNKNP(I,J), J=1,4)
C     WRITE(6,32)  n, (unknp(i,j), j=1,4)
C  32 FORMAT(I8,4E16.8)
      READ(5,*)  N, (UNKNP(I,J), J=1,4)
   30 CONTINUE
C
C     SAVE THESE NODAL INITIAL CONDITIONS.
C
      DO 40  J=1,NAMAT
      DO 40  I=1,NPOIN
      UNKNP1(I,J) = UNKNP(I,J)
   40 CONTINUE
C
C     OBTAIN NODAL CONSERVATION VARIABLES.
C
      DO 50  J=2,NAMAT
      DO 50  I=1,NPOIN
      UNKNP(I,J) = UNKNP(I,1)*UNKNP(I,J)
   50 CONTINUE
C
C     COMPUTE ELEMENT QUANTITIES.
C
      C13 = 1./3.
      DO 60  IE=1,NELEM
      NCOUNT = 3
      FACTOR = C13
      DO 60  IA=1,NAMAT
```

```
      UNKNO(IE,IA) = 0.
      DO 60  IN=1,NCOUNT
      IP = INTMA(IE,IN)
      UNKNO(IE,IA) = UNKNO(IE,IA) + FACTOR*UNKNP(IP,IA)
   60 CONTINUE
C
C     READ NODAL BOUNDARY CONDITIONS :
C          BSIDO(NBOUN,1)  --  1ST NODE ON THE SIDE
C          BSIDO(NBOUN,2)  --  2ND NODE ON THE SIDE
C          BSIDO(NBOUN,3)  --  ELEMENT NO. OF THAT SIDE
C          BSIDO(NBOUN,4)  --  B.C. OF THAT SIDE
C
      READ(5,1)  TEXT
C     WRITE(6,1)  text
      DO 70  IB=1,NBOUN
C     READ(5,72)  (BSIDO(IB,J), J=1,4)
C     WRITE(6,72)  (bsido(ib,j), j=1,4)
C  72 FORMAT(4I8)
      READ(5,*)  (BSIDO(IB,J), J=1,4)
   70 CONTINUE
C
      RETURN
      END
C*--------------------------------------------------------------------*
C*   [SUB. SIDE] IDENTIFY THE SIDES.                                  *
C*--------------------------------------------------------------------*
      SUBROUTINE SIDE(NELEM, NPOIN, NSIDE, INTMA, ISIDE, LWHER, LHOWM,
     *                ICONE, NBOUN, NBNOI, BSIDO, JESID, COORD, RSIDO)
C
      IMPLICIT  REAL*8(A-H,O-Z)
C
      DIMENSION INTMA(NELEM,3), ISIDE(NSIDE,4), COORD(NPOIN,2)
      DIMENSION LWHER(NPOIN),LHOWM(NPOIN),ICONE(4*NELEM)
      DIMENSION JESID(NELEM,4),RSIDO(NSIDE,3)
C
      INTEGER BSIDO(NBOUN,NBNOI)
C
C     FILL IN LHOWM : NO. OF ELEMENTS PER NODE.
C
      DO 110 IS=1,NSIDE
      ISIDE(IS,3)=0
      ISIDE(IS,4)=0
  110 CONTINUE
C
      DO 115 IP=1,NPOIN
      LHOWM(IP)=0
  115 CONTINUE
C
      DO 120 IE=1,NELEM
      NCOUNT=3
      DO 120 IN=1,NCOUNT
      IP=INTMA(IE,IN)
      LHOWM(IP)=LHOWM(IP)+1
  120 CONTINUE
C
C     FILL IN LWHER : LOCATION OF EACH NODE INSIDE ICONE.
C
      LWHER(1)=0
      DO 125 IP=2,NPOIN
      LWHER(IP)=LWHER(IP-1)+LHOWM(IP-1)
  125 CONTINUE
C
C     FILL IN ICONE : ELEMENTS IN EACH NODE.
C
      DO 130 IP=1,NPOIN
      LHOWM(IP)=0
  130 CONTINUE
      DO 135 IE=1,NELEM
      NCOUNT=3
      DO 135 IN=1,NCOUNT
      IP=INTMA(IE,IN)
      LHOWM(IP)=LHOWM(IP)+1
      JLOCA=LWHER(IP)+LHOWM(IP)
      ICONE(JLOCA)=IE
  135 CONTINUE
C
C     LOOP OVER THE NODES.
```

```
C
      ILOCA=0
C
      DO 140 IP=1,NPOIN
      ILOC1=ILOCA
      IELE=LHOWM(IP)
      IF(IELE.EQ.0) GOTO 140
C
      IWHER=LWHER(IP)
C
C     LOOP OVER ELEMENTS SURROUNDING THE POINT IP.
C
      IP1=IP
      DO 145 IEL=1,IELE
      IE=ICONE(IWHER+IEL)
      NCOUNT=3
C
C     FIND OUT POSITION OF IP IN THE CONECTIVITY MATRIX.
C
      DO 150 IN=1,NCOUNT
      IN1=IN
      IPT=INTMA(IE,IN)
      IF(IPT.EQ.IP) GOTO 155
  150 CONTINUE
  155 CONTINUE
C
      J=0
      DO 160 JNOD=1,NCOUNT-1,NCOUNT-2
      J=J+1
      IN2=IN1+JNOD
      IF(IN2.GT.NCOUNT) IN2=IN2-NCOUNT
      IP2=INTMA(IE,IN2)
      IF(IP2.LT.IP1) GOTO 160
C
C     CHECK THE SIDE -----> NEW OR OLD.
C
      IF(ILOCA.EQ.ILOC1) GOTO 165
      DO 170 IS=ILOC1+1,ILOCA
      JLOCA=IS
      IF(ISIDE(IS,2).EQ.IP2) GOTO 175
  170 CONTINUE
  165 CONTINUE
C
C     NEW SIDE.
C
      ILOCA=ILOCA+1
      ISIDE(ILOCA,1)=IP1
      ISIDE(ILOCA,2)=IP2
      ISIDE(ILOCA,2+J)=IE
      GOTO 180
C
C     OLD SIDE
C
  175 CONTINUE
      ISIDE(JLOCA,2+J)=IE
  180 CONTINUE
C
  160 CONTINUE
C
C     END LOOP OVER ELEMENTS SURROUNDING POINT IP.
C
  145 CONTINUE
C
      DO 185 IS=ILOC1+1,ILOCA
      IF(ISIDE(IS,3).NE.0) GOTO 185
      ISIDE(IS,3)=ISIDE(IS,4)
      ISIDE(IS,4)=0
      ISIDE(IS,1)=ISIDE(IS,2)
      ISIDE(IS,2)=IP1
  185 CONTINUE
C
C     END LOOP OVER POINTS.
C
  140 CONTINUE
C**   DO 234 IS=1,NSIDE
C**   WRITE(6,3111) IS,(ISIDE(IS,IJ),IJ=1,4)
C3111 FORMAT(5I5)
```

```
C*234 CONTINUE
C
C ***  NOW RESET THE BOUNDARY MARKERS.
C
      DO 190 IS=1,NSIDE
      IF(ISIDE(IS,4).NE.0)GO TO 190
      IL=ISIDE(IS,1)
      IR=ISIDE(IS,2)
      IE=ISIDE(IS,3)
      DO 195 IB=1,NBOUN
      IBE=BSIDO(IB,3)
      IF(IBE.NE.IE)GO TO 195
      ILB=BSIDO(IB,1)
      IRB=BSIDO(IB,2)
      IF(ILB.NE.IL.OR.IRB.NE.IR)GO TO 195
      ISIDE(IS,4)=-BSIDO(IB,4)
      GO TO 190
  195 CONTINUE
  190 CONTINUE
C**   DO 235 IS=1,NSIDE
C**   WRITE(6,3111) IS,(ISIDE(IS,IJ),IJ=1,4)
C*235 CONTINUE
C
C *** FORM THE ELEMENT/SIDES CONNECTIVITY ARRAY.
C
      DO 200 IS=1,NSIDE
      IEL=ISIDE(IS,3)
      IER=ISIDE(IS,4)
      INODE=ISIDE(IS,1)
      JNODE=ISIDE(IS,2)
      NCOUNT=3
      DO 205 IN=1,NCOUNT
      I1=INTMA(IEL,IN)
      IN1=IN+1
      IF(IN1.GT.NCOUNT)IN1=1
      I2=INTMA(IEL,IN1)
      IF(INODE.EQ.I1.AND.JNODE.EQ.I2)
     .JESID(IEL,IN)=IS
  205 CONTINUE
      IF(IER.GT.0) THEN
      NCOUNT=3
      DO 210 IN=1,NCOUNT
      I1=INTMA(IER,IN)
      IN1=IN+1
      IF(IN1.GT.NCOUNT)IN1=1
      I2=INTMA(IER,IN1)
      IF(INODE.EQ.I2.AND.JNODE.EQ.I1)
     .JESID(IER,IN)=IS
  210 CONTINUE
      ENDIF
C
  200 CONTINUE
C**   WRITE(23,9901)(IE,(JESID(IE,IN),IN=1,4),IE=1,NELEM)
C**   WRITE(23,9901)(IS,(ISIDE(IS,IN),IN=1,4),IS=1,NSIDE)
C9901 FORMAT(5I5)
C
C     COMPUTE COMPONENTS OF UNIT NORMAL VECTOR TO THE SIDE
C     AND THE SIDE LENGTHS.
C
      DO 215  IS=1,NSIDE
      IPI = ISIDE(IS,1)
      IPJ = ISIDE(IS,2)
      DX  = COORD(IPJ,1) - COORD(IPI,1)
      DY  = COORD(IPJ,2) - COORD(IPI,2)
      DL  = SQRT(DX*DX + DY*DY)
      RSIDO(IS,1) =  DY/DL
      RSIDO(IS,2) = -DX/DL
      RSIDO(IS,3) =  DL
  215 CONTINUE
C
      RETURN
      END
C*--------------------------------------------------------------------*
C*   [SUB. GETMAT] COMPUTE ALL ELEMENT AREAS ANS MATRICES.            *
C*--------------------------------------------------------------------*
      SUBROUTINE GETMAT(NDIMN, NELEM, NPOIN, INTMA, COORD, AMLP, AREA)
C
```

```
      IMPLICIT  REAL*8(A-H,O-Z)
C
      DIMENSION  COORD(NPOIN,NDIMN), AMLP(NPOIN), AREA(NELEM)
      DIMENSION  X(3), Y(3)
C
      INTEGER  INTMA(NELEM,3)
C
      DO 230  IP=1,NPOIN
      AMLP(IP) = 0.
  230 CONTINUE
C
C     LOOP OVER ALL ELEMENTS.
C
      DO 240  IE=1,NELEM
      DO 250  IA=1,3
      N = INTMA(IE,IA)
      X(IA) = COORD(N,1)
      Y(IA) = COORD(N,2)
  250 CONTINUE
C
      B1 = Y(2) - Y(3)
      B2 = Y(3) - Y(1)
      B3 = Y(1) - Y(2)
      C1 = X(3) - X(2)
      C2 = X(1) - X(3)
      C3 = X(2) - X(1)
C
      AREA(IE) = 0.5*(X(1)*B1 + X(2)*B2 + X(3)*B3)
C**   WRITE(23,1111)  IE, AREA(IE)
C1111 FORMAT(I8, E16.5)
      AMLT     = AREA(IE)/3.
      DO 260  IA=1,3
      N = INTMA(IE,IA)
      AMLP(N) = AMLP(N) + AMLT
  260 CONTINUE
C
  240 CONTINUE
C
      RETURN
      END
C*--------------------------------------------------------------------*
C*   [SUB. ELELEN] COMPUTE ELEMENT LENGTH FOR LOCAL TIME STEP.        *
C*--------------------------------------------------------------------*
      SUBROUTINE ELELEN(NDIMN, NNODE, NSIDE, NPOIN, NELEM, INTMA, COORD,
     *                  ISIDE, JESID,  SLEN)
C
C     DETERMINE REPRESENTATIVE 'ELEMENT LENGTHS' FOR TIME STEP
C     COMPUTATION.  THERE ARE 2 VALUES FOR EACH SIDE, EACH REPRESENTS
C     THE NORMAL DISTANCE FROM THE ELEMENT CENTROIDS (2 ELEMENTS
C     ON BOTH SIDES) TO THE SIDE CONSIDERED.
C
      IMPLICIT  REAL*8(A-H,O-Z)
C
      DIMENSION  COORD(NPOIN,NDIMN), SLEN(NSIDE,2)
C
      INTEGER  INTMA(NELEM,NNODE)
      INTEGER  ISIDE(NSIDE,4), JESID(NELEM,4)
C
      TOL = 1.E-10
C
C     LOOP OVER NUMBER OF ELEMENTS.
C
      DO 310  IE=1,NELEM
      XC = 0.0
      YC = 0.0
      DO 320  IN=1,3
      IP = INTMA(IE,IN)
      XC = XC + COORD(IP,1)
      YC = YC + COORD(IP,2)
  320 CONTINUE
      XC = XC/3.
      YC = YC/3.
      DO 330  IN=1,3
      IS = JESID(IE,IN)
      IL = ISIDE(IS,1)
      IR = ISIDE(IS,2)
      IF(IL.EQ.INTMA(IE,IN)) THEN
```

```
      IES=1
      ELSE
      IES=2
      ENDIF
      XL = COORD(IL,1)
      YL = COORD(IL,2)
      XR = COORD(IR,1)
      YR = COORD(IR,2)
      IF(ABS(YR-YL).LT.TOL) THEN
      DIST = ABS(YC-YL)
      ELSE
      IF(ABS(XR-XL).GT.TOL) THEN
      RM   = (YR-YL)/(XR-XL)
      C    = YL - RM*XL
      RM1  =-1.0/RM
C
C     COMPUTE INTERSECTION POINT.
C
      XP   = (YC - RM1*XC - C)/(RM - RM1)
      YP   = RM1*(XP - XC) + YC
      DIST = SQRT((XP-XC)*(XP-XC) + (YP-YC)*(YP-YC))
      ELSE
      DIST = ABS(XC-XL)
      ENDIF
      ENDIF
      SLEN(IS,IES) = DIST
C
C     DEAL WITH BOUNDARY ELEMENT LENGTHS.
C
      IEL = ISIDE(IS,3)
      IER = ISIDE(IS,4)
      IF(IEL.GT.0)  GO TO 340
      SLEN(IS,1) = SLEN(IS,2)
      GO TO 350
  340 IF(IER.GT.0)  GO TO 350
      SLEN(IS,2) = SLEN(IS,1)
  350 CONTINUE
C
C     WRITE(23,998)  IS, IEL, IER, SLEN(IS,1), SLEN(IS,2)
C 998 FORMAT(3I5, 2E15.7)
C
  330 CONTINUE
  310 CONTINUE
C
      RETURN
      END
C*--------------------------------------------------------------------*
C*   [SUB. COMPUTE] COMPUTE SOLUTIONS BY EXPLICIT, CELL CENTERED,     *
C*                  UPWIND AND ROE'S AVERAGING METHOD.                *
C*--------------------------------------------------------------------*
      SUBROUTINE COMPUTE(NAMAT, NSIDE, NBNOR, NPOIN, NELEM, ISIDE,
     *                   RSIDO, UNKNO, UNKN1, UNKNP, AREA,  RHS0,
     *                   CSAFE, GAMMA, EPSLAM,DELTE, SLEN)
C
      IMPLICIT  REAL*8(A-H,O-Z)
C
C     DIMENSION  COORD(NPOIN,NDIMN), AMLP(NPOIN)
      DIMENSION  RHS0(NELEM,NAMAT)
      DIMENSION  UNKNO(NELEM,NAMAT), UNKN1(NELEM,NAMAT)
      DIMENSION  RSIDO(NSIDE,NBNOR), DELTE(NELEM), AREA(NELEM)
      DIMENSION  UNKNP(NPOIN,NAMAT)
      DIMENSION  SLEN(NSIDE,2)
      DIMENSION  RLAM(4), R(4,4), RI(4,4), DU(4)
      DIMENSION  DISS(4), FSUM(4), FLUX(4), AVROE(4,4)
C
C     INTEGER  INTMA(NELEM,NNODE), BSIDO(NBOUN,4), JESID(NELEM,4)
      INTEGER  ISIDE(NSIDE,4)
C
      GAM1 = GAMMA - 1.
C
C     INITIALIZE ELEMENT TIME STEPS.
C
      DO 20  IE=1,NELEM
      DELTE(IE) = 1.E+10
   20 CONTINUE
C
C     INITIALIZE  RHS0  VECTOR:
```

```
C
      DO 30  IA=1,NAMAT
      DO 30  IE=1,NELEM
      RHSO(IE,IA) = 0.
   30 CONTINUE
C
C     LOOP OVER THE SIDES:
C
      DO 1000 IS=1,NSIDE
C
C     IDENTIFY THE LEFT AND RIGHT ELEMENT NUMBERS.
C
      IEL = ISIDE(IS,3)
      IER = ISIDE(IS,4)
C
C     GET COMPONENTS OF NORMAL VECTOR FOR THE SIDE CONSIDERED.
C
      RNX  = RSIDO(IS,1)
      RNY  = RSIDO(IS,2)
      RLEN = RSIDO(IS,3)
C
C     COLLECT THE "LEFT" ELEMENT VALUES.
C
      RHOL = UNKNO(IEL,1)
      UXL  = UNKNO(IEL,2)/RHOL
      VYL  = UNKNO(IEL,3)/RHOL
      TEL  = UNKNO(IEL,4)/RHOL
      PRESL= GAM1*(UNKNO(IEL,4) - 0.5*RHOL*(UXL*UXL+VYL*VYL))
C     IF(IVISC.EQ.1)  TEMPL = (TEL - 0.5*(UXL*UXL+VYL*VYL))/CV
C
C     "LEFT" NORMAL AND TANGENTIAL VELOCITIES AND TOTAL ENTHALPY.
C
      UNL =  UXL*RNX + VYL*RNY
      VTL = -UXL*RNY + VYL*RNX
      UL2 =  UNL*UNL + VTL*VTL
      HL  =  GAMMA*TEL - 0.5*GAM1*UL2
C
C     IS THIS SIDE ON THE ACTUAL FLOW BOUNDARY ?
C
      IF(IER.GT.0)  GO TO 100
C
C     APPLY BOUNDARY CONDITIONS.
C
      IF(IER.EQ.-1)  GO TO 110
      IF(IER.EQ.-2)  GO TO 120
      IF(IER.EQ.-3)  GO TO 130
      GO TO 100
C
C     SUPERSONIC INFLOW.
C
  110 CONTINUE
      II   = ISIDE(IS,1)
      JJ   = ISIDE(IS,2)
      RHOR = 0.5*(                UNKNP(II,1) +              UNKNP(JJ,1))
      UXR  = 0.5*(UNKNP(II,2)/UNKNP(II,1) + UNKNP(JJ,2)/UNKNP(JJ,1))
      VYR  = 0.5*(UNKNP(II,3)/UNKNP(II,1) + UNKNP(JJ,3)/UNKNP(JJ,1))
      TER  = 0.5*(UNKNP(II,4)/UNKNP(II,1) + UNKNP(JJ,4)/UNKNP(JJ,1))
      PRESR= GAM1*(RHOR*TER - 0.5*RHOR*(UXR*UXR+VYR*VYR))
      GO TO 200
C
C     SUPERSONIC OUTFLOW.
C
  120 CONTINUE
      RHOR = RHOL
      UXR  = UXL
      VYR  = VYL
      TER  = TEL
      PRESR= PRESL
      GO TO 200
C
C     INVISCID WALL.
C
  130 CONTINUE
      RHOR = RHOL
      UXR  =-RNX*UNL - RNY*VTL
      VYR  =-RNY*UNL + RNX*VTL
      PRESR= PRESL
```

```
      TER  = (PRESR/(GAM1*RHOR)) + 0.5*(UXR*UXR+VYR*VYR)
C**   WRITE(23,2444)  (ISIDE(IS,KK), KK=1,4), RNX, RNY, RLEN
C2444 FORMAT(4I8, 3E12.5)
      GO TO 200
C
C     THE RIGHT SIDE IS CONNECTED TO ACTUAL ELEMENT.
C
  100 CONTINUE
      RHOR = UNKNO(IER,1)
      UXR  = UNKNO(IER,2)/RHOR
      VYR  = UNKNO(IER,3)/RHOR
      TER  = UNKNO(IER,4)/RHOR
      PRESR= GAM1*(UNKNO(IER,4) - 0.5*RHOR*(UXR*UXR+VYR*VYR))
C
  200 CONTINUE
C
C     "RIGHT" NORMAL AND TANGENTIAL VELOCITIES AND TOTAL ENTHALPY.
C
      UNR =  UXR*RNX + VYR*RNY
      VTR = -UXR*RNY + VYR*RNX
      UR2 =  UNR*UNR + VTR*VTR
      HR  =  GAMMA*TER - 0.5*GAM1*UR2
C
C     COMPUTE INTERFACE VALUES.
C
      BI = SQRT(RHOR/RHOL)
      AI = 1./(1.+BI)
      UI = (BI*UXR + UXL)*AI
      VI = (BI*VYR + VYL)*AI
      HI = (BI*HR  + HL )*AI
      CI2= GAM1*(HI - 0.5*(UI*UI+VI*VI))
      IF(CI2.LT.0.) THEN
      WRITE(6,211)  IER, IEL
  211 FORMAT(2X,'NEGATIVE SOUND SPEED BETWEEN ELEMENTS', 2I5)
      STOP
      END IF
      CI   = SQRT(CI2)
      UCAP = UI*RNX + VI*RNY
      VCAP =-UI*RNY + VI*RNX
      CX   = CI*RNX
      CY   = CI*RNY
      ALP  = 0.5*(UI*UI + VI*VI)
C
C     COMPUTE THE FOUR ABSOLUTE EIGENVALUES.
C
      RLAM(1) = ABS(UCAP)
      RLAM(2) = ABS(UCAP)
      RLAM(3) = ABS(UCAP+CI)
      RLAM(4) = ABS(UCAP-CI)
C
C     RESET THESE EIGENVALUES SO THAT THE RANGE IS FROM ZERO TO ONE.
C
      EIGMAX = ABS(UCAP) + CI
      DO 220  IR=1,4
      RLAM(IR) = RLAM(IR)/EIGMAX
  220 CONTINUE
C
      EPSACT = EPSLAM
C
C     RESET THESE EIGENVALUES IF THEY ARE LESS THAN EPSLAM.
C
      DO 230  IR=1,4
      IF(RLAM(IR).GE.EPSACT)  GO TO 230
      RLAM(IR) = 0.5*(RLAM(IR)*RLAM(IR)/EPSACT + EPSACT)
  230 CONTINUE
C
C     RESET BACK THE CORRECT (DIMENSION) EIGENVALUES.
C
      DO 240  IR=1,4
      RLAM(IR) = RLAM(IR)*EIGMAX
  240 CONTINUE
C
C     COMPUTE ELEMENT TIME STEP ASSOCIATED WITH THIS SIDE.
C
      REPLEN = SLEN(IS,1) + SLEN(IS,2)
      EIGMAX = ABS(UCAP) + CI
      AUX    = 0.
```

```
      DTL    = CSAFE*(REPLEN/EIGMAX)/(1. + AUX)
      DELTE(IEL) = MIN(DELTE(IEL),DTL)
      IF(IER.LE.0)  GO TO 260
      DELTE(IER) = MIN(DELTE(IER),DTL)
  260 CONTINUE
C
C     COMPUTE  [R]  MATRIX :
C
      R(1,1) =  ALP*GAM1 - CI2
      R(1,2) = -GAM1*UI
      R(1,3) = -GAM1*VI
      R(1,4) =  GAM1
      R(2,1) = -VCAP
      R(2,2) = -RNY
      R(2,3) =  RNX
      R(2,4) =  0.
      R(3,1) =  ALP*GAM1 - UCAP*CI
      R(3,2) =  CX - GAM1*UI
      R(3,3) =  CY - GAM1*VI
      R(3,4) =  GAM1
      R(4,1) =  ALP*GAM1 + UCAP*CI
      R(4,2) = -CX - GAM1*UI
      R(4,3) = -CY - GAM1*VI
      R(4,4) =  GAM1
C
C     COMPUTE  [R]  MATRIX INVERSE :
C
      RI(1,1) = -1./CI2
      RI(1,2) =  0.
      RI(1,3) =  0.5/CI2
      RI(1,4) =  0.5/CI2
      RI(2,1) = -UI/CI2
      RI(2,2) = -RNY
      RI(2,3) =  (UI+CX)/(2.*CI2)
      RI(2,4) =  (UI-CX)/(2.*CI2)
      RI(3,1) = -VI/CI2
      RI(3,2) =  RNX
      RI(3,3) =  (VI+CY)/(2.*CI2)
      RI(3,4) =  (VI-CY)/(2.*CI2)
      RI(4,1) = -ALP/CI2
      RI(4,2) =  VCAP
      RI(4,3) =  (ALP+UCAP*CI)/(2.*CI2) + 1./(2.*GAM1)
      RI(4,4) =  (ALP-UCAP*CI)/(2.*CI2) + 1./(2.*GAM1)
C
C     COMPUTE  [AS]  =  [RI] [EIG] [R] :
C
      DO 300  I=1,4
      DO 300  J=1,4
      R(I,J) = RLAM(I)*R(I,J)
  300 CONTINUE
C
      DO 310  I=1,4
      DO 310  J=1,4
      AVROE(I,J) = 0.
      DO 310  L=1,4
      AVROE(I,J) = AVROE(I,J) + RI(I,L)*R(L,J)
  310 CONTINUE
C
C     COMPUTE THE DIFFERENCE OF THE CONSERVATION VARIABLES
C     BETWEEN THE RIGHT AND THE LEFT ELEMENTS :
C
      DU(1) = RHOR - RHOL
      DU(2) = RHOR*UXR - RHOL*UXL
      DU(3) = RHOR*VYR - RHOL*VYL
      DU(4) = RHOR*TER - RHOL*TEL
C
C     COMPUTE  DISS  =  [AS] UR-UL :
C
      DO 320  I=1,4
      DISS(I) = 0.
      DO 320  J=1,4
      DISS(I) = DISS(I) + AVROE(I,J)*DU(J)
  320 CONTINUE
C
C     COMPUTE SUM OF THE LEFT AND THE RIGHT FLUXES :
C
      FSUM(1) = RHOL*UNL + RHOR*UNR
```

```
      FSUM(2) = RNX*(PRESL+PRESR)
     &        + RHOL*UXL*UNL + RHOR*UXR*UNR
      FSUM(3) = RNY*(PRESL+PRESR)
     &        + RHOL*VYL*UNL + RHOR*VYR*UNR
      FSUM(4) = (RHOL*TEL + PRESL)*UNL
     &        + (RHOR*TER + PRESR)*UNR
C
C     THE INVISCID FLUX ON  RHS  OF THE EQ. IS :
C
      DO 330  I=1,4
      FLUX(I) = 0.5*(FSUM(I) - DISS(I))
  330 CONTINUE
C
C     CONTRIBUTION OF THIS FLUX TO THE "LEFT" ELEMENT:
C
      DO 340  I=1,4
      RHS0(IEL,I) = RHS0(IEL,I) - RLEN*FLUX(I)
  340 CONTINUE
C
C     CONTRIBUTION OF THIS FLUX TO THE "RIGHT" ELEMENT:
C
      IF(IER.LT.0)  GO TO 345
      DO 350  I=1,4
      RHS0(IER,I) = RHS0(IER,I) + RLEN*FLUX(I)
  350 CONTINUE
  345 CONTINUE
C
C     END LOOP OVER ALL THE SIDES
C
 1000 CONTINUE
C
C     SOLVE FOR NODAL INCREMENT AND UPDATE CONSERVATION VARIABLES:
C
      DO 1100  IE=1,NELEM
      DO 1100  IA=1,NAMAT
      UNKNO(IE,IA) = UNKN1(IE,IA) + DELTE(IE)*RHS0(IE,IA)/AREA(IE)
 1100 CONTINUE
C
      RETURN
      END
C*--------------------------------------------------------------------*
C*   [SUB. OUTPUT] CONVERT ELEMENT QUANTITIES TO NODAL QUANTITIES      *
C*                 AND PRINT OUT SOLUTIONS.                           *
C*--------------------------------------------------------------------*
      SUBROUTINE OUTPUT(NNODE, NAMAT, NBOUN, NPOIN, NELEM, BSIDO,
     *                  UNKNO, UNKNP, UNKNP1,COORD, INTMA, IDUM1,
     *                  GAMMA)
C
      IMPLICIT  REAL*8(A-H,O-Z)
C
      DIMENSION  UNKNO(NELEM,NAMAT), UNKNP(NPOIN,NAMAT)
      DIMENSION  UNKNP1(NPOIN,NAMAT)
      DIMENSION  COORD(NPOIN,2), INTMA(NELEM,NNODE)
C
      INTEGER  BSIDO(NBOUN,4), IDUM1(NPOIN)
C
C     CONVERT ELEMENT QUANTITIES TO NODAL QUANTITIES.
C
      DO 50 IA=1,NAMAT
      DO 50 IP=1,NPOIN
      UNKNP(IP,IA) = 0.
   50 CONTINUE
C
      DO 70  IP=1,NPOIN
      IDUM1(IP) = 0
   70 CONTINUE
C
      DO 100  IE=1,NELEM
      DO 101  IN=1,NNODE
      IP = INTMA(IE,IN)
      IF(IP.EQ.0)  GO TO 103
      IDUM1(IP) = IDUM1(IP) + 1
      DO 102  IA=1,NAMAT
      UNKNP(IP,IA) = UNKNP(IP,IA) + UNKNO(IE,IA)
  102 CONTINUE
  101 CONTINUE
  103 CONTINUE
```

```
  100 CONTINUE
C
      DO 105  IA=1,NAMAT
      DO 105  IP=1,NPOIN
      UNKNP(IP,IA) = UNKNP(IP,IA)/FLOAT(IDUM1(IP))
  105 CONTINUE
C
C     TRANSFORM CONSERVATION VARIABLES BACK TO PRIMATIVE VARIABLES.
C
      DO 110  IP=1,NPOIN
      DO 110  IA=2,NAMAT
      UNKNP(IP,IA) = UNKNP(IP,IA)/UNKNP(IP,1)
  110 CONTINUE
C
C     CONSTRAINT SOME NODAL QUANTITIES TO INLET BOUNDARY CONDITIONS.
C
      DO 120  IB=1,NBOUN
      IBC = BSIDO(IB,4)
      IF(IBC.NE.1)  GO TO 120
      II = BSIDO(IB,1)
      JJ = BSIDO(IB,2)
      DO 125  IA=1,NAMAT
      UNKNP(II,IA) = UNKNP1(II,IA)
      UNKNP(JJ,IA) = UNKNP1(JJ,IA)
  125 CONTINUE
  120 CONTINUE
C
C     PRINT OUT SOLUTIONS : *.SO_ = SOLUTION FILE
C                           *.RN_ = ADAPTIVE FILE
C
      NN1 = 1
      NN2 = 1
      WRITE(26,131)
  131 FORMAT(3X,'NPOIN',4X,'NELEM',4X,'N1',6X,'N2')
C
C     SEARCH NO. OF INVISCID WALLS.
C
      DO 140 IW=1,NBOUN
      IBC = BSIDO(IW,4)
      IF(IBC.EQ.3) NN2 = NN2 + 1
  140 CONTINUE
C
      WRITE(26,151) NPOIN, NELEM, NN1, NN2
  151 FORMAT(4I8)
C
      WRITE(7,161)
  161 FORMAT(4X, 4HNODE, 9X, 3HRHO, 14X, 1HU, 15X, 1HV, 15X, 1HE)
C
      DO 170  IP=1,NPOIN
      WRITE(7,173)  IP, (UNKNP(IP,IA), IA=1,NAMAT)
  173 FORMAT(I8, 4E16.8)
C
      WRITE(26,174) IP, (UNKNP(IP,IA), IA=1,NAMAT),(COORD(IP,ID),ID=1,2)
  174 FORMAT(I8, 6E16.8)
  170 CONTINUE
C
      DO 180  IS=1,NELEM
      WRITE(26,183) IS, (INTMA(IS,J), J=1,3)
  183 FORMAT(4I5)
  180 CONTINUE
C
C     PRINT OUT SOLUTIONS : *.JU_ = FEPLOT FILE
C
      WRITE(23,193)
  193 FORMAT('   NPOIN   NELEM    NVAR')
C
      WRITE(23,194)  NPOIN, NELEM, NAMAT
  194 FORMAT(3I8)
C
      WRITE(23,196)  NPOIN
  196 FORMAT(' NODAL COORDINATES & SOLUTIONS [', I5, ']')
C
      DO 210  IP=1,NPOIN
      WRITE(23,214) IP,(COORD(IP,J),J=1,2),(UNKNP(IP,IA), IA=1,NAMAT)
  214 FORMAT(I6, 6E13.5)
  210 CONTINUE
C
```

```
      WRITE(23,217)  NELEM
  217 FORMAT(' ELEMENT NODAL CONNECTIONS [', I5, ']')
C
      DO 220  IE=1,NELEM
      WRITE(23,228) IE, (INTMA(IE,J), J=1,3)
  228 FORMAT(4I8)
  220 CONTINUE
C
C     PRINT OUT SOLUTIONS : *.X_ = NASTRAN OUTPUT FILE
C
      WRITE(24,242)
  242 FORMAT('1 MSC/NASTRAN PAGE')
      WRITE(24,244)
  244 FORMAT('0')
      WRITE(24,246)
  246 FORMAT(' D I S P L A C E M E N T')
      WRITE(24,248)
  248 FORMAT('      POINT ID. TYPE',4X,'U',12X,'V',12X,'0',12X,'P',12X,
     +       'E',12X,'PR')
      DO 260  IN=1,NPOIN
      ZS = 0.
      PS = UNKNP(IN,1)
      US = UNKNP(IN,2)
      VS = UNKNP(IN,3)
      TES = UNKNP(IN,4)
      PRS = (GAMMA-1)*PS*(TES-0.5*(US*US+VS*VS))
      WRITE(24,263) IN, US, VS, ZS, PS, TES, PRS
  263 FORMAT(9X,I6,4X,'G',6E13.5)
  260 CONTINUE
      WRITE(24,264)
  264 FORMAT('1')
C
C     PREPARE DATA FOR INPUT SPACE.FOR:
C
C     WRITE(25,2068) NPOIN
C2068 FORMAT(I5)
C     DO 207  IN=1,NPOIN
C     WRITE(25,2083) IN,UNKNP(IN,1)
C 207 CONTINUE
C2083 FORMAT(9X,I5,4X,E13.5)
C
      RETURN
      END
C*--------------------------------------------------------------------*
```

# ภาคผนวก ข

## รายละเอียดของโปรแกรม REMESH

โปรแกรม REMESH จะมีรายละเอียดเริ่มจากโปรแกรมหลักและตามด้วยโปรแกรมย่อยต่างๆ ทั้งหมดดังนี้

```
C     PROGRAM REMESH
C
C     A FINITE ELEMENT COMPUTER PROGRAM FOR GENERATING INITIAL AND ADAPTIVE
C      MESH
C
C     THE VALUES DECLARED IN THE PARAMETER STATEMENT BELOW SHOULD BE
C     SET ACCORDING TO THE SIZE OF THE PROBLEMS :
C        MXPOI = MAXIMUM NUMBER OF NODES IN THE MODEL.
C        MXELE = MAXIMUM NUMBER OF ELEMENTS IN THE MODEL.
C        MXBOU = MAXIMUM NUMBER OF BOUNDARIES IN THE MODEL.
C
      PROGRAM MESH
      REAL MMATG
C
C     PARAMETER (MXPOI=20000, MXELE=30000, MXBOU=10000)
C
       COMMON/ASTORE/COORG(2,5000),UNKNG(4,5000),DELTA(4,5000),
     1 GEOMG(7,10000),MMATG(5000),DERIP(2,5000),DERI2(4,5000),
     2 HELP(4,5000),DERIA(4,5000),STREC(4,5000),COORN(2,5000),
     3 COOR(2,5000),COORO(2,5000),RCOND(2,700),TEXT(20),AR(3),
     4 IELEG(3,10000),NPFRONT(10,2000),NQFRONT(10,2000),IEL(3,10000),
     5 LCOOR(5000),LCORE(5000),LCOID(5000),ISIDE(4,30000),LBOUD(700),
     6 INTMEG(3,10000),LWHER(5000),LHOWM(5000),ICONE(30000),LPOSI(5000),
     7 NBNO(30,500),NREGI(10,2000),LBOU(2,30),ICOND(30),NNN(30),
     8 IBCS(10,30),MBCS(10),NONF(10),NONR(10)
C
      CHARACTER FILNAM*4, CV*2
C
C
C     WRITE OUTPUT FILE FOR ANSYS
C
      WRITE(6,'(/,A,$)') ' PLEASE TYPE INPUT FILE NAME : '
      READ(5,'(A)') FILNAM
      WRITE(6,'(A,$)')   ' PLEASE TYPE CURRENT VERSION : '
      READ(5,'(A)') CV
C
C     UNIT  7 ==> INPUT DATA FROM SCREEN
C     UNIT  8 ==> OUTPUT FILE
C     UNIT  9 ==> REMESH DATA FILE
C     UNIT 10 ==> OUTPUT FILE FOR FINITE
C     UNIT 17 ==> FIXED INPUT FILE
C
      OPEN(UNIT= 7,FILE=FILNAM//'.G'//CV,STATUS='OLD')
      OPEN(UNIT= 8,FILE=FILNAM//'.KK'//CV,STATUS='UNKNOWN')
      OPEN(UNIT= 9,FILE=FILNAM//'.RE'//CV,STATUS='OLD')
      OPEN(UNIT=10,FILE=FILNAM//'.IN'//CV,STATUS='UNKNOWN')
      OPEN(UNIT=17,FILE=FILNAM//'.FIX',STATUS='OLD')
      OPEN(UNIT=19,FILE=FILNAM//'.SHO',STATUS='UNKNOWN')
C
C     UNIT 11 ==> INPUT ANSYS FILE:NODE
C     UNIT 12 ==> INPUT ANSYS FILE:ELEMENT
C     UNIT 13 ==> INPUT NASTRAN FILE
C
C     OPEN(11,FILE=FILNAM//'.A'//CV,STATUS='UNKNOWN')
C     OPEN(12,FILE=FILNAM//'.B'//CV,STATUS='UNKNOWN')
```

```
      OPEN(13,FILE=FILNAM//'.D'//CV,STATUS='UNKNOWN')
C
      IGRAPH = 0
      ILAST=1
      CALL RFILLIV(LCOOR,5000,0)
C
C *** READ THE DATA CORRESPONDING TO THE BACKGROUND MESH
C
      WRITE (6,*) ' *** READING DATA'
      WRITE(19,*) ' *** READING DATA'
C
C *** CONTROL
C
      READ(9,1) TEXT
      READ(9,*) NPOIG,NELEG,N1BODY,N2BODY
C
C *** READ THE BACGROUND GRID DATA
C
      CALL INPUT(NELEG,NPOIG,IELEG,COORG,DELTA,UNKNG)
C
C *** ADD ADITIONAL TRIANGLE IN ORDER TO MAKE THE DOMAIN COVERED
C     BY THE BACKGROUND GRID CONVEX
C
      WRITE (6,*) ' *** FILLING IN THE GAPS'
      WRITE(19,*) ' *** FILLING IN THE GAPS'
      CALL CONVEX(NELEG,NPOIG,IELEG,COORG,NPFRONT,NQFRONT,LCOOR,
     1            NREGI)
C
C *** FILL IN ISIDE
C
      WRITE (6,*) ' *** FILLING ISIDE FOR THE BACKGROUND GRID'
      WRITE(19,*) ' *** FILLING ISIDE FOR THE BACKGROUND GRID'
      CALL SIDE(NELEG,NPOIG,NSIDG,IELEG,ISIDE,LWHER,LHOWM,ICONE)
C
C *** FILL IN INTMEG
C
      WRITE (6,*) ' *** FILLING ELEMENT CONECTIVITY MATRIX'
      WRITE(19,*) ' *** FILLING ELEMENT CONECTIVITY MATRIX'
      CALL ELECON(NSIDG,IELEG,ISIDE,INTMEG)
C
      XMAX=-1.E+6
      XMIN=1.E+6
      YMAX=-1.E+6
      YMIN=1.E+6
C
C *** READ BASIC DATA CORRESPONDING TO THE NEW MESH
C
      READ(17,1) TEXT
      READ(17,*) NREG,NFN,NBCS
C
C *** SPECIFY COORDINATES OF FIXED NODES
C
      READ(17,1) TEXT
      DO 10 IN=1,NFN
      READ(17,*) IP,(COORN(J,IP),J=1,2)
C
C *** FIND OUT MAX AND MIN COORDINATES FOR PLOTTING
C
      XL=COORN(1,IP)
      YL=COORN(2,IP)
      IF(XL.LT.XMIN) XMIN=XL
      IF(YL.LT.YMIN) YMIN=YL
      IF(XL.GT.XMAX) XMAX=XL
      IF(YL.GT.YMAX) YMAX=YL
   10 CONTINUE
C
      INEW=0
C
C *** EXPAND THE BACKGROUND MESH TO INCLUDE ALL THE DOMAIN
C
      CALL EXPAND(NELEG,NPOIG,COORG,COOR0,IELEG,INTMEG,N1BODY,N2BODY)
C
      IBACK=0
      IPOII=0
      IPOIP=0
C
      NODE=0
```

```
C
C *** SPECIFY BOUNDARY SEGMENTS
C
      WRITE (6,*) ' *** GENERATING BOUNDARY NODES '
      WRITE(19,*) ' *** GENERATING BOUNDARY NODES '
C
      NBOU=0
      READ(17,1) TEXT
      DO 20 IBS=1,NBCS
C
C *** GENERATE BOUNDARY NODES
C
      CALL INTERP(NPOIG,NELEG,NODE,COORG,IELEG,INTMEG,DELTA,COOR,
     1            COORN,NBNO,NNN,LCOOR,LBOU,NBOU,IBS,ILAST,IPOII,
     2            IPOIP,LBOUD,RCOND,IGRAPH)
C
   20 CONTINUE
C
      NBOUN=NODE
C
C *** NOW TAKE CARE OF THE COMMON POINTS
C
      DO 30 IBS=1,NBCS
      DO 34 I=1,2
      J=1
      IF(I.EQ.2) J=NNN(IBS)
      KPOI=NBNO(IBS,J)
      DO 32 IBOU=1,NBOU
      KNEW=LBOU(2,IBOU)
      KTRY=LBOU(1,IBOU)
      IF(KTRY.EQ.KPOI) GOTO 33
   32 CONTINUE
   33 NBNO(IBS,J)=KNEW
   34 CONTINUE
   30 CONTINUE
C
C *** SPECIFY REGION BOUNDARY SEGMENTS
C
      READ(17,1)TEXT
C
      DO 40 IREG=1,NREG
      READ(17,*)IP,MBCS(IREG)
      READ(17,*)(IBCS(IREG,JB),JB=1,MBCS(IREG))
   40 CONTINUE
C
C     ENQUIRE IF GENERATED NODES ARE REQUIRED
C
  150 CONTINUE
C
C     SET UP STREC FOR THE ALREADY EXISTING NODES
C
      WRITE (6,*) ' *** INTERPOLATING FROM THE BACKGROUND GRID'
      WRITE(19,*) ' *** INTERPOLATING FROM THE BACKGROUND GRID'
C
      DO 1075 I=1,NODE
      X=COOR(1,I)
      Y=COOR(2,I)
C
      INORM=1
      CALL FINDEL(NPOIG,NELEG,COORG,IELEG,INTMEG,X,Y,ILAST,
     1            AR,I1,I2,I3,IENR)
      CALL GETVALUE(NPOIG,I1,I2,I3,AR,DELTA,DIS,ALP,ANX,ANY,
     1              INORM)
C
      STREC(1,I)=DIS
      STREC(2,I)=ALP
      STREC(3,I)=ANX
      STREC(4,I)=ANY
 1075 CONTINUE
C     WRITE (9,20001)
20001 FORMAT(' STREC',/)
20002 FORMAT(I8,2X,1P6E15.7)
C
C     SET UP INITIAL FRONTS
C
      CALL SETUP(NREG,MBCS,IBCS,NBNO,NNN,COOR,NODE,
     1           NPFRONT,NQFRONT,NONF,NREGI,NONR)
```

```
C
C     TRIANGULATE REGIONS
C
      CALL TRIANGLE(NREG,NONF,NPFRONT,NQFRONT,NREGI,NONR,IEL,
     1              COOR,NELEM,NODE,NELEG,NPOIG,IELEG,INTMEG,
     2              COORG,DELTA,STREC,TOLER,ILAST,IGRAPH)
C
C     PRINT FULL NODAL INFORMATION
C
C     WRITE(6,7)
C     DO 140 KN=1,NODE
C     WRITE(6,3) KN,(COOR(JN,KN),JN=1,2)
C 140 CONTINUE
C
C     FIND OUT BOUNDARY POINTS
C
      CALL BOUNDAR(NODE,NELEM,LCOOR,IEL)
C
C     FIND OUT REAL AND IDEAL NUMBER OF CONECTIVITIES
C
      CALL CONERE(NODE,NELEM,IEL,LCORE)
      CALL CONEID(NODE,NELEM,IEL,LCOOR,LCOID,COOR,STREC)
C
C     FILL IN ISIDE
C
      WRITE (6,*) ' *** FILLING ISIDE FOR THE GENERATED MESH'
      WRITE(19,*) ' *** FILLING ISIDE FOR THE GENERATED MESH'
      CALL SIDE(NELEM,NODE,NSIDE,IEL,ISIDE,LWHER,LHOWM,ICONE)
C
19999 CONTINUE
      WRITE (6,*) ' *** WHAT SHALL WE DO NOW ?'
      WRITE (6,*) '     0 - QUIT'
      WRITE (6,*) '     1 - PLOT THE MESH'
      WRITE (6,*) '     2 - SMOOTH THE MESH'
      WRITE (6,*) '     3 - SWAP DIAGONALS'
      WRITE (6,*) '     4 - EAT 3 S '
      WRITE (6,*) '     5 - AREA CHECK/OUTPUT NO OF NODES AND ELEMENTS'
      WRITE (6,*) '     6 - GET THE RE-START FILE'
C
      WRITE(19,*) ' *** WHAT SHALL WE DO NOW ?'
      WRITE(19,*) '     0 - QUIT'
      WRITE(19,*) '     1 - PLOT THE MESH'
      WRITE(19,*) '     2 - SMOOTH THE MESH'
      WRITE(19,*) '     3 - SWAP DIAGONALS'
      WRITE(19,*) '     4 - EAT 3 S '
      WRITE(19,*) '     5 - AREA CHECK/OUTPUT NO OF NODES AND ELEMENTS'
      WRITE(19,*) '     6 - GET THE RE-START FILE'
      READ (7,*)  IWHAT
      WRITE(6,*)  IWHAT
      WRITE(19,*)  IWHAT
      IF(IWHAT.GE.7) GOTO 19999
      IF(IWHAT.EQ.0) GOTO 20000
      GOTO(30000,40000,50000,60000,70000,80000) IWHAT
C
C     PLOT GENERATED MESH IF REQUIRED
C
30000 CONTINUE
      GOTO 19999
C
C     SMOOTH MESH IF NECESSARY
C
40000 CONTINUE
      WRITE (6,*) ' ENTER NUMBER OF SMOOTHING LOOPS'
      WRITE(19,*) ' ENTER NUMBER OF SMOOTHING LOOPS'
      READ (7,*) NSMOO
      WRITE(6,*) NSMOO
      WRITE(19,*) NSMOO
      CALL SMOOTH(2,3,NELEM,NODE,NSMOO,LCOOR,LCORE,IEL,COOR,COOR0)
      GOTO 19999
C
C     SWAPPING DIAGONALS
C
50000 CONTINUE
      CALL SWAPDI(NODE,NELEM,NSIDE,ISIDE,IEL,LCORE,LCOID,LCOOR,COOR,
     &            LWHER,LHOWM,ICONE)
      GOTO 19999
C
```

```
C     EAT 3'S
C
60000 CONTINUE
      CALL EAT3(NODE,NELEM,NSIDE,IEL,ISIDE,LCOOR,LPOSI,
     &          LWHER,LHOWM,LCORE,LCOID,ICONE,COOR)
      GOTO 19999
C
C     AREA CHECKING
C
70000 CONTINUE
      CALL AREACH(NODE,NELEM,IEL,COOR)
      GOTO 19999
C
C     RE-START FILE
C
80000 CALL OUTPUT(NPOIG,NELEG,COORG,IELEG,INTMEG,UNKNG,NODE,
     1            NELEM,COOR,IEL,RCOND,NBOUN,LBOUD,ISIDE,NSIDE,
     2            TOLER,ILAST)
      GOTO 19999
C
20000 CONTINUE
C     WRITE (9,20002) (IP,(COOR(IJ,IP),IJ=1,2),(STREC(IJ,IP),
C    .                 IJ=1,4),IP=1,NODE)
C
    1 FORMAT(20A4)
C   3 FORMAT(I5,1P6E15.7)
C  31 FORMAT(I5,1P2E15.7,4I5)
C   5 FORMAT(' INITIAL BOUNDARY/FIXED NODES')
C   6 FORMAT((16I5))
C   7 FORMAT(' INPUT AND GENERATED NODES')
C   8 FORMAT(' DO YOU WANT TO SEE GENERATED NODES? 1 YES 0 NO')
    9 FORMAT(' DO YOU WANT TO SEE INITIAL DATA? 1 YES 0 NO')
 2010 FORMAT(' DO YOU WANT TO NUMBER NODES? 1 YES 0 NO')
C
      STOP
      END
C
C     ------------------------------------------------
C
      SUBROUTINE INPUT(NELEG,NPOIG,IELEG,COORG,DELTA,UNKNG)
C
      REAL TEXT(20)
      REAL COORG(2,NPOIG),DELTA(4,NPOIG),UNKNG(4,NPOIG)
C
      INTEGER IELEG(3,NELEG)
C
      DO 117 IN=1,NPOIG
      READ(9,*) J,(UNKNG(I,J),I=1,4),(COORG(I,J),I=1,2)
  117 CONTINUE
C
C *** CONNECTIVITIES
C
      DO 116 IE=1,NELEG
      READ(9,*) J,(IELEG(I,J),I=1,3)
  116 CONTINUE
C
C **** GET THE VALUE OF THE REFINEMENT INDICATORS
C
      WRITE(6,'(A,$)')
     *'  GENERATING AN INITIAL GRID(0) OR REMESHING(1)? '
      WRITE(19,'(A,$)')
     *'  GENERATING AN INITIAL GRID(0) OR REMESHING(1)? '
      READ (7,*)  IO
      WRITE(6,*)  IO
      WRITE(19,*)  IO
C
      IF(IO.EQ.1) THEN
      CALL REF2(NPOIG,NELEG)
      ELSE
      DO 1006 IP=1,NPOIG
      DELTA(1,IP)=UNKNG(1,IP)
      DELTA(2,IP)=UNKNG(2,IP)
      DELTA(3,IP)=UNKNG(3,IP)
      DELTA(4,IP)=UNKNG(4,IP)
 1006 CONTINUE
C
C *** READ THE VALUE OF THE UNKNOWNS FOR INTEPOLATION
```

```
C
      READ(9,1) TEXT
      DO 1007 IP=1,NPOIG
      READ(9,*) I,(UNKNG(IA,I),IA=1,4)
 1007 CONTINUE
C
      ENDIF
C
    1 FORMAT(20A4)
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE CONVEX(NELEM,NPOIN,INTMAT,COORD,NPFRONT,NQFRONT,
     1                  LCOOR,NREGI)
C
C *** THIS SUBROUTINE ADD TRIANGLES TO THE BACGROUND GRID IN ORDER
C     TO MAKE A CONVEX REGION
C
      REAL COORD(2,5000)
C
      INTEGER INTMAT(3,10000),LCOOR(5000)
      INTEGER NPFRONT(10,2000),NQFRONT(10,2000)
      INTEGER NREGI(10,2000)
C
C *** FIRST FIND OUT THE BOUNDARY POINTS
C
      CALL BOUNDAR(NPOIN,NELEM,LCOOR,INTMAT)
C
C *** STORE BOUNDARY POINTS
C
      NSIDE=0
      DO 1000 IP=1,NPOIN
      IF(LCOOR(IP).EQ.0) GOTO 1000
      NSIDE=NSIDE+1
      NREGI(1,NSIDE)=IP
 1000 CONTINUE
C
C *** NOW SET UP THE FRONT
C
C *** IDENTIFY A SIDE
C
      ISIDE=0
      IEL=0
 1999 IEL=IEL+1
      IHOW=0
      DO 2001 IN=1,3
      IP=INTMAT(IN,IEL)
      IF(LCOOR(IP).NE.0) IHOW=IHOW+1
 2001 CONTINUE
      IF(IHOW.GE.2) GOTO 2002
 2000 GOTO 1999
 2002 CONTINUE
C
C *** GET THE FIRST SIDE
C
      DO 3000 IN=1,3
      IP=INTMAT(IN,IEL)
      IF(LCOOR(IP).EQ.0) GOTO 3000
      IN1=IN+1
      IF(IN1.EQ.4) IN1=1
      IP1=INTMAT(IN1,IEL)
      IF(LCOOR(IP1).EQ.0) GOTO 3000
C
C *** CHECK FOR THE NEXT FOUR SIDES
C
      IP2=IP1+LCOOR(IP)
      IF(IP2.LE.0) GOTO 3000
      IF(LCOOR(IP2).EQ.0) GOTO 3000
      IP3=IP+LCOOR(IP2)
      IF(IP3.LE.0) GOTO 3000
      IF(LCOOR(IP3).EQ.0) GOTO 3000
      IP4=IP2+LCOOR(IP3)
      IF(IP4.LE.0) GOTO 3000
      IF(LCOOR(IP4).EQ.0) GOTO 3000
```

```
      IP5=IP3+LCOOR(IP4)
      IF(IP5.LE.0) GOTO 3000
      IF(LCOOR(IP5).EQ.0) GOTO 3000
C
      GOTO 3001
C
 3000 CONTINUE
      GOTO 1999
 3001 CONTINUE
      IOLD=IP1
C
C *** THE FIRST SIDE IS IP1-IP
C     WE CAN START FILLING THE FRONT
C
 4000 ISIDE=ISIDE+1
      NPFRONT(1,ISIDE)=IP1
      NQFRONT(1,ISIDE)=IP
      IKEEP=IP
      IF(LCOOR(IP).EQ.0) GOTO 4010
      IP=IP1+LCOOR(IP)
      IF(IP.LE.0) GOTO 4010
      GOTO 4011
 4010 WRITE (6,*) ' ERROR IN CONVEX'
      WRITE(19,*) ' ERROR IN CONVEX'
      STOP
 4011 CONTINUE
      LCOOR(IKEEP)=0
      IP1=IKEEP
      IF(IP1.NE.IOLD) GOTO 4000
C
C *** CHECK IF ISIDE.EQ.NSIDE
C
      IF(ISIDE.EQ.NSIDE) GOTO 4500
C
C *** THERE ARE MORE HOLES, FIND THEM
C
      GOTO 1999
 4500 CONTINUE
C
C *** LOOP OVER THE NUMBER OF SIDES
C
      NPBOU=NSIDE
      IPOSI=0
 6000 IPOSI=IPOSI+1
      KN1=NPFRONT(1,IPOSI)
      IF(KN1.EQ.0) GOTO 5900
      KN=NQFRONT(1,IPOSI)
      XN=COORD(1,KN)
      YN=COORD(2,KN)
      XN1=COORD(1,KN1)
      YN1=COORD(2,KN1)
      A=YN1-YN
      B=XN-XN1
      C=(XN1-XN)*YN1+(YN-YN1)*XN1
      INUM=0
      ANG1=0.0
      DO 6500 IS=1,NPBOU
      KEN=NREGI(1,IS)
      IF(KEN.EQ.KN.OR.KEN.EQ.KN1) GOTO 6500
      XKEN=COORD(1,KEN)
      YKEN=COORD(2,KEN)
      IF(A*XKEN+B*YKEN+C.LE.0.0) GOTO 6500
      XDIF1=XKEN-XN1
      YDIF1=YKEN-YN1
      XDIF2=XKEN-XN
      YDIF2=YKEN-YN
      DIST1=SQRT(XDIF1*XDIF1+YDIF1*YDIF1)
      DIST2=SQRT(XDIF2*XDIF2+YDIF2*YDIF2)
      COSA=(XDIF1*XDIF2+YDIF1*YDIF2)/(DIST1*DIST2)
      IF(COSA.GT.1.0) COSA=1.0
      IF(COSA.LT.-1.0) COSA=-1.0
      ANGL=ACOS(COSA)
      IF(ANGL.LT.0.1) GOTO 6500
C
C *** SEE IF IT IS A POSSIBLE CONECTIVITY
C
      CALL POSSIBLE(KN1,KN,KEN,XN1,YN1,XN,YN,XKEN,YKEN,1,NSIDE,
```

```
     1              NPBOU,NPFRONT,NQFRONT,NREGI,COORD,IYON)
C
      IF(IYON.EQ.0) GOTO 6500
      INUM=1
      IF(ANGL.LT.ANG1) GOTO 6500
      ANG1=ANGL
      KPO=KEN
 6500 CONTINUE
C
      IF(INUM.EQ.0) GOTO 5900
C
C *** CREATE A NEW ELEMENT
C
      NELEM=NELEM+1
      INTMAT(1,NELEM)=KN1
      INTMAT(2,NELEM)=KN
      INTMAT(3,NELEM)=KPO
C
C *** UPDATE THE FRONT
C
      NPFRONT(1,IPOSI)=0
      NQFRONT(1,IPOSI)=0
      IND=0
      DO 5700 IS=1,NSIDE
      IF(NPFRONT(1,IS).NE.KPO.OR.NQFRONT(1,IS).NE.KN1) GOTO 5700
      IND=1
      NPFRONT(1,IS)=0
      NQFRONT(1,IS)=0
      GOTO 5701
 5700 CONTINUE
 5701 CONTINUE
      IF(IND.EQ.1) GOTO 5790
      NSIDE=NSIDE+1
      NPFRONT(1,NSIDE)=KN1
      NQFRONT(1,NSIDE)=KPO
 5790 IND=0
      DO 5800 IS=1,NSIDE
      IF(NPFRONT(1,IS).NE.KN.OR.NQFRONT(1,IS).NE.KPO) GOTO 5800
      IND=1
      NPFRONT(1,IS)=0
      NQFRONT(1,IS)=0
      GOTO 5801
 5800 CONTINUE
 5801 CONTINUE
      IF(IND.EQ.1) GOTO 5900
      NSIDE=NSIDE+1
      NPFRONT(1,NSIDE)=KPO
      NQFRONT(1,NSIDE)=KN
C
 5900 IF(IPOSI.LT.NSIDE) GOTO 6000
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      SUBROUTINE EXPAND(NELEG,NPOIG,COORG,COOR0,IELEG,INTMEG,
                        N1BODY,N2BODY)
C
      REAL COORG(2,NPOIG),COOR0(2,NPOIG)
C
      INTEGER IELEG(3,NELEG),INTMEG(3,NELEG)
C
C *** THIS SUBROUTINE MODIFIES THE COORDINATES OF THE BACKGROUND
C     GRID BOUNDARY  POINTS  IN  ORDER  TO  COMPLETELY COVER THE
C     DOMAIN OF INTEREST
C
      DATA EXPA/0.5/
C
      WRITE (6,*) ' ENTER EXPANSION FACTOR'
      WRITE(19,*) ' ENTER EXPANSION FACTOR'
      READ (7,*)  EXPA
      WRITE(6,*)  EXPA
      WRITE(19,*)  EXPA
C
      CALL RFILLM(COOR0,2,NPOIG,0.0)
C
```

```
C *** LOOP OVER THE ELEMENTS
C
      AE=1.E+6
      DO 1000 IE=1,NELEG
         DO 1001 IN=1,3
         IF(INTMEG(IN,IE).NE.0) GOTO 1001
C
C *** MOVE NODES
C
         IN1=IN+1
         IN2=IN+2
         IF(IN1.GT.3) IN1=IN1-3
         IF(IN2.GT.3) IN2=IN2-3
         IP1=IELEG(IN1,IE)
         IP2=IELEG(IN2,IE)
         AX=COORG(2,IP2)-COORG(2,IP1)
         AY=COORG(1,IP1)-COORG(1,IP2)
         AT=SQRT(AX*AX+AY*AY)
         IF(AT.LT.AE) AE=AT
 1001    CONTINUE
 1000 CONTINUE
      WRITE (6,20100) AE
      WRITE(19,20100) AE
20100 FORMAT(' AE = ',1PE15.7)
C
      DO 2000 IE=1,NELEG
      EXP=EXPA*AE
      IF (IE.GE.N1BODY.AND.IE.LE.N2BODY) EXP=EXP*1.E-2
         DO 2001 IN=1,3
         IF(INTMEG(IN,IE).NE.0) GOTO 2001
C
C *** MOVE NODES
C
         IN1=IN+1
         IN2=IN+2
         IF(IN1.GT.3) IN1=IN1-3
         IF(IN2.GT.3) IN2=IN2-3
         IP1=IELEG(IN1,IE)
         IP2=IELEG(IN2,IE)
         AX=COORG(2,IP2)-COORG(2,IP1)
         AY=COORG(1,IP1)-COORG(1,IP2)
         AT=SQRT(AX*AX+AY*AY)
         AX=AX*EXP/AT
         AY=AY*EXP/AT
         COOR0(1,IP1)=COOR0(1,IP1)+AX
         COOR0(2,IP1)=COOR0(2,IP1)+AY
         COOR0(1,IP2)=COOR0(1,IP2)+AX
         COOR0(2,IP2)=COOR0(2,IP2)+AY
 2001    CONTINUE
 2000 CONTINUE
C
C *** ADD
C
      DO 3000 I=1,NPOIG
      DO 3001 J=1,2
      COORG(J,I)=COORG(J,I)+COOR0(J,I)
 3001 CONTINUE
 3000 CONTINUE
C
      RETURN
      END
C
C     ----------------------------------------------------------
C
      SUBROUTINE INTERP(NPOIG,NELEG,NPOIN,COORG,IELEG,INTMEG,
     1                  DELTA,COORD,COORN,NBNO,NNN,LCOOR,
     2                  LBOU,NBOU,IBS,ILAST,IPOII,IPOIP,
     3                  LBOUD,RCOND,IGRAPH)
C
      REAL COORG(2,NPOIG),COORD(2,5000),DELTA(4,NPOIG)
      REAL COORN(2,5000)
C
      INTEGER IELEG(3,NELEG),INTMEG(3,NELEG),LBOUD(700)
      INTEGER NBNO(30,500),NNN(30),LCOOR(NPOIG)
C
      REAL X(2,500),C(2,500),T(2,500),XX(2,500)
      REAL X0(2),XC(2),X1(2),TI(500),TS(500),TD(500)
```

```
      REAL TG(500),TL(500),TX(500),RCOND(2,700)
C
      INTEGER LN(500),LX(500),LS(500),LBOU(2,30)
C
C *** THIS SUBROUTINE IS  CALLED  ONCE  FOR EACH BOUNDARY SEGMENT
C     IT INTERPOLATES ADDITIONAL BOUNDARY POINTS ACCORDING TO THE
C     SPACING SPECIFIED BY THE BACKGROUND GRID
C
C *** READ THE DATA
C
      READ(17,*) IB,NN,IGEOM,ICOND
      READ(17,*) (LN(I),I=1,NN)
C
C *** STORE THE INTERSECTION POINTS
C
      IF(NBOU.EQ.0) THEN
        NBOU=2
        NPOIN=2
        LBOU(1,1)=LN(1)
        LBOU(1,2)=LN(NN)
        LBOU(2,1)=1
        LBOU(2,2)=2
        DO 108 J=1,2
        RCOND(J,1)=0.0
        RCOND(J,2)=0.0
        COORD(J,1)=COORN(J,LN(1))
        COORD(J,2)=COORN(J,LN(NN))
  108   CONTINUE
      ELSE
        KFIR=LN(1)
        KSEC=LN(NN)
        IFIR=1
        ISEC=1
        DO 100 IBOU=1,NBOU
        IF(LBOU(1,IBOU).EQ.KFIR) IFIR=0
        IF(LBOU(1,IBOU).EQ.KSEC) ISEC=0
  100   CONTINUE
        IF(IFIR.EQ.0) GOTO 110
        NBOU=NBOU+1
        NPOIN=NPOIN+1
        RCOND(1,NPOIN)=0.0
        RCOND(2,NPOIN)=0.0
        COORD(1,NPOIN)=COORN(1,KFIR)
        COORD(2,NPOIN)=COORN(2,KFIR)
        LBOU(1,NBOU)=KFIR
        LBOU(2,NBOU)=NPOIN
  110   IF(ISEC.EQ.0) GOTO 120
        NBOU=NBOU+1
        NPOIN=NPOIN+1
        RCOND(1,NPOIN)=0.0
        RCOND(2,NPOIN)=0.0
        COORD(1,NPOIN)=COORN(1,KSEC)
        COORD(2,NPOIN)=COORN(2,KSEC)
        LBOU(1,NBOU)=KSEC
        LBOU(2,NBOU)=NPOIN
  120   CONTINUE
      ENDIF
C
      CALL RFILLM(T,2,NN,0,0)
      CALL RFILLIV(LX,NN,0)
C
C *** TRANSFER
C
      DO 200 I=1,NN
      DO 200 J=1,2
      X(J,I)=COORN(J,LN(I))
  200 CONTINUE
C
C *** GET THE CONTROL POINTS
C
      CALL GETCNTRL(NN,X,C,LX,T,LS)
C
C *** PLOT THE CURVES
C
      DO 30 I=1,NN-1
      DO 31 J=1,2
      XO(J)=X(J  ,I)
```

```
      XC(J)=C(J  ,I)
   31 X1(J)=X(J,I+1)
      IF (IGRAPH.EQ.1) CALL PLOTSP(X0,XC,X1)
   30 CONTINUE
C
C *** FIND INTERSECTIONS WITH THE BACKGROUND GRID
C
      CALL FINDTIN(NPOIG,NELEG,IELEG,INTMEG,COORG,ILAST,NN,X,C,NI,
     1             TI,IPOII)
C
C *** FIND THE ARC LENGTH COORDINATE OF THE INTERSECTION POINTS
C
      CALL FINDTSN(NN,X,C,NI,TI,TS)
C
C *** FIND THE SPACING REQUIRED
C
      CALL FINDTDN(NN,X,C,NI,TI,TD,NPOIG,NELEG,IELEG,INTMEG,
     1             COORG,DELTA,ILAST,IGEOM)
C
C *** FIND THE LOCATION OF THE BOUNDARY POINTS
C
      CALL FINDTPO(NN,NI,NP,X,C,TS,TG,TD,TL,TX,XX)
C
C *** PLOT GENERATED POINT IF REQUIRED
C
      IF(IPOIP.EQ.0.OR.IGRAPH.NE.1) GOTO 8000
      DO 7000 IP=1,NP
 7000 CONTINUE
 8000 CONTINUE
C
      NNN(IBS)=NP
      NBNO(IBS,1)=LN(1)
      NBNO(IBS,NP)=LN(NN)
C
C *** NOW STORE ALL POINTS BUT THE END ONES
C
      IF(NP.EQ.2) GOTO 9000
      DO 9001 IP=2,NP-1
      NPOIN=NPOIN+1
      NBNO(IBS,IP)=NPOIN
      ANORX=XX(2,IP+1)-XX(2,IP-1)
      ANORY=XX(1,IP-1)-XX(1,IP+1)
      AMO=SQRT(ANORX**2+ANORY**2)
      RCOND(1,NPOIN)=ANORX/AMO
      RCOND(2,NPOIN)=ANORY/AMO
      DO 9002 J=1,2
      COORD(J,NPOIN)=XX(J,IP)
 9002 CONTINUE
      LBOUD(NPOIN)=ICOND
 9001 CONTINUE
 9000 CONTINUE
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE GETCNTRL(N,X,C,LX,T,LS)
C
C *** THIS SUBROUTINE EVALUATES THE CONTROL POINTS TO OBTAIN A
C     CUADRATIC BEZIERS FITTING.
C
      REAL X(2,N),C(2,N),T(2,N)
C
      INTEGER LX(N),LS(N)
C
      REAL A(2),T0(2),T1(2),X0(2),X1(2),XC(2)
C
      CALL RFILLIV(LS,N,0)
      IF(N.EQ.2) GOTO 6800
C
C *** IDENTIFY STRAIGHT LINE SEGMENTS
C
      DO 500 I=2,N-1
      DO 501 J=1,2
      T0(J)=X(J,I)-X(J,I-1)
  501 T1(J)=X(J,I+1)-X(J,I)
```

```
C
C *** SCALAR PRODUCT
C
      SCA=T0(1)*T1(1)+T0(2)*T1(2)
      IF(SCA.LE.0.0) GOTO 5000
C
C *** FIND MODULUS
C
      AM0=SQRT(T0(1)*T0(1)+T0(2)*T0(2))
      AM1=SQRT(T1(1)*T1(1)+T1(2)*T1(2))
      VECT=(T0(2)*T1(1)-T0(1)*T1(2))/(AM0*AM1)
C
      IF(ABS(VECT).LT.1.E-3) GOTO 500
      IF(VECT.GT.1.E-3) LS(I)=1
      IF(VECT.LT.-1.E-3) LS(I)=-1
  500 CONTINUE
      LS(1)=2
      LS(N)=2
C
      CALL RFILLIV(LX,N,0)
C
C *** LOOP OVER SEGMENTS
C
      IF(N.EQ.3) GOTO 601
      DO 600 I=2,N-2
C
C *** IF CHANGE IN CURVATURE MARK THEM
C
      ILO=LS(I)*LS(I+1)
      IF(ILO.LT.0) LX(I)=1
  600 CONTINUE
C
C *** CHECK NEIGHBOURS
C
      DO 700 I=2,N-2
      IF(LX(I).EQ.0) GOTO 700
      IF(LX(I-1).EQ.1.OR.LX(I+1).EQ.1) GOTO 5000
      IF(LS(I-1).EQ.0.OR.LS(I+1).EQ.0) GOTO 5000
  700 CONTINUE
  601 CONTINUE
C
C *** THE POINTS ARE ALL RIGHT !!!!
C
C *** GET FIRST THE TANGENTS PARALEL TO THE SIDES
C
      DO 680 I=1,N-1
      IF(LS(I).EQ.0) GOTO 680
      IF(LS(I+1).NE.0) GOTO 680
      LS(I)=0
      DO 690 J=1,2
  690 T(J,I)=X(J,I+1)-X(J,I)
  680 CONTINUE
C
      DO 681 K=1,N-1
      I=N+1-K
      IF(LS(I).EQ.0) GOTO 681
      IF(LS(I-1).NE.0) GOTO 681
      LS(I)=0
      DO 691 J=1,2
  691 T(J,I)=X(J,I)-X(J,I-1)
  681 CONTINUE
C
      DO 800 I=2,N-2
      IF(LX(I).EQ.0) GOTO 800
      DO 801 J=1,2
  801 A(J)=X(J,I+1)-X(J,I)
C
      DO 803 K=I,I+1
      LS(K)=0
      DO 802 J=1,2
      T(J,K)=A(J)
  802 CONTINUE
  803 CONTINUE
  800 CONTINUE
C
C *** FIND OUT THE POINT SLOPES
C
```

```
      DO 1000 I=1,N-1
C
      DO 1001 J=1,2
 1001 A(J)=X(J,I+1)-X(J,I)
C
C *** NOW ADD
C
      DO 1002 K=I,I+1
      IF(LS(K).EQ.0) GOTO 1002
      DO 1003 J=1,2
 1003 T(J,K)=T(J,K)+A(J)
 1002 CONTINUE
C
 1000 CONTINUE
C
C *** GET THE END SLOPES
C
C *** NO LINK BETWEEN 1 AND N ----> GET TANGENT VECTORS
C
      DO 3101 J=1,2
 3101 A(J)=X(J,2)-X(J,1)
C
      AMODU=SQRT(A(1)*A(1)+A(2)*A(2))
      DO 3102 J=1,2
 3102 A(J)=A(J)/AMODU
      SCA=0.0
      DO 3103 J=1,2
 3103 SCA=SCA+A(J)*T(J,2)
C
      DO 3104 J=1,2
 3104 T(J,1)=T(J,2)+2.*(SCA*A(J)-T(J,2))
C
      DO 3111 J=1,2
 3111 A(J)=X(J,N)-X(J,N-1)
C
      AMODU=SQRT(A(1)*A(1)+A(2)*A(2))
      DO 3112 J=1,2
 3112 A(J)=A(J)/AMODU
      SCA=0.0
      DO 3113 J=1,2
 3113 SCA=SCA+A(J)*T(J,N-1)
C
      DO 3114 J=1,2
 3114 T(J,N)=T(J,N-1)+2.*(SCA*A(J)-T(J,N-1))
C
C *** GET FINALLY CONTROL POINTS
C
 6800 DO 6000 I=1,N-1
C
      DO 6001 J=1,2
      X0(J)=X(J,I)
      X1(J)=X(J,I+1)
      T0(J)=T(J,I)
      T1(J)=T(J,I+1)
 6001 CONTINUE
C
C *** FIND INTERSECTION
C
      ILINE=0
      IF(LS(I).EQ.0.AND.LS(I+1).EQ.0) ILINE=1
      CALL INTERSEC(X0,X1,T0,T1,XC,ILINE)
C
      DO 6002 J=1,2
 6002 C(J,I)=XC(J)
C
 6000 CONTINUE
C
      RETURN
C
 5000 WRITE (6,*) ' *** MORE POINTS ARE NEEDED TO DEFINE THE BOUNDARY'
      WRITE(19,*) ' *** MORE POINTS ARE NEEDED TO DEFINE THE BOUNDARY'
      STOP
C
      END
C
C     ----------------------------------------------------------------
C
```

```
      SUBROUTINE INTERSEC(X0,X1,T0,T1,XC,ILINE)
C
C *** THIS SUBROUTINE FINDS THE INTERSECTION POINT XC BETWEEN THE
C     LINES X0->T0 AND X1->T1
C
      REAL X0(2),X1(2),T0(2),T1(2),XC(2)
C
      DENOM=T1(2)*T0(1)-T1(1)*T0(2)
      IF(ILINE.EQ.0) GOTO 1000
C
C *** THIS IS A STRAIGHT LINE
C
      DO 2000 J=1,2
 2000 XC(J)=0.5*(X0(J)+X1(J))
      RETURN
C
 1000 S=((X1(1)-X0(1))*T0(2)-(X1(2)-X0(2))*T0(1))/DENOM
      DO 3000 J=1,2
 3000 XC(J)=X1(J)+S*T1(J)
C
      RETURN
      END
C
C     ----------------------------------------------------------------
C
      SUBROUTINE FINDTIN(NPOIG,NELEG,IELEG,INTMEG,COORG,ILAST,
     1                   NN,X,C,NI,TI,IPOII)
C
C *** THIS SUBROUTINE FINDS OUT THE INTERSECTION POINTS OF A BEZIER
C     QUADRATIC CURVE WITH THE BACGROUND GRID MESH
C
      REAL COORG(2,NPOIG),X(2,NN),C(2,NN-1)
      REAL TI(500)
      REAL X0(2),XC(2),X1(2),XP(2),XQ(2),XT(2),AR(3),TPOS(3)
C
      INTEGER IELEG(3,NELEG),INTMEG(3,NELEG)
C
C *** FIRST POINT
C
      T=0.0
      NI=1
      TI(NI)=0.0
C
C *** LOOP OVER THE NUMBER OF LOCAL SEGMENTS
C
      DO 1000 I=1,NN-1
         TREF=REAL(I-1)
            DO 1001 J=1,2
            X0(J)=X(J,I)
            XC(J)=C(J,I)
            X1(J)=X(J,I+1)
 1001       CONTINUE
C
 2020       T=AMAX1((TI(NI)-TREF+0.0001),0.0)
      CALL GETXY(T,X0,XC,X1,XT)
      CALL FINDEL(NPOIG,NELEG,COORG,IELEG,INTMEG,XT(1),XT(2),ILAST,
     1            AR,I1,I2,I3,IENR)
C
            DO 1004 J=1,3
            J1=J+1
            J2=J+2
            IF(J1.GT.3) J1=J1-3
            IF(J2.GT.3) J2=J2-3
            IP=IELEG(J1,IENR)
            IQ=IELEG(J2,IENR)
               DO 1005 K=1,2
               XP(K)=COORG(K,IP)
               XQ(K)=COORG(K,IQ)
 1005          CONTINUE
            TMIN=AMAX1((TI(NI)-TREF+0.0001),0.0)
      CALL CROSS(X0,XC,X1,XP,XQ,TMIN,IND,T)
            IF(IND.EQ.0) THEN
            TPOS(J)=1.E+6
            ELSE
            TPOS(J)=TREF+T
            ENDIF
 1004       CONTINUE
```

```
C
C *** ANY POINTS ?
C
            TPMIN=AMIN1(TPOS(1),TPOS(2),TPOS(3))
            IF(TPMIN.GT.1.E+5) GOTO 990
C
C *** YES !!!
C
            NI=NI+1
            TI(NI)=TPMIN
      CALL GETXY(TPMIN-TREF,X0,XC,X1,XT)
            DO 1100 J=1,3
            IF(TPOS(J).LT.1.01*TPMIN) K=J
1100        CONTINUE
            ILAST=INTMEG(K,IENR)
            GOTO 2020
C
  990 CONTINUE
C
C *** ADD T=0.5 IF NECESSARY
C
      TCOMP=TREF+0.5
      DO 970 II=1,NI
      IF(ABS(TI(II)-TCOMP).LT.0.3) GOTO 1000
  970 CONTINUE
      NI=NI+1
      TI(NI)=TCOMP
      DO 980 II=1,NI
      IF(TI(NI-II).LT.TCOMP) GOTO 981
      TI(NI-II+1)=TI(NI-II)
      TI(NI-II)=TCOMP
  980 CONTINUE
  981 CONTINUE
      CALL GETXY(0.5,X0,XC,X1,XT)
C
 1000 CONTINUE
C
C *** ADD THE LAST POINT
C
      IF(TI(NI).GT.TREF+0.999) RETURN
      NI=NI+1
      TI(NI)=TREF+1
C
      RETURN
      END
C
C     ---------------------------------------------------------------
C
      SUBROUTINE CROSS(X0,XC,X1,XP,XQ,TMIN,IND,T)
C
      REAL X0(2),XC(2),X1(2),XP(2),XQ(2)
C
C *** THIS SUBROUTINE FINDS OUT  THE INTERSECTION  BETWEEN
C     A QUADRATIC BEZIER CURVE AND A STRAIGHT LINE SEGMENT
C
      IND=0
C
      AX=X0(1)-2.*XC(1)+X1(1)
      AY=X0(2)-2.*XC(2)+X1(2)
      BX=2.*(XC(1)-X0(1))
      BY=2.*(XC(2)-X0(2))
      CX=X0(1)
      CY=X0(2)
      DX=XP(1)
      DY=XP(2)
      EX=XQ(1)-XP(1)
      EY=XQ(2)-XP(2)
C
      A=AX*EY-AY*EX
      B=BX*EY-BY*EX
      C=(CX-DX)*EY-(CY-DY)*EX
C
      IF(ABS(A).GT.1.E-3) THEN
C
C *** THE DISCRIMINANT
C
            DIS=B*B-4.*A*C
```

```
            IF(DIS.LT.0.0) RETURN
            DIS=SQRT(DIS)
            I1=1
            I2=1
            ROOT1=(-B+DIS)/(2.*A)
            IF(ROOT1.GT.1.0.OR.ROOT1.LE.TMIN) I1=0
            ROOT2=(-B-DIS)/(2.*A)
            IF(ROOT2.GT.1.0.OR.ROOT2.LE.TMIN) I2=0
C
            IF(ABS(EX).GT.1.E-4) THEN
               S1=(AX*ROOT1*ROOT1+BX*ROOT1+CX-DX)/EX
               S2=(AX*ROOT2*ROOT2+BX*ROOT2+CX-DX)/EX
            ELSE
               S1=(AY*ROOT1*ROOT1+BY*ROOT1+CY-DY)/EY
               S2=(AY*ROOT2*ROOT2+BY*ROOT2+CY-DY)/EY
            ENDIF
            IF(S1.LT.0.0.OR.S1.GT.1.0) I1=0
            IF(S2.LT.0.0.OR.S2.GT.1.0) I2=0
C
            IF(I1.EQ.0.AND.I2.EQ.0) RETURN
            IND=1
            IF(I1.EQ.0) ROOT1=1.E+6
            IF(I2.EQ.0) ROOT2=1.E+6
            T=AMIN1(ROOT1,ROOT2)
C
      ELSE
C
C *** ONLY ONE ROOT AT MOST
C
            IF(ABS(B).LT.1.E-5) THEN
               IF(ABS(C).GT.1.E-8) RETURN
C
C *** WE ARE IN TROUBLE !!!!
C     THIS CASE IS VERY PECULIAR AND NEEDS MORE ELABORATION
C
               WRITE (6,*) ' TROUBLE IN CROSS !!!'
               WRITE(19,*) ' TROUBLE IN CROSS !!!'
               STOP
            ELSE
               I1=1
               ROOT1=-C/B
               IF(ROOT1.GT.1.0.OR.ROOT1.LT.TMIN) I1=0
               IF(ABS(EX).GT.1.E-3) THEN
                  S1=(BX*ROOT1+CX-DX)/EX
               ELSE
                  S1=(BY*ROOT1+CY-DY)/EY
               ENDIF
               IF(S1.LT.0.0.OR.S1.GT.1.0) I1=0
               IF(I1.EQ.0) RETURN
               IND=1
               T=ROOT1
            ENDIF
C
      ENDIF
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE FINDTSN(NN,X,C,NI,TI,TS)
C
      REAL X(2,NN),C(2,NN-1),TI(NI),TS(NI)
      REAL X0(2),XC(2),X1(2)
C
C *** THIS SUBROUTINE COMPUTES THE ARC LENGTH COORDINATE OF THE
C     INTERSECTION POINTS BETWEEN THE BOUNDARY SEGMENT AND  THE
C     BACGROUND GRID
C
      II=1
      TS(II)=0.0
      SREF=0.0
      DO 1000 I=1,NN-1
      TREF=REAL(I-1)
      TREF1=TREF+1.
      DO 1001 J=1,2
      X0(J)=X(J,I)
```

```
      XC(J)=C(J,I)
      X1(J)=X(J,I+1)
 1001 CONTINUE
 1020 IF(TI(II+1).GT.TREF1) GOTO 1030
      T=TI(II+1)-TREF
      CALL GETARCLN(T,X0,XC,X1,S)
      TS(II+1)=SREF+S
      II=II+1
      IF(II.LE.NI) GOTO 1020
 1030 CONTINUE
      CALL GETARCLN(1.0,X0,XC,X1,S)
      SREF=SREF+S
 1000 CONTINUE
C
      TS(NI)=SREF
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE GETARCLN(T,X0,XC,X1,S)
C
C *** THIS SUBROUTINE COMPUTES THE ARC LENGTH COORDINATE S
C     FOR A BEZIER QUADRATIC SPLINE GIVEN THE VALUE OF THE
C     T PARAMETER
C
      REAL X0(2),XC(2),X1(2),X(2),T,S
C
C *** GET AX,BX,AY AND BY
C
      AX=XC(1)-X0(1)
      BX=X0(1)-2.*XC(1)+X1(1)
      AY=XC(2)-X0(2)
      BY=X0(2)-2.*XC(2)+X1(2)
C
      A=BX*BX+BY*BY
      B=2.*(AX*BX+AY*BY)
      C=AX*AX+AY*AY
      D=B*B-4.*A*C
C
C *** IF D=0 THEN IT IS A STRAIGHT LINE
C
      IF(ABS(D).GT.(1.E-6*C*C)) GOTO 1000
      CALL GETXY(T,X0,XC,X1,X)
      S=SQRT((X0(1)-X(1))**2+(X0(2)-X(2))**2)
      RETURN
 1000 CONTINUE
C
C *** GET THE INTEGRATION CONSTANT
C
      CONST=(B*SQRT(C))/(4.*A)
      CONST=CONST-D*LOG(ABS(2.*SQRT(A*C)+B))/(8.*A*SQRT(A))
C
      SROOT=A*T*T+B*T+C
      AUX=2.*A*T+B
C
      S1=AUX*SQRT(SROOT)/(4.*A)
C
      S2=ABS(2.*SQRT(A*SROOT)+AUX)
      S2=LOG(S2)/SQRT(A)
      S2=S2*D/(8.*A)
C
C *** NOW GET S BY ADDING UP THE TWO CONTRIBUTIONS
C
      S=2.*(S1-S2-CONST)
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE GETXY(T,X0,XC,X1,X)
C
C *** THIS SUBROUTINE COMPUTES THE POINT COORDINATES X AND Y
C     FOR  BEZIER QUADRATIC SPLINE GIVEN THE VALUE OF THE  T
C     PARAMETER
```

```
C
      REAL X0(2),XC(2),X1(2),X(2),A1,A2,A3,T
C
      A1=(1.-T)**2
      A2=2.*T*(1.-T)
      A3=T**2
C
      DO 1000 I=1,2
      X(I)=A1*X0(I)+A2*XC(I)+A3*X1(I)
 1000 CONTINUE
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      SUBROUTINE GETTAN(T,X0,XC,X1,TTAN,AMO)
C
C *** THIS SUBROUTINE COMPUTES THE POINT TANGENT  COMPONENTS
C     FOR A BEZIER QUADRATIC SPLINE GIVEN THE VALUE OF THE T
C     PARAMETER
C
      REAL X0(2),XC(2),X1(2),TTAN(2),T
C
      DO 1000 I=1,2
      TTAN(I)=(X0(I)-2.*XC(I)+X1(I))*T+(XC(I)-X0(I))
 1000 CONTINUE
C
C *** NORMALIZE
C
      AMO=SQRT(TTAN(1)*TTAN(1)+TTAN(2)*TTAN(2))
      DO 2000 I=1,2
      TTAN(I)=TTAN(I)/AMO
 2000 CONTINUE
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      SUBROUTINE GETCUR(T,X0,XC,X1,CURV)
C
C *** THIS  SUBROUTINE  COMPUTES   THE   POINT   CURVATURE
C     FOR A BEZIER QUADRATIC SPLINE GIVEN THE VALUE OF THE
C     T PARAMETER
C
      REAL X0(2),XC(2),X1(2),TTAN(2),ANR(2),CURV,T
C
C *** GET THE TANGENT
C
      CALL GETTAN(T,X0,XC,X1,TTAN,AMO)
C
C *** THE NORMAL
C
      ANR(1)= TTAN(2)
      ANR(2)=-TTAN(1)
C
      CURV=0.0
      DO 1000 I=1,2
      CURV=CURV+(X0(I)-2.*XC(I)+X1(I))*ANR(I)
 1000 CONTINUE
C
      CURV=ABS(2.*CURV*AMO*AMO)
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      SUBROUTINE PLOTSP(X0,XC,X1)
C
C *** THIS SUBROUTINE PLOTS A QUADRATIC  BEZIER  SPLINE  AS  A
C     COLLECTION OF NPART STRAIGHT LINES
C
      REAL X0(2),XC(2),X1(2),X(2)
C
      DATA NPART/15/
```

```
C
      TINC=1./REAL(NPART)
      T=0.0
C
C *** INITIALIZE
C
      DO 100 IPART=1,NPART
      T=T+TINC
C
C *** GET X AND Y
C
      CALL GETXY(T,X0,XC,X1,X)
C
C
  100 CONTINUE
      RETURN
      END
C
C     ------------------------------------------------------
C
      SUBROUTINE FINDTDN(NN,X,C,NI,TI,TD,NPOIG,NELEG,IELEG,INTMEG,
     1                   COORG,DELTA,ILAST,IGEOM)
C
C     REAL COORG(2,NPOIG),DELTA(4,NPOIG)
      REAL COORG(2,NPOIG),DELTA(4,5000)
      REAL X(2,NN),C(2,NN-1),AR(3)
      REAL TI(NI),TD(NI),X0(2),XC(2),X1(2),XT(2),TTAN(2)
C
      INTEGER IELEG(3,NELEG),INTMEG(3,NELEG)
C
C     XTR(XQ,YQ,AX,AY,ALPH)=AX*ALPH*(AX*XQ+AY*YQ)+AY*(AY*XQ-AX*YQ)
C     YTR(XQ,YQ,AX,AY,ALPH)=AY*ALPH*(AX*XQ+AY*YQ)-AX*(AY*XQ-AX*YQ)
C     XBA(XQ,YQ,AX,AY,ALPH)=(AX*(AX*XQ+AY*YQ)/ALPH)+AY*(AY*XQ-AX*YQ)
C     YBA(XQ,YQ,AX,AY,ALPH)=(AY*(AX*XQ+AY*YQ)/ALPH)-AX*(AY*XQ-AX*YQ)
C
C
C *** THIS SUBROUTINE COMPUTES THE VALUE OF THE SPACING BETWEEN NODES ON
C     THE BOUNDARY SEGMENTS.  THE SPACING IS DIRECTLY INTERPOLATED  FROM
C     THE BACKGROUND GRID AND FACTORS SUCH AS  STRETCHING  AND GEOMETRY,
C     IF REQURED, ARE TAKEN INTO ACCOUNT.
C
C     (IGEOM: 1 FOLLOW IT ---> 10 POINTS PER RAD, IGEOM:0 IGNORE IT)
C
      DATA ATOLE/0.1/
C
C *** LOOP OVER NUMBER OF POINTS
C
      DO 1000 II=1,NI
      TOT=TI(II)-0.00001
      IN=INT(TOT)
      T=TOT-REAL(IN)+0.00001
      IN=IN+1
C
C *** TRANSFER
C
      DO 1001 J=1,2
      X0(J)=X(J,IN)
      XC(J)=C(J,IN)
      X1(J)=X(J,IN+1)
 1001 CONTINUE
C
C *** GET THE COORDINATES
C
      CALL GETXY(T,X0,XC,X1,XT)
C
C *** GET THE TANGENT DIRECTION
C
      CALL GETTAN(T,X0,XC,X1,TTAN,AMO)
C
C *** THE CURVATURE
C
      CURV=0.0
      IF(IGEOM.EQ.1) CALL GETCUR(T,X0,XC,X1,CURV)
C
C *** NOW INTERPOLATE THE SPACING
C
      INORM=1
```

```
      CALL FINDEL(NPOIG,NELEG,COORG,IELEG,INTMEG,XT(1),XT(2),ILAST,
     1            AR,I1,I2,I3,IENR)
      CALL GETVALUE(NPOIG,I1,I2,I3,AR,DELTA,DIS,ALP,ANX,ANY,INORM)
C
C
C *** FILL IN TD
C
C
C     XFI=XBA(TTAN(1),TTAN(2),ANX,ANY,ALP)
C     YFI=YBA(TTAN(1),TTAN(2),ANX,ANY,ALP)
      XFI=FXBA(TTAN(1),TTAN(2),ANX,ANY,ALP)
      YFI=FYBA(TTAN(1),TTAN(2),ANX,ANY,ALP)
      AFI=SQRT(XFI*XFI+YFI*YFI)
      XFI=XFI/AFI
      YFI=YFI/AFI
C     XRE=XTR(XFI,YFI,ANX,ANY,ALP)
C     YRE=YTR(XFI,YFI,ANX,ANY,ALP)
      XRE=FXTR(XFI,YFI,ANX,ANY,ALP)
      YRE=FYTR(XFI,YFI,ANX,ANY,ALP)
      RED=SQRT(XRE*XRE+YRE*YRE)
C
      DIS=DIS*RED
      IF(IGEOM.NE.1) GOTO 2000
      IF(CURV*DIS.LT.ATOLE) GOTO 2000
      DIS=ATOLE/CURV
C
 2000 TD(II)=1./DIS
C
 1000 CONTINUE
C
      RETURN
      END
C
C     ---------------------------------------------------------
C
      SUBROUTINE FINDTPO(NN,NI,NP,X,C,TS,TG,TD,TL,TX,XX)
C
      REAL X(2,NN),C(2,NN-1),TS(NI),TG(NI),TD(NI)
      REAL TL(500),TX(500),XX(2,500)
      REAL XO(2),XC(2),X1(2),XT(2)
C
C *** THIS SUBROUTINE COMPUTES THE  NUMBER AND THE T COORDINATE  OF  THE
C     GENERATED BOUNDARY POINTS.  THE PROCESS IS  AS FOLLOWS ; INTEGRATE
C     THE SPACING: TG ---> DETERMINE NUMBER OF POINTS: NP ---> DETERMINE
C     THEIR S COORDINATE: TL ---> DETERMINE THEIR T COORDINATE: TX  --->
C     DETERMINE THE X AND Y COORDINATES: XX.
C
      DATA NITER/15/
C
C *** INTEGRATE
C
      TG(1)=0.0
      DO 1000 II=2,NI
      SINT=TS(II)-TS(II-1)
      TG(II)=TG(II-1)+0.5*(TD(II-1)+TD(II))*SINT
 1000 CONTINUE
C
C *** FIND OUT NUMBER OF POINTS
C
      NP=INT(TG(NI))+1
      SP=TG(NI)/REAL(NP)
      NP=NP+1
C
C *** FIND THE S COORDINATE
C
      IP=1
      TL(IP)=0.0
      DO 2000 II=2,NI
 2001 TS1=REAL(IP)*SP
      IF(TG(II).LT.TS1) GOTO 2000
           SL=TS1-TG(II-1)
           A1=0.5*(TD(II)-TD(II-1))/(TS(II)-TS(II-1))
           SINT=0.5*(TS(II)-TS(II-1))
           STRY=SINT
           DO 5000 K=1,NITER
           SINT=0.5*SINT
           SRES=TD(II-1)*STRY+A1*STRY**2
```

```
            IF(SRES-SL) 5001,5001,5002
 5001       STRY=STRY+SINT
            GOTO 5000
 5002       STRY=STRY-SINT
 5000       CONTINUE
      IP=IP+1
      TL(IP)=TS(II-1)+STRY
      GOTO 2001
 2000 CONTINUE
C
      TL(NP)=TS(NI)
      TL(NP+1)=1.E+6
C
C *** NOW FIND THE T COORDINATE
C
      IP=1
      TX(IP)=0.0
      DO 5500 J=1,2
      XX(J,1)=X(J,1)
 5500 CONTINUE
      SRE1=0.0
      DO 6000 I=1,NN-1
            SREF=SRE1
            DO 6010 J=1,2
            X0(J)=X(J,I)
            XC(J)=C(J,I)
            X1(J)=X(J,I+1)
 6010       CONTINUE
            CALL GETARCLN(1.0,X0,XC,X1,SREIN)
            SRE1=SREF+SREIN
 6001       IF(TL(IP+1).GT.SRE1) GOTO 6000
            CL=TL(IP+1)-SREF
C
            TINT=0.5
            TTRY=TINT
            DO 8000 K=1,NITER
            TINT=0.5*TINT
            CALL GETARCLN(TTRY,X0,XC,X1,CRES)
C           IF(CRES-CL) 8001,8001,8002
C8001       TTRY=TTRY+TINT
C           GOTO 8000
C8002       TTRY=TTRY-TINT
            CTMP = CRES - CL
            IF(CTMP.LE.0.) TTRY=TTRY+TINT
            IF(CTMP.GT.0.) TTRY=TTRY-TINT
 8000       CONTINUE
      IP=IP+1
      TX(IP)=REAL(I-1)+TTRY
      CALL GETXY(TTRY,X0,XC,X1,XT)
      DO 6900 J=1,2
      XX(J,IP)=XT(J)
 6900 CONTINUE
      GOTO 6001
 6000 CONTINUE
C
      TX(NP)=REAL(NI-1)
      DO 9000 J=1,2
      XX(J,NP)=X(J,NN)
 9000 CONTINUE
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE SETUP(NREG,MBCS,IBCS,NBNO,NPOIN,COOR,NODE,
     1                 NPFRONT,NQFRONT,NONF,NREGI,NONR)
C
C        SET UP GENERATION FRONTS AND NODES - REGION BY REGION
C
      DIMENSION COOR(2,NODE)
      DIMENSION NBNO(30,500),NPOIN(30)
      DIMENSION IBCS(10,30),MBCS(10)
      DIMENSION NONF(10),NREGI(10,2000),NONR(10)
      DIMENSION NPFRONT(10,2000),NQFRONT(10,2000)
C
C     DIMENSION NODEL(5000)
```

```
C
C
C        LOOP OVER REGIONS
C
      DO 100 IREG=1,NREG
C
C
C       COLLECT REGION BOUNDARY NODES
C
C        LOOP OVER BOUNDARY SEGMENTS
C
      LNODE=0
      DO 520 ISEG=1,NBCS(IREG)
      NOSEG=IABS(IBCS(IREG,ISEG))
      NP1=NBNO(NOSEG,1)
      NN=NPOIN(NOSEG)
      NP2=NBNO(NOSEG,NN)
      NL1=NN-1
      IF(IBCS(IREG,ISEG).LT.0) GO TO 540
      DO 550 KN=1,NL1
      LNODE=LNODE+1
      NPFRONT(IREG,LNODE)=NBNO(NOSEG,KN)
      NQFRONT(IREG,LNODE)=NBNO(NOSEG,KN+1)
  550 CONTINUE
      GO TO 560
  540 DO 570 KN=1,NL1
      IN=NL1+1-KN
      LNODE=LNODE+1
      NPFRONT(IREG,LNODE)=NBNO(NOSEG,IN+1)
      NQFRONT(IREG,LNODE)=NBNO(NOSEG,IN)
  570 CONTINUE
  560 CONTINUE
C     WRITE(6,1000)IREG,ISEG
C     WRITE(6,1010)((ITEST,NODEL(ITEST)),ITEST=1,LNODE)
C1000 FORMAT(2I5)
C1010 FORMAT(10(I5,I5))
C
C        END OF LOOP OVER BOUNDARY SEGMENTS
C
  520 CONTINUE
C
C        SET UP INITIAL SET OF NODES FOR REGION
C
      DO 310 KNODE=1,LNODE
      NREGI(IREG,KNODE)=NPFRONT(IREG,KNODE)
  310 CONTINUE
      NONR(IREG)=LNODE
      NONF(IREG)=LNODE
C
C     WRITE(6,9020) (NREGI(IREG,KNODE),KNODE=1,LNODE)
C9020 FORMAT(10I10)
C
C        END LOOP OVER REGIONS
C
  100 CONTINUE
      RETURN
C
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE TRIANGLE(NREG,NONF,NPFRONT,NQFRONT,NREGI,NONR,IEL,
     1                    COOR,NELEM,NODE,NELEG,NPOIG,IELEG,INTMEG,
     2                    COORG,DELTA,STREC,TOLER,ILAST,IGRAPH)
C
C        GENERATE POINTS AND TRIANGULATE REGION BY REGION
C
      DIMENSION NONF(10),NREGI(10,2000),NONR(10)
      DIMENSION NPFRONT(10,2000),NQFRONT(10,2000)
      DIMENSION COOR(2,5000),STREC(4,5000)
      DIMENSION IEL(3,10000),NEAR(400),NEAR1(400),HOWF(400)
      DIMENSION HOWF1(400),AR(3)
      DIMENSION X(5000),Y(5000)
      DIMENSION NCHECK(5000),INTMEG(3,NELEG)
      DIMENSION IELEG(3,NELEG),COORG(2,NPOIG),DELTA(4,NPOIG)
C
C     XTR(XQ,YQ,AX,AY,ALPH)=AX*ALPH*(AX*XQ+AY*YQ)+AY*(AY*XQ-AX*YQ)
```

```
C     YTR(XQ,YQ,AX,AY,ALPH)=AY*ALPH*(AX*XQ+AY*YQ)-AX*(AY*XQ-AX*YQ)
C     XBA(XQ,YQ,AX,AY,ALPH)=(AX*(AX*XQ+AY*YQ)/ALPH)+AY*(AY*XQ-AX*YQ)
C     YBA(XQ,YQ,AX,AY,ALPH)=(AY*(AX*XQ+AY*YQ)/ALPH)-AX*(AY*XQ-AX*YQ)
C
      WRITE (6,*) 'AW FOR RE-ORDERING, (AW=1.:ALWAYS,AW=LARGE:NEVER)'
      WRITE(19,*) 'AW FOR RE-ORDERING, (AW=1.:ALWAYS,AW=LARGE:NEVER)'
      READ (7,*) AW
      WRITE(6,*) AW
      WRITE(19,*) AW
C
      DO 5 IN=1,NODE
      X(IN)=COOR(1,IN)
      Y(IN)=COOR(2,IN)
    5 CONTINUE
C
C
C        LOOP OVER REGIONS
C
      NELEM=0
C
      DO 100 IREG=1,NREG
C
      DO 570 IK=1,NODE
      NCHECK(IK)=-100
  570 CONTINUE
C
C     SET UP NCHECK VALUES FOR REGION
C
      DO 580 IK=1,NONF(IREG)
      K1=NPFRONT(IREG,IK)
      NCHECK(K1)=2
  580 CONTINUE
C
      DISW=0.0
C
  160 CONTINUE
      NL=NONF(IREG)
  161 CONTINUE
      KN=NQFRONT(IREG,NL)
      KN1=NPFRONT(IREG,NL)
      XN=COOR(1,KN)
      YN=COOR(2,KN)
      XN1=COOR(1,KN1)
      YN1=CDOR(2,KN1)
      ALPH=AMIN1(STREC(2,KN1),STREC(2,KN))
      ANX=0.5*(STREC(3,KN1)+STREC(3,KN))
      ANY=0.5*(STREC(4,KN1)+STREC(4,KN))
      ANM=SQRT(ANX*ANX+ANY*ANY)
      ANX=ANX/ANM
      ANY=ANY/ANM
C     XNF=XBA(XN,YN,ANX,ANY,ALPH)
C     YNF=YBA(XN,YN,ANX,ANY,ALPH)
C     XN1F=XBA(XN1,YN1,ANX,ANY,ALPH)
C     YN1F=YBA(XN1,YN1,ANX,ANY,ALPH)
      XNF=FXBA(XN,YN,ANX,ANY,ALPH)
      YNF=FYBA(XN,YN,ANX,ANY,ALPH)
      XN1F=FXBA(XN1,YN1,ANX,ANY,ALPH)
      YN1F=FYBA(XN1,YN1,ANX,ANY,ALPH)
      X12F=XNF-XN1F
      Y12F=YNF-YN1F
      ALEN1F=SQRT(X12F*X12F+Y12F*Y12F)
      IF(ALEN1F.LT.AW*DISW) GOTO 162
C
      DISW1=0.0
      CALL ORDER(NL,IREG,NPFRONT,NQFRONT,COOR,NPOIG,NELEG,
     1           COORG,IELEG,DELTA,STREC,DISW,DISW1)
C
      GOTO 161
  162 CONTINUE
C
      TOLE1=0.00001*ALEN1F
C
C      FIND OUT AVERAGE SPACING
C
      AVERAGE=AMIN1(STREC(1,KN1),STREC(1,KN))
      X12=XN-XN1
      Y12=YN-YN1
```

```
      ALEN1=SQRT(X12*X12+Y12*Y12)
C
C      CREATE A NEW NODE
C
C
C     SAFETY FACTOR
C
      CSAFE=1.0
      DSIDE=CSAFE*AVERAGE
      IF(DSIDE.GT.2.0*ALEN1F) DSIDE=2.0*ALEN1F
      IF(DSIDE.LT.0.55*ALEN1F) DSIDE=0.55*ALEN1F
      TWOD2=2.0*DSIDE*DSIDE
      XBARF=0.5*(XNF+XN1F)
      YBARF=0.5*(YNF+YN1F)
      XDIFF=XNF-XN1F
      YDIFF=YNF-YN1F
      DKARG=XDIFF*XDIFF+YDIFF*YDIFF
      D12=SQRT(DKARG)
      HKARG=0.5*TWOD2-0.25*D12*D12
      HK=SQRT(HKARG)
      XCBF=-HK*YDIFF/D12
      YCBF= HK*XDIFF/D12
      XTEMPF=XBARF+XCBF
      YTEMPF=YBARF+YCBF
C     XTEMP=XTR(XTEMPF,YTEMPF,ANX,ANY,ALPH)
C     YTEMP=YTR(XTEMPF,YTEMPF,ANX,ANY,ALPH)
      XTEMP=FXTR(XTEMPF,YTEMPF,ANX,ANY,ALPH)
      YTEMP=FYTR(XTEMPF,YTEMPF,ANX,ANY,ALPH)
C
C      LOOP OVER POSSIBLE NODES - FIND CLOSEST NEIGHBOURS
C
      A=YN1-YN
      B=XN-XN1
      C=(XN1-XN)*YN1+(YN-YN1)*XN1
      RADIUS=DSIDE
      H1=0.8*RADIUS
      INUM=0
      DO 110 KP=1,NONR(IREG)
      KEN=NREGI(IREG,KP)
      IF(KEN.EQ.KN.OR.KEN.EQ.KN1) GO TO 110
      XKEN=COOR(1,KEN)
      YKEN=COOR(2,KEN)
C     XKENF=XBA(XKEN,YKEN,ANX,ANY,ALPH)
C     YKENF=YBA(XKEN,YKEN,ANX,ANY,ALPH)
      XKENF=FXBA(XKEN,YKEN,ANX,ANY,ALPH)
      YKENF=FYBA(XKEN,YKEN,ANX,ANY,ALPH)
      XDIFF1=XKENF-XTEMPF
      YDIFF1=YKENF-YTEMPF
      DISTF=SQRT(XDIFF1*XDIFF1+YDIFF1*YDIFF1)
      IF(DISTF.GT.H1) GO TO 110
      IF(A*XKEN+B*YKEN+C.LE.0.0) GOTO 110
C
      INUM=INUM+1
      HOWF1(INUM)=DISTF
      NEAR1(INUM)=KEN
  110 CONTINUE
C
C          DECIDE WHICH OF THE NODES IS CHOSEN
C
       IF(INUM.EQ.0) THEN
       INUM=1
       NEAR(1)=0
       HOWF(1)=0.0
       ELSE
C
C          ORDER THEM
C
       DO 599 I=1,INUM
       COMP=1.E+6
       DO 598 J=1,INUM
       IF(NEAR1(J).EQ.0) GOTO 598
       IF(HOWF1(J).GT.COMP) GOTO 598
       IS=J
       COMP=HOWF1(J)
  598  CONTINUE
       NEAR(I)=NEAR1(IS)
       HOWF(I)=HOWF1(IS)
```

```
      NEAR1(IS)=0
  599 CONTINUE
C
C     ADD THE NEW POINT TO THE LIST
C
      INUM=INUM+1
      NEAR(INUM)=0
      HOWF(INUM)=0.0
C
C     IF THE DISTANCE IS TOO LARGE ADD POINT
C
C     INUM=INUM+1
C     NEAR(INUM)=0
C     HOWF(INUM)=0.0
C     DO 720 K=1,INUM
C     KI=NEAR(K)
C     CNX=COOR(1,KI)
C     CNY=COOR(2,KI)
C     CNXF=XBA(CNX,CNY,ANX,ANY,ALPH)
C     CNYF=YBA(CNX,CNY,ANX,ANY,ALPH)
C     WX1=CNXF-XTEMPF
C     WY1=CNYF-YTEMPF
C     D1=SQRT(WX1*WX1+WY1*WY1)
C     IF(D1.LT.0.5*DSIDE) GOTO 720
C     DO 575 I=1,INUM-K
C     J=INUM-I
C     NEAR(J+1)=NEAR(J)
C     HOWF(J+1)=HOWF(J)
C 575 CONTINUE
C     NEAR(K)=0
C     HOWF(K)=0.0
C     GOTO 721
C 720 CONTINUE
C 721 CONTINUE
C
      ENDIF
C
C     SELECT ---> START BY THE CLOSEST
C
      DO 601 I=1,INUM
      KP=NEAR(I)
        IF(KP.EQ.0) THEN
            XP=XTEMP
            YP=YTEMP
        ELSE
            XP=COOR(1,KP)
            YP=COOR(2,KP)
        ENDIF
C
C     SEE IF THIS CONNECTION IS POSSIBLE
C
      CALL POSSIBLE(KN1,KN,KP,XN1,YN1,XN,YN,XP,YP,IREG,NONF,
     1              NONR,NPFRONT,NQFRONT,NREGI,COOR,IYON)
C
      IF(IYON.EQ.0) GOTO 601
      GOTO 603
C
C       CHECK NOW THE SIZE
C
C     XPF=XBA(XP,YP,ANX,ANY,ALPH)
C     YPF=YBA(XP,YP,ANX,ANY,ALPH)
C     D1=SQRT((XPF-XN1F)**2+(YPF-YN1F)**2)
C     D2=SQRT((XPF-XNF)**2+(YPF-YNF)**2)
C     DI=AMAX1(D1,D2)
C     IF(DI.LT.1.5*DSIDE) GOTO 603
  601 CONTINUE
C
C     WE ARE IN TROUBLE !!!!!
C     FIND THE 30 EXISTING NODES THAT GIVE MAXIMUM ANGLE
C
      INUM=0
      ANG1=0.0
      DO 210 KP=1,NONR(IREG)
      KEN=NREGI(IREG,KP)
      IF(KEN.EQ.KN.OR.KEN.EQ.KN1) GO TO 210
      XKEN=COOR(1,KEN)
      YKEN=COOR(2,KEN)
```

```
      IF(A*XKEN+B*YKEN+C.LE.0.0) GOTO 210
C
C     SEE IF THIS CONNECTION IS POSSIBLE
C
      CALL POSSIBLE(KN1,KN,KEN,XN1,YN1,XN,YN,XKEN,YKEN,IREG,
     1              NONF,NONR,NPFRONT,NQFRONT,NREGI,COOR,IYON)
C
      IF(IYON.EQ.0) GOTO 210
C     XKENF=XBA(XKEN,YKEN,ANX,ANY,ALPH)
C     YKENF=YBA(XKEN,YKEN,ANX,ANY,ALPH)
      XKENF=FXBA(XKEN,YKEN,ANX,ANY,ALPH)
      YKENF=FYBA(XKEN,YKEN,ANX,ANY,ALPH)
      XDIFF1=XKENF-XN1F
      YDIFF1=YKENF-YN1F
      XDIFF2=XKENF-XNF
      YDIFF2=YKENF-YNF
      DISTF1=SQRT(XDIFF1*XDIFF1+YDIFF1*YDIFF1)
      DISTF2=SQRT(XDIFF2*XDIFF2+YDIFF2*YDIFF2)
      COSA=(XDIFF1*XDIFF2+YDIFF1*YDIFF2)/(DISTF1*DISTF2)
      IF(COSA.GT.1.0) COSA=1.0
      IF(COSA.LT.-1.0) COSA=-1.0
      ANGL=ACOS(COSA)
      IF(ANGL.LT.ANG1) GO TO 210
C
      HOWF(INUM+1)=ANGL
      NEAR(INUM+1)=KEN
C
      IF(INUM.EQ.0) GOTO 311
      DO 310 I=1,INUM
      K=I
      ANGI=HOWF(I)
      IF(ANGI.GT.ANGL) GOTO 310
      DO 410 J=1,INUM-K
      L=INUM-J
      NEAR(L+1)=NEAR(L)
      HOWF(L+1)=HOWF(L)
  410 CONTINUE
      NEAR(K)=KEN
      HOWF(K)=ANGL
      GOTO 311
  310 CONTINUE
  311 CONTINUE
      IF(INUM.LT.30) INUM=INUM+1
      IF(INUM.EQ.30) ANG1=HOWF(30)
  210 CONTINUE
C
C     SELECT ---> START BY THE CLOSEST
C
      IF(INUM.EQ.0) GOTO 703
      WFAR=1.E+6
      DO 701 I=1,INUM
      KP=NEAR(I)
      ANGI=HOWF(I)
      XP=COOR(1,KP)
      YP=COOR(2,KP)
C
C     SEE IF THIS CONNECTION IS POSSIBLE
C
C     CALL POSSIBLE(KN1,KN,KP,XN1,YN1,XN,YN,XP,YP,IREG,NONF,
C    1              NONR,NPFRONT,NQFRONT,NREGI,COOR,IYON)
C
C     IF(IYON.EQ.0) GOTO 701
C     IF(ANGI.GT.0.7853981) GOTO 603
C
C        CHECK NOW THE SIZE
C
C     XPF=XBA(XP,YP,ANX,ANY,ALPH)
C     YPF=YBA(XP,YP,ANX,ANY,ALPH)
      XPF=FXBA(XP,YP,ANX,ANY,ALPH)
      YPF=FYBA(XP,YP,ANX,ANY,ALPH)
      D1=SQRT((XPF-XN1F)**2+(YPF-YN1F)**2)
      D2=SQRT((XPF-XNF)**2+(YPF-YNF)**2)
      DI=AMAX1(D1,D2)
      IF(DI.LT.WFAR) KP1=KP
      IF(DI.LT.WFAR) WFAR=DI
  701 CONTINUE
      KP=KP1
```

```
      XP=COOR(1,KP)
      YP=COOR(2,KP)
      GOTO 603
  703 CONTINUE
C
      WRITE (6,*) ' CANNOT FIND THE CONNECTIVITY'
      WRITE(19,*) ' CANNOT FIND THE CONNECTIVITY'
C
C     TRY ANOTHER SIDE
C
      DISW1=1.01*ALEN1F
      CALL ORDER(NL,IREG,NPFRONT,NQFRONT,COOR,NPOIG,NELEG,
     1           COORG,IELEG,DELTA,STREC,DISW,DISW1)
C
      GOTO 161
C
C     NOTHING WRONG WITH IT
C
  603 CONTINUE
C
      INDIC=1
      KNEAR=KP
      IF(KNEAR.NE.0) INDIC=0
C
      IF(INDIC.EQ.0) GO TO 620
      NODE=NODE+1
      COOR(1,NODE)=XP
      COOR(2,NODE)=YP
C
C     FIND OUT INTERPOLATED VARIABLES
C
      INORM=1
      CALL FINDEL(NPOIG,NELEG,COORG,IELEG,INTMEG,XP,YP,ILAST,
     1            AR,I1,I2,I3,IENR)
      CALL GETVALUE(NPOIG,I1,I2,I3,AR,DELTA,DIS,ALP,ANX,ANY,INORM)
      STREC(1,NODE)=DIS
      STREC(2,NODE)=ALP
      STREC(3,NODE)=ANX
      STREC(4,NODE)=ANY
C
      NONR(IREG)=NONR(IREG)+1
      NUNO=NONR(IREG)
      NREGI(IREG,NUNO)=NODE
      KNEAR=NODE
      NCHECK(KNEAR)=-100
  620 CONTINUE
C
C       FORM ELEMENT
C
      NELEM=NELEM+1
      IEL(1,NELEM)=KN1
      IEL(2,NELEM)=KN
      IEL(3,NELEM)=KNEAR
C     WRITE(6,5000) NELEM,(IEL(IK1,NELEM),IK1=1,3)
C5000 FORMAT(4I5)
C
C      UPDATE FRONT AND ACTIVE NODES
C
      NL2=NL+1
      NQFRONT(IREG,NL2)=KN
      NQFRONT(IREG,NL)=KNEAR
      NPFRONT(IREG,NL2)=KNEAR
      IF(NCHECK(KNEAR).LT.0) NCHECK(KNEAR)=0
      NCHECK(KNEAR)=NCHECK(KNEAR)+2
      NONF(IREG)=NL2
C
C      DELETE SIDES FROM ACTIVE LIST
C
      NCHT=0
      NPASS=1
      NTOP=KN1
      NBOT=KNEAR
      MARKER=0
      KNON=NONF(IREG)-2
  360 DO 300 KP=1,KNON
      KTOP=NPFRONT(IREG,KP)
      KBOT=NQFRONT(IREG,KP)
```

```
      IF(MARKER.GT.0) GO TO 330
      IF((KTOP.EQ.NBOT).AND.(KBOT.EQ.NTOP)) GO TO 320
      GO TO 300
  320 MARKER=1
      NONF(IREG)=NONF(IREG)-2
      NCHECK(KTOP)=NCHECK(KTOP)-2
      NCHECK(KBOT)=NCHECK(KBOT)-2
      NCHT=1
      GO TO 300
  330 KP1=KP-1
      NPFRONT(IREG,KP1)=KTOP
      NQFRONT(IREG,KP1)=KBOT
  300 CONTINUE
C
      NPASS=NPASS+1
      IF(NPASS.GT.2) GO TO 340
      IF(MARKER.EQ.0) GO TO 350
      NPFRONT(IREG,NONF(IREG))=KNEAR
      NQFRONT(IREG,NONF(IREG))=KN
      MARKER=0
      KNON=KNON-1
      GO TO 400
  350 KNON=KNON+1
  400 NTOP=KNEAR
      NBOT=KN
      GO TO 360
  340 CONTINUE
C
C     REMOVE NODES FROM ACTIVE LIST
C
      IF(NCHT.EQ.0) GO TO 370
      IRED=0
      KNON=NONR(IREG)
      DO 380 KP=1,KNON
      KKN=NREGI(IREG,KP)
      KCH=NCHECK(KKN)
      IF(KCH.NE.0) GO TO 390
      IRED=IRED+1
      GO TO 380
  390 KP1=KP-IRED
      NREGI(IREG,KP1)=NREGI(IREG,KP)
  380 CONTINUE
      NONR(IREG)=KNON-IRED
  370 CONTINUE
      IF(NONF(IREG).GT.0) GO TO 160
C
C      END OF LOOP OVER REGIONS
C
  100 CONTINUE
C
      RETURN
      END
C
C     ---------------------------------------------------------
C
      SUBROUTINE POSSIBLE(KN1,KN,KP,XN1,YN1,XN,YN,XP,YP,IREG,
     1                    NONF,NONR,NPFRONT,NQFRONT,NREGI,
     2                    COOR,IYON)
C
C     THIS SUBROUTINE FINDS OUT WHETHER CONNECTION WHITH POINT KP
C     IS POSSIBLE IYON=1 OR NOT IYON=0
C
      DIMENSION NONF(10),NONR(10),NREGI(10,2000)
      DIMENSION NPFRONT(10,2000),NQFRONT(10,2000)
      DIMENSION COOR(2,5000)
C
      IYON=1
C
C     LOOP OVER THE FRONT NOOES
C
      DO 1000 IT=1,NONR(IREG)
      KJ=NREGI(IREG,IT)
      IF(KJ.EQ.KN1.OR.KJ.EQ.KN.OR.KJ.EQ.KP) GOTO 1000
      XT=COOR(1,KJ)
      YT=COOR(2,KJ)
C
C     CHECK IF THE POINT IS INTERIOR
```

```
C
      CALL INTERIOR(IIN,XN1,YN1,XN,YN,XP,YP,XT,YT)
      IF(IIN.EQ.0) GOTO 1000
      IYON=0
      RETURN
 1000 CONTINUE
C
C     EQUATION OF THE MID-BASE : KP LINE
C
      XMB=0.5*(XN1+XN)
      YMB=0.5*(YN1+YN)
      AS=YMB-YP
      BS=XP-XMB
      CS=(XMB-XP)*YMB+(YP-YMB)*XMB
C
C     LOOP OVER THE FRONT SIDES : CHECK FOR INTERSECTION
C
      DO 2000 IR=1,NONF(IREG)
      KNT1=NPFRONT(IREG,IR)
      IF(KNT1.EQ.0) GOTO 2000
      KNT=NQFRONT(IREG,IR)
C
      IF(KNT1.EQ.KN1.AND.KNT.EQ.KN) GOTO 2000
      IF(KNT1.EQ.KP.OR.KNT.EQ.KP) GOTO 2000
C
      XNT1=COOR(1,KNT1)
      YNT1=COOR(2,KNT1)
      XNT=COOR(1,KNT)
      YNT=COOR(2,KNT)
C
      AT=YNT1-YNT
      BT=XNT-XNT1
      CT=(XNT1-XNT)*YNT1+(YNT-YNT1)*XNT1
C
      S1=AT*XMB+BT*YMB+CT
      S2=AT*XP+BT*YP+CT
      S3=AS*XNT1+BS*YNT1+CS
      S4=AS*XNT+BS*YNT+CS
C
      SIG1=S1*S2
      SIG2=S3*S4
C
      IF(SIG1.GT.0.0.OR.SIG2.GT.0.0) GOTO 2000
      IYON=0
      RETURN
C
 2000 CONTINUE
C
      RETURN
      END
C
C     ---------------------------------------------------------
C
      SUBROUTINE INTERIOR(IIN,X1,Y1,X2,Y2,X3,Y3,X,Y)
C
C     DETERMINANT FUNCTION
C
C     DETER(P1,Q1,P2,Q2,P3,Q3)=P2*Q3-P3*Q2-P1*Q3+P3*Q1+P1*Q2-P2*Q1
C
      IIN=1
      AREA2=DETER(X1,Y1,X2,Y2,X3,Y3)
      IF(AREA2.LT.1.E-12) RETURN
      A1=DETER(X ,Y ,X2,Y2,X3,Y3)/AREA2
      A2=DETER(X1,Y1,X ,Y ,X3,Y3)/AREA2
      A3=DETER(X1,Y1,X2,Y2,X ,Y )/AREA2
C
      WCOMP=AMIN1(A1,A2,A3)
      IF(WCOMP.LT.-0.001) IIN=0
C
      RETURN
      END
C
C     ---------------------------------------------------------
C
      SUBROUTINE SMOOTH(NDIMN,NNODE,NELEM,NPOIN,NSMOO,
     &                  LCOOR,LCORE,INTMAT,COORD,COORO)
C
```

```
      REAL    COORD(2,NPOIN),COOR0(2,NPOIN)
      REAL    RDIVN(30),X(6,2),CN
      INTEGER LCOOR(NPOIN),LCORE(NPOIN)
      INTEGER INTMAT(3,NELEM),NODE(6)
C
      NDIVN=30
C
      DO 500 IDIVN=1,NDIVN
      RDIVN(IDIVN)=1./REAL(IDIVN)
  500 CONTINUE
C
C     -----SMOOTH OUT THE GRID IN NSMOO STEPS
C
      IF(NSMOO.EQ.0) GOTO 10001
C
      DO 10000 ISMOO=1,NSMOO
C
C     ----SET COOR0=0
C
      DO 1200 IP=1,NPOIN
      DO 1201 ID=1,2
      COOR0(ID,IP)=0.0
1201   CONTINUE
1200   CONTINUE
C
C     ----LOOP OVER THE ELEMENTS
C
      DO 2000 IELEM=1,NELEM
       DO 2100 IC=1,NNODE
       IN=INTMAT(IC,IELEM)
       DO 2101 ID=1,NDIMN
       X(      IC,ID)=COORD(ID,IN)
       X(NNODE+IC,ID)=COORD(ID,IN)
 2101  CONTINUE
       NODE(IC)=IN
 2100  CONTINUE
C
       DO 2200 IC=1,NNODE
       IN=NODE(IC)
       IF(LCOOR(IN).NE.0) GOTO 2199
        DO 2300 JC=1,NNODE-1
        DO 2301 ID=1,NDIMN
        COOR0(ID,IN)=COOR0(ID,IN)+X(IC+JC,ID)
 2301   CONTINUE
 2300   CONTINUE
 2199  CONTINUE
 2200  CONTINUE
C
 2000 CONTINUE
C
      DO 3000 ICOOR=1,NPOIN
      IF(LCOOR(ICOOR).NE.0) GOTO 3000
      IS=2*LCORE(ICOOR)
      CN=RDIVN(IS)
       DO 3100 ID=1,NDIMN
       COORD(ID,ICOOR)=CN*COOR0(ID,ICOOR)
 3100  CONTINUE
 3000 CONTINUE
C
10000 CONTINUE
10001 CONTINUE
C
      RETURN
      END
C
C     ---------------------------------------------------------------
C
      SUBROUTINE FINDEL(NPOIG,NELEG,COORG,IELEG,INTMEG,X,Y,ILAST,
     1                  AR,I1,I2,I3,IE)
C
C     THIS  SUBROUTINE   FINDS   OUT  IN  WHICH ELEMENT DOES THE POINT
C     (X,Y) LIE, AND CALCULATES THE AREA COORDINATES FOR INTERPOLATION
C
      DIMENSION COORG(2,NPOIG),IELEG(3,NELEG)
      DIMENSION INTMEG(3,NELEG)
      DIMENSION AR(3),ILO(3),OR(3)
C
```

```
C     DETERMINANT FUNCTION
C
C     DETER(P1,Q1,P2,Q2,P3,Q3)=P2*Q3-P3*Q2-P1*Q3+P3*Q1+P1*Q2-P2*Q1
C
C     START SEARCHING WITH ILAST
C
      IMEMO=0
      IF(ILAST.NE.0) GOTO 9
      WRITE (6,*) ' ILAST=0, THE BACKGROUND GRID NEEDS TO BE EXPANDED '
      WRITE(19,*) ' ILAST=0, THE BACKGROUND GRID NEEDS TO BE EXPANDED '
      STOP
    9 CONTINUE
      GOTO 10
   11 IMEMO=1
   10 CONTINUE
 1000 IE=ILAST
      I1=IELEG(1,IE)
      I2=IELEG(2,IE)
      I3=IELEG(3,IE)
      X1=COORG(1,I1)
      X2=COORG(1,I2)
      X3=COORG(1,I3)
      Y1=COORG(2,I1)
      Y2=COORG(2,I2)
      Y3=COORG(2,I3)
C
C     AREA COORDINATES
C
      AREA2=DETER(X1,Y1,X2,Y2,X3,Y3)
      AR(1)=DETER(X ,Y ,X2,Y2,X3,Y3)/AREA2
      AR(2)=DETER(X1,Y1,X ,Y ,X3,Y3)/AREA2
      AR(3)=DETER(X1,Y1,X2,Y2,X ,Y )/AREA2
C
C     ORDER THEM
C
      DO 101 I=1,3
      OR(I)=AR(I)
      ILO(I)=I
  101 CONTINUE
C
      DO 102 I=1,3
      J1=I+1
      DO 102 J=J1,3
      IF(OR(I)-OR(J)) 102,102,103
  103 TEMP=OR(I)
      ITEMP=ILO(I)
      OR(I)=OR(J)
      ILO(I)=ILO(J)
      OR(J)=TEMP
      ILO(J)=ITEMP
  102 CONTINUE
C
      IF(OR(1).GE.-1.E-2) GOTO 2000
C
C     GET THE NEXT ELEMENT
C
      INEXT=INTMEG(ILO(1),IE)
      IF(INEXT.EQ.0) THEN
            INEXT=INTMEG(ILO(2),IE)
            IF(OR(2).LT.-1.E-2.AND.INEXT.NE.0) THEN
            ILAST=INEXT
            GOTO 1000
            ELSE
            IF(IMEMO.EQ.0) GOTO 880
            WRITE (6,*) ' ERROR IN FINDEL'
            WRITE(19,*) ' ERROR IN FINDEL'
            STOP
  880       CONTINUE
C
C    LOOK FOR THE CLOSEST NODAL POINT
C
            ADIS=1.E+6
            DO 1010 IP=1,NPOIG
            XP=COORG(1,IP)
            YP=COORG(2,IP)
            ADI1=(X-XP)**2+(Y-YP)**2
            IF(ADI1.GT.ADIS) GOTO 1010
```

```
            ADIS=ADI1
            KPO=IP
 1010       CONTINUE
C
C      NOW GET AN ELEMENT CONTAINING THIS POINT
C
            DO 1020 IE=1,NELEG
            ILAST=IE
            DO 1021 IN=1,3
            IPO=IELEG(IN,IE)
            IF(IPO.EQ.KPO) GOTO 1022
 1021       CONTINUE
            GOTO 1020
 1022       CONTINUE
            GOTO 10
 1020       CONTINUE
            ENDIF
      ELSE
      ILAST=INEXT
      GOTO 1000
      ENDIF
C
 2000 CONTINUE
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE GETVALUE(NPOIG,I1,I2,I3,AR,DELTA,DIS,ALP,ANX,ANY,
     1                    INORM)
C
C     REAL DELTA(4,NPOIG),AR(3)
      REAL DELTA(4,5000),AR(3)
C
C     INTERPOLATE
C
      DIS=AR(1)*DELTA(1,I1)+AR(2)*DELTA(1,I2)+AR(3)*DELTA(1,I3)
      ALP=AR(1)*DELTA(2,I1)+AR(2)*DELTA(2,I2)+AR(3)*DELTA(2,I3)
      ANX=AR(1)*DELTA(3,I1)+AR(2)*DELTA(3,I2)+AR(3)*DELTA(3,I3)
      ANY=AR(1)*DELTA(4,I1)+AR(2)*DELTA(4,I2)+AR(3)*DELTA(4,I3)
C
C     NORMALIZE IF REQUIRED
C
      IF(INORM.EQ.0) RETURN
      ANM=SQRT(ANX*ANX+ANY*ANY)
      ANX=ANX/ANM
      ANY=ANY/ANM
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE ORDER(NL,IREG,NPFRONT,NQFRONT,COOR,NPOIG,
     1                 NELEG,COORG,IELEG,DELTA,STREC,DISW,
     2                 DISW1)
C
      DIMENSION NPFRONT(10,2000),NQFRONT(10,2000)
      DIMENSION DELTA(4,NPOIG),COOR(2,NPOIG),STREC(4,NPOIG)
      DIMENSION COORG(2,NPOIG),IELEG(3,NELEG)
C
      XBA(XQ,YQ,AX,AY,ALPH)=(AX*(AX*XQ+AY*YQ)/ALPH)+AY*(AY*XQ-AX*YQ)
      YBA(XQ,YQ,AX,AY,ALPH)=(AY*(AX*XQ+AY*YQ)/ALPH)-AX*(AY*XQ-AX*YQ)
C
      DISW=1.E+6
C     WRITE (6,*) 'ORDERING THE FRONT'
      DO 1000 IL=1,NL
      KN=NQFRONT(IREG,IL)
      KN1=NPFRONT(IREG,IL)
      XN=COOR(1,KN)
      YN=COOR(2,KN)
      XN1=COOR(1,KN1)
      YN1=COOR(2,KN1)
      ALPH=AMIN1(STREC(2,KN1),STREC(2,KN))
      ANX=0.5*(STREC(3,KN1)+STREC(3,KN))
      ANY=0.5*(STREC(4,KN1)+STREC(4,KN))
```

```
      ANM=SQRT(ANX*ANX+ANY*ANY)
      ANX=ANX/ANM
      ANY=ANY/ANM
      XNF2=XBA(XN,YN,ANX,ANY,ALPH)
      YNF2=YBA(XN,YN,ANX,ANY,ALPH)
      XN1F2=XBA(XN1,YN1,ANX,ANY,ALPH)
      YN1F2=YBA(XN1,YN1,ANX,ANY,ALPH)
      X12=XNF2-XN1F2
      Y12=YNF2-YN1F2
      DIS=SQRT(X12*X12+Y12*Y12)
      IF(DIS.LT.DISW1) GOTO 1000
      IF(DIS.GT.DISW) GOTO 1000
      IWH=IL
      DISW=DIS
      IQ=KN
      IP=KN1
 1000 CONTINUE
C
C     SWAP VALUES
C
      NPFRONT(IREG,IWH)=NPFRONT(IREG,NL)
      NQFRONT(IREG,IWH)=NQFRONT(IREG,NL)
      NPFRONT(IREG,NL)=IP
      NQFRONT(IREG,NL)=IQ
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      SUBROUTINE BOUNDAR(NPOIN,NELEM,LCOOR,INTMAT)
C
      DIMENSION LCOOR(NPOIN),INTMAT(3,NELEM)
C
C     THIS SUBROUTINE FINDS OUT THE BOUNDARY POINTS
C     LCOOR(IPOIN).EQ.0 --> INTERIOR
C     LCOOR(IPOIN).NE.0 --> BOUNDARY
C
C     INITIALIZE
C
      DO 1000 IP=1,NPOIN
      LCOOR(IP)=0
 1000 CONTINUE
C
C     LOOP OVER THE ELEMENTS
C
      DO 2000 IE=1,NELEM
      DO 2001 IN=1,3
      IN1=IN+1
      IF(IN1.GT.3) IN1=IN1-3
      IN2=IN+2
      IF(IN2.GT.3) IN2=IN2-3
      IP=INTMAT(IN,IE)
      LCOOR(IP)=LCOOR(IP)+INTMAT(IN2,IE)-INTMAT(IN1,IE)
 2001 CONTINUE
 2000 CONTINUE
C
      RETURN
      END
C
C     ------------------------------------------------
C
      SUBROUTINE CONERE(NPOIN,NELEM,INTMAT,LCORE)
C
      DIMENSION INTMAT(3,NELEM),LCORE(NPOIN)
C
      DO 1000 IP=1,NPOIN
      LCORE(IP)=0
 1000 CONTINUE
C
      DO 2000 IE=1,NELEM
      DO 2001 IN=1,3
      IP=INTMAT(IN,IE)
      LCORE(IP)=LCORE(IP)+1
 2001 CONTINUE
 2000 CONTINUE
C
```

```
      RETURN
      END
C
C     ------------------------------------------------
C
      SUBROUTINE CONEID(NPOIN,NELEM,INTMAT,LCOOR,LCOID,COORD,STREC)
C
      DIMENSION INTMAT(3,NELEM),LCOOR(NPOIN),LCOID(NPOIN)
      DIMENSION COORD(2,NPOIN),STREC(4,NPOIN)
C
      XBA(XQ,YQ,AX,AY,ALPH)=(AX*(AX*XQ+AY*YQ)/ALPH)+AY*(AY*XQ-AX*YQ)
      YBA(XQ,YQ,AX,AY,ALPH)=(AY*(AX*XQ+AY*YQ)/ALPH)-AX*(AY*XQ-AX*YQ)
C
      PI=3.1415927
C
C     THIS SUBR. FINDS OUT THE OPTIMAL NUMBER OF CONECTIVITIES
C     FOR EACH NODE
C
      DO 1000 IP=1,NPOIN
      LCOID(IP)=6
 1000 CONTINUE
C
C     SEARCH FOR A SIDE TO START
C
      DO 2000 IP=1,NPOIN
      IC=IP
      IF(LCOOR(IP).NE.0) GOTO 2001
 2000 CONTINUE
 2001 CONTINUE
C
      IA=0
      IB=0
      DO 3000 IE=1,NELEM
      DO 3001 IN=1,3
      IP=INTMAT(IN,IE)
      IF(IP.NE.IC) GOTO 3001
      INB=IN-1
      IF(INB.LT.1) INB=INB+3
      IF(LCOOR(INTMAT(INB,IE)).NE.0) IB=INTMAT(INB,IE)
      IF(IB.NE.0) GOTO 3002
 3001 CONTINUE
 3000 CONTINUE
 3002 CONTINUE
C
      IMEMO=IC
 4000 IA=IB-LCOOR(IC)
      ALPH=STREC(2,IC)
      ANX=STREC(3,IC)
      ANY=STREC(4,IC)
      X1R=COORD(1,IB)-COORD(1,IC)
      Y1R=COORD(2,IB)-COORD(2,IC)
      X2R=COORD(1,IA)-COORD(1,IC)
      Y2R=COORD(2,IA)-COORD(2,IC)
      X1=XBA(X1R,Y1R,ANX,ANY,ALPH)
      Y1=YBA(X1R,Y1R,ANX,ANY,ALPH)
      X2=XBA(X2R,Y2R,ANX,ANY,ALPH)
      Y2=YBA(X2R,Y2R,ANX,ANY,ALPH)
      COSA=(X1*X2+Y1*Y2)/(SQRT(X1*X1+Y1*Y1)*SQRT(X2*X2+Y2*Y2))
      IF(COSA.GT.1.0) COSA=1.0
      IF(COSA.LT.-1.0) COSA=-1.0
      THETA=ACOS(COSA)
      IF((X2*Y1-X1*Y2).LT.0.0) THETA=2.*PI-THETA
      DIVI=(THETA+(PI/6.))/(PI/3.)
      LCOID(IC)=DIVI
      IF(LCOID(IC).LT.1) LCOID(IC)=1
      IB=IC
      IC=IA
      IF(IC.NE.IMEMO) GOTO 4000
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE SIDE(NELEM,NPOIN,ILOCA,INTMAT,ISIDE,LWHER,LHOWM,
     1                ICONE)
C
```

```
      DIMENSION INTMAT(3,NELEM),ISIDE(4,30000)
      DIMENSION LWHER(NPOIN),LHOWM(NPOIN),ICONE(30000)
C
C     FILL IN LHOWM : NR. OF ELEMENTS PER NODE
C
      DO 1490 IP=1,NPOIN
      LHOWM(IP)=0
 1490 CONTINUE
      DO 1500 IE=1,NELEM
      DO 1500 IN=1,3
      IP=INTMAT(IN,IE)
      LHOWN(IP)=LHOWM(IP)+1
 1500 CONTINUE
 1501 CONTINUE
C
C      FILL IN LWHER : LOCATION OF EACH NODE INSIDE ICONE
C
      LWHER(1)=0
      DO 1600 IP=2,NPOIN
      LWHER(IP)=LWHER(IP-1)+LHOWM(IP-1)
 1600 CONTINUE
C
C      FILL IN ICONE : ELEMENTS IN EACH NODE
C
      DO 1690 IP=1,NPOIN
      LHOWM(IP)=0
 1690 CONTINUE
      DO 1700 IE=1,NELEM
      DO 1701 IN=1,3
      IP=INTMAT(IN,IE)
      LHOWM(IP)=LHOWM(IP)+1
      JLOCA=LWHER(IP)+LHOWM(IP)
      ICONE(JLOCA)=IE
 1701 CONTINUE
 1700 CONTINUE
C
C     LOOP OVER THE NODES
C
      ILOCA=0
C
      DO 3000 IP=1,NPOIN
      ILOC1=ILOCA
      IELE=LHOWN(IP)
      IF(IELE.EQ.0) GOTO 3000
C
C     INITIALIZE ISIDE ----> IMPORTANT FOR BOUNDARY SIDES
C
      DO 3001 IS=1,IELE+2
      ISIDE(3,IS+ILOC1)=0
      ISIDE(4,IS+ILOC1)=0
 3001 CONTINUE
C
      IWHER=LWHER(IP)
C
C     LOOP OVER ELEMENTS SURROUNDING THE POINT IP
C
      IP1=IP
      DO 3090 IEL=1,IELE
      IE=ICONE(IWHER+IEL)
C
C     FIND OUT POSITION OF IP IN THE CONEIVITY MATRIX
C
      DO 3091 IN=1,3
      IN1=IN
      IPT=INTMAT(IN,IE)
      IF(IPT.EQ.IP) GOTO 3092
 3091 CONTINUE
 3092 CONTINUE
C
      DO 3100 J=1,2
      IN2=IN1+J
      IF(IN2.GT.3) IN2=IN2-3
      IP2=INTMAT(IN2,IE)
      IF(IP2.LT.IP1) GOTO 3100
C
C     CHECK THE SIDE ----->  NEW OR OLD
C
```

```
      IF(ILOCA.EQ.ILOC1) GOTO 7304
      DO 5600 IS=ILOC1+1,ILOCA
      JLOCA=IS
      IF(ISIDE(2,IS).EQ.IP2) GOTO 7303
 5600 CONTINUE
 7304 CONTINUE
C
C     NEW SIDE
C
      ILOCA=ILOCA+1
      ISIDE(1,ILOCA)=IP1
      ISIDE(2,ILOCA)=IP2
      ISIDE(2+J,ILOCA)=IE
      GOTO 3012
C
C     OLD SIDE
C
 7303 CONTINUE
      ISIDE(2+J,JLOCA)=IE
 3012 CONTINUE
C
 3100 CONTINUE
C
C     END LOOP OVER ELEMENTS SURROUNDING POINT IP
C
 3090 CONTINUE
C
      DO 8000 IS=ILOC1+1,ILOCA
      IF(ISIDE(3,IS).NE.0) GOTO 8000
      ISIDE(3,IS)=ISIDE(4,IS)
      ISIDE(4,IS)=0
      ISIDE(1,IS)=ISIDE(2,IS)
      ISIDE(2,IS)=IP1
 8000 CONTINUE
C
C     END LOOP OVER POINTS
C
 3000 CONTINUE
C
      RETURN
      END
C
C     ---------------------------------------------------------
C
      SUBROUTINE SWAPDI(NPOIN,NELEM,NSIDE,ISIDE,INTMAT,LCORE,
     1                  LCOID,LCOOR,COORD,LWHER,LHOWM,ICONE)
C
      DIMENSION ISIDE(4,30000),INTMAT(3,NELEM),LCOOR(NPOIN)
      DIMENSION LCORE(NPOIN),LCOID(NPOIN),COORD(2,NPOIN)
      DIMENSION LWHER(5000),LHOWM(5000),ICONE(30000)
C
C     THIS SUBROUTINE SWAPS THE DIAGONALS TO OBTAIN A MORE
C     EVEN DISTRIBUTION OF ELEMENTS PER NODE
C
C     DETERMINANT FUNCTION
C
C     DETER(P1,Q1,P2,Q2,P3,Q3)=P2*Q3-P3*Q2-P1*Q3+P3*Q1+P1*Q2-P2*Q1
C
      ISWAPLP=0
 1000 ICHAN=0
C
C     LOOP OVER THE SIDES
C
      DO 2000 IS=1,NSIDE
      I1=ISIDE(1,IS)
      I2=ISIDE(2,IS)
      IE1=ISIDE(3,IS)
      IE2=ISIDE(4,IS)
C
C      CHECK FOR BOUNDARY SIDES
C
      IF(IE2.EQ.0) GOTO 2000
C
C     DETERMINE I3 & I4
C
      DO 5000 IN=1,3
      IN1=IN+1
```

```
      IF(IN1.GT.3) IN1=IN1-3
      IN2=IN+2
      IF(IN2.GT.3) IN2=IN2-3
      IP11=INTMAT(IN,IE1)
      IP12=INTMAT(IN1,IE1)
      IP13=INTMAT(IN2,IE1)
      IF(IP11.EQ.I1.AND.IP12.EQ.I2) I3=IP13
      IP21=INTMAT(IN,IE2)
      IP22=INTMAT(IN1,IE2)
      IP23=INTMAT(IN2,IE2)
      IF(IP21.EQ.I2.AND.IP22.EQ.I1) I4=IP23
 5000 CONTINUE
C
C     FIND 'DEFICIT' OR 'SUPERHAVIT' OF CONNECTIVITIES
C
      IH1=IABS(LCORE(I1)-LCOID(I1))
      IH2=IABS(LCORE(I2)-LCOID(I2))
      IH3=IABS(LCORE(I3)-LCOID(I3))
      IH4=IABS(LCORE(I4)-LCOID(I4))
      IHF1=IABS(LCORE(I1)-1-LCOID(I1))
      IHF2=IABS(LCORE(I2)-1-LCOID(I2))
      IHF3=IABS(LCORE(I3)+1-LCOID(I3))
      IHF4=IABS(LCORE(I4)+1-LCOID(I4))
C
C     CHECK IF IT IS WORTH TO SWAP
C
      ISWAP=0
      IACTU=IH1+IH2+IH3+IH4
      IFUTU=IHF1+IHF2+IHF3+IHF4
      IF(IACTU.GT.IFUTU) ISWAP=1
      IAM=MAX0(IH1,IH2,IH3,IH4)
      IFM=MAX0(IHF1,IHF2,IHF3,IHF4)
      IF(IACTU.EQ.IFUTU.AND.IAM.GT.(IFM+1)) ISWAP=1
      IF(ISWAP.EQ.0) GOTO 2000
C
C     CHECK AREA
C
      X1=COORD(1,I1)
      X2=COORD(1,I4)
      X3=COORD(1,I3)
      Y1=COORD(2,I1)
      Y2=COORD(2,I4)
      Y3=COORD(2,I3)
C     IF(DETER(X1,Y1,X2,Y2,X3,Y3).LE.0.0) GOTO 2000
      DTMP = DETER(X1,Y1,X2,Y2,X3,Y3)
      IF(DTMP.LE.0.0) GOTO 2000
      X1=COORD(1,I3)
      X2=COORD(1,I4)
      X3=COORD(1,I2)
      Y1=COORD(2,I3)
      Y2=COORD(2,I4)
      Y3=COORD(2,I2)
C     IF(DETER(X1,Y1,X2,Y2,X3,Y3).LE.0.0) GOTO 2000
      DTMP = DETER(X1,Y1,X2,Y2,X3,Y3)
      IF(DTMP.LE.0.0) GOTO 2000
C
C     SWAP
C
      ICHAN=ICHAN+1
      INTMAT(1,IE1)=I1
      INTMAT(2,IE1)=I4
      INTMAT(3,IE1)=I3
      INTMAT(1,IE2)=I3
      INTMAT(2,IE2)=I4
      INTMAT(3,IE2)=I2
      LCORE(I1)=LCORE(I1)-1
      LCORE(I2)=LCORE(I2)-1
      LCORE(I3)=LCORE(I3)+1
      LCORE(I4)=LCORE(I4)+1
C
C     DETECT THE SIDES I1-I4 AND I3-I2
C
      IST1=0
      IST2=0
      DO 4000 IST=1,NSIDE
      I1T=ISIDE(1,IST)
      I2T=ISIDE(2,IST)
```

```
      IF((I1T.EQ.I1.AND.I2T.EQ.I4).OR.(I1T.EQ.I4.AND.I2T.EQ.I1))
     1IST1=IST
      IF((I1T.EQ.I3.AND.I2T.EQ.I2).OR.(I1T.EQ.I2.AND.I2T.EQ.I3))
     1IST2=IST
      IF(IST1.NE.0.AND.IST2.NE.0) GOTO 4001
 4000 CONTINUE
      WRITE (6,*) ' ERROR IN SWAPDI '
      WRITE(19,*) ' ERROR IN SWAPDI '
      STOP
 4001 CONTINUE
      IF(ISIDE(3,IST1).EQ.IE2) ISIDE(3,IST1)=IE1
      IF(ISIDE(4,IST1).EQ.IE2) ISIDE(4,IST1)=IE1
      IF(ISIDE(3,IST2).EQ.IE1) ISIDE(3,IST2)=IE2
      IF(ISIDE(4,IST2).EQ.IE1) ISIDE(4,IST2)=IE2
C
C     UPDATE ISIDE
C
      ISIDE(1,IS)=I4
      ISIDE(2,IS)=I3
C
 2000 CONTINUE
C
      WRITE (6,1002) ICHAN
      WRITE(19,1002) ICHAN
 1002 FORMAT(I6,' SIDES HAVE BEEN SWAPPED')
C
      ISWAPLP=ISWAPLP+1
      IF(ICHAN.GE.1) GOTO 1000
C
C       FILL IN ISIDE
C
      IF (ISWAPLP.GT.1) THEN
       WRITE (6,*) ' *** FILLING ISIDE'
       WRITE(19,*) ' *** FILLING ISIDE'
       CALL SIDE(NELEM,NPOIN,NSIDE,INTMAT,ISIDE,LWHER,LHOWM,ICONE)
      ENDIF
C
      RETURN
      END
C
C     ------------------------------------------------------
C
        SUBROUTINE EAT3(NPOIN,NELEM,NSIDE,INTMAT,ISIDE,LCOOR,LPOSI,
     1                  LWHER,LHOWM,LCORE,LCOID,ICONE,COORD)
C
        INTEGER INTMAT(3,10000),ISIDE(4,30000),LCOOR(5000)
        INTEGER LPOSI(5000),LWHER(5000),LHOWM(5000)
        INTEGER LCORE(5000),LCOID(5000),ICONE(30000)
C
        REAL COORD(2,5000)
C
C       THIS SUBROUTINE REMOVES THE POINTS WHERE ONLY THREE
C       ELEMENTS COINCIDE
C
    1   NTRES=0
        KPOIN=0
C
C       LOOP OVER NUMBER OF NODES
C
        DO 1000 IP=1,NPOIN
        KPOIN=KPOIN+1
        LPOSI(IP)=KPOIN
C
C       IF BOUNDARY POINT LEAVE IT
C
        IF(LCOOR(IP).NE.0) GOTO 1000
C
C       CHECK NUMBER OF ELEMENTS
C
        IF(LCORE(IP).NE.3) GOTO 1000
C
C       A POINT WITH ONLY THREE ELEMENTS
C
C       GET THE ELEMENTS FROM ICONE
C
        ILOCA=LWHER(IP)
        IE1=ICONE(ILOCA+1)
```

```
          IE2=ICONE(ILOCA+2)
          IE3=ICONE(ILOCA+3)
        IF (IE1.EQ.0.OR.IE2.EQ.0.OR.IE3.EQ.0) GO TO 1000
          NTRES=NTRES+1
          KPOIN=KPOIN-1
          LPOSI(IP)=0
C
C         GET NEW CONECTIVITY POINT FOR IE1 FROM IE2
C
          IP1=INTMAT(1,IE1)
          IP2=INTMAT(2,IE1)
          IP3=INTMAT(3,IE1)
C
          DO 1002 IN=1,3
          IPT=INTMAT(IN,IE2)
          IF(IPT.NE.IP1.AND.IPT.NE.IP2.AND.IPT.NE.IP3) INO=IPT
 1002     CONTINUE
C
C         REPLACE CONECTIVITY
C
          DO 1003 IN=1,3
          IF(INTMAT(IN,IE1).EQ.IP) INTMAT(IN,IE1)=INO
          LCORE(INTMAT(IN,IE1))=LCORE(INTMAT(IN,IE1))-1
 1003     CONTINUE
C
          DO 1004 IN=1,3
          INTMAT(IN,IE2)=0
          INTMAT(IN,IE3)=0
 1004     CONTINUE
C
C         END LOOP OVER POINTS
C
C       WRITE (6,30001) IP,IE1,IE2,IE3,(INTMAT(IN,IE1),IN=1,3)
C30001  FORMAT(' EATING ',I8,2X,3I8,2X,3I8)
        ILOCA=LWHER(INO)
        DO 1005 IN=1,LCORE(INO)
        IF (ICONE(ILOCA+IN).EQ.IE2.OR.
     .      ICONE(ILOCA+IN).EQ.IE3) ICONE(ILOCA+IN)=0
 1005 CONTINUE
 1000   CONTINUE
C
C         TRANSFER
C
          DO 2000 IP=1,NPOIN
          IL=LPOSI(IP)
          IF(IL.EQ.0) GOTO 2000
          LCOOR(IL)=LCOOR(IP)
          LCORE(IL)=LCORE(IP)
          LCOID(IL)=LCOID(IP)
          DO 2001 ID=1,2
          COORD(ID,IL)=COORD(ID,IP)
 2001     CONTINUE
 2000     CONTINUE
C
          JE=0
          DO 3000 IE=1,NELEM
          IF(INTMAT(1,IE).EQ.0) GOTO 3000
          JE=JE+1
          DO 3001 IN=1,3
          IOLD=INTMAT(IN,IE)
          INEW=LPOSI(IOLD)
          INTMAT(IN,JE)=INEW
 3001     CONTINUE
 3000     CONTINUE
C
C         GET NPOIN AND NELEM
C
          NPOIN=NPOIN-NTRES
          NELEM=NELEM-2*NTRES
C
C         OUTPUT NR. OF EATEN POINTS
C
          WRITE (6,*) ' *** NR. OF 3"S REMOVED = ',NTRES
          WRITE(19,*) ' *** NR. OF 3"S REMOVED = ',NTRES
C
C         FILL IN ISIDE
C
```

```
        IF (NTRES.GT.0) THEN
         WRITE (6,*) ' *** FILLING ISIDE'
         WRITE(19,*) ' *** FILLING ISIDE'
         CALL SIDE(NELEM,NPOIN,NSIDE,INTMAT,ISIDE,LWHER,LHOWM,ICONE)
        ENDIF
C       IF (NTRES.GT.0) GO TO 1
C
         RETURN
         END
C
C     ------------------------------------------------------
C
      SUBROUTINE AREACH(NPOIN,NELEM,INTMAT,COORD)
C
      DIMENSION COORD(2,NPOIN),INTMAT(3,NELEM)
C
C     DETERMINANT FUNCTION
C
C     DETER(P1,Q1,P2,Q2,P3,Q3)=P2*Q3-P3*Q2-P1*Q3+P3*Q1+P1*Q2-P2*Q1
C
      ISUCC=0
      DO 1000 IE=1,NELEM
      I1=INTMAT(1,IE)
      I2=INTMAT(2,IE)
      I3=INTMAT(3,IE)
C
      X1=COORD(1,I1)
      X2=COORD(1,I2)
      X3=COORD(1,I3)
      Y1=COORD(2,I1)
      Y2=COORD(2,I2)
      Y3=COORD(2,I3)
C
      DTMP=DETER(X1,Y1,X2,Y2,X3,Y3)
      IF(DTMP.GT.0.0) GOTO 1000
C
      WRITE (6,1002) IE,I1,X1,Y1,I2,X2,Y2,I3,X3,Y3
      WRITE(19,1002) IE,I1,X1,Y1,I2,X2,Y2,I3,X3,Y3
 1002 FORMAT('  AREA ERROR IN ELEMENT ',I5,/,
     1       '  NODE 1 =',I5,'  X1 =',F10.5,'  Y1 =',F10.5,/,
     2       '  NODE 2 =',I5,'  X2 =',F10.5,'  Y2 =',F10.5,/,
     3       '  NODE 3 =',I5,'  X3 =',F10.5,'  Y3 =',F10.5,/)
C
      ISUCC=1
C
 1000 CONTINUE
C
      IF(ISUCC.EQ.0) WRITE (6,*) '  THE CHECKING WAS SUCCESSFUL !!!!'
      IF(ISUCC.EQ.0) WRITE(19,*) '  THE CHECKING WAS SUCCESSFUL !!!!'
C
C     OUTPUT NUMBER OF NODES AND ELEMENTS
C
      WRITE(6,228) NPOIN,NELEM
      WRITE(19,228) NPOIN,NELEM
  228 FORMAT(' TOTAL NUMBER OF GENERATED POINTS :',I5,/
     1       ' TOTAL NUMBER OF GENERATED ELEMENTS :',I5)
C
      RETURN
      END
C
C     ------------------------------------------------------------
C
      SUBROUTINE ELECON(NSIDE,INTMAT,ISIDE,INTMEL)
C
C     THIS SUBROUTINE FILLS IN THE ELEMENT CONECTIVITY MATRIX NEEDED
C     FOR THE FAST SEARCHING ALGORITHM
C
      DIMENSION INTMAT(3,10000),ISIDE(4,30000),INTMEL(3,10000)
C
C     LOOP OVER THE SIDES
C
      DO 1000 IS=1,NSIDE
      IP1=ISIDE(1,IS)
      IE1=ISIDE(3,IS)
      IE2=ISIDE(4,IS)
C
C     FIRST IE1
```

```
C
      I1=INTMAT(1,IE1)
      I2=INTMAT(2,IE1)
      I3=INTMAT(3,IE1)
      IF(IP1.EQ.I1) IPOS=3
      IF(IP1.EQ.I2) IPOS=1
      IF(IP1.EQ.I3) IPOS=2
C
C     GO INTO INTMEL
C
      INTMEL(IPOS,IE1)=IE2
C
C     NOW IE2 IF IE2.NE.0
C
      IF(IE2.EQ.0) GOTO 1000
      I1=INTMAT(1,IE2)
      I2=INTMAT(2,IE2)
      I3=INTMAT(3,IE2)
      IF(IP1.EQ.I1) IPOS=2
      IF(IP1.EQ.I2) IPOS=3
      IF(IP1.EQ.I3) IPOS=1
C
C     GO INTO INTMEL
C
      INTMEL(IPOS,IE2)=IE1
C
C     END LOOP OVER THE SIDES
C
 1000 CONTINUE
C
      RETURN
      END
C
C     ------------------------------------------------------
C
       SUBROUTINE OUTPUT(NPOIG,NELEG,COORG,INTMAG,INTMEG,UNKNG,
     1                   NPOIN,NELEM,COORD,INTMAT,RCOND,
     2                   NBOUN,LPOIN,ISIDE,NSIDE,TOLER,ILAST)
C
       DIMENSION COORG(2,NPOIG),UNKNG(4,NPOIG)
       DIMENSION INTMAG(3,NELEG),INTMEG(3,NELEG)
       DIMENSION COORD(2,NPOIN),INTMAT(3,NELEM),AR(3)
       DIMENSION RCOND(2,700),LPOIN(NBOUN),ISIDE(4,30000)
C
C     -----THIS SUB IS AN OUTPUT-FOR-INPUT ROUTINE
C
C     -----READ THE UNKNOWNS FOR INTERPOLATION
C
      ZERO = 0.
      IZ = 0
C     WRITE(11,'(A)') ' -888'
      DO 100 I=1,NPOIN
C       WRITE(11,50) I, (COORD(J,I), J=1,2), ZERO, ZERO, ZERO, ZERO
C  50   FORMAT(I5,1x,6(F12.8,4x))
       WRITE(13,55) I, IZ, (COORD(J,I), J=1,2), ZERO, IZ
   55  FORMAT('GRID',7x,i5,7x,i1,3(1x,f7.4),7x,i1)
  100 CONTINUE
      N = 1
      L = 0
      DO 200 I=1,NELEM
C      WRITE(12,150) (INTMAT(J,I),J=1,3),INTMAT(3,I),N,N,N,N,N,N,N,I,L
C150  FORMAT(13I6)
       WRITE(13,155) I, N, (INTMAT(J,I),J=1,3)
  155  FORMAT('CTRIA3',5x,i5,7x,i1,3(3x,i5))
  200 CONTINUE
C
      WRITE(8,1)NELEM,NPOIN,NBOUN
      WRITE(8,6)
C
C     NZERO = 0
      WRITE(10,21)  NELEM, NPOIN, NBOUN
C
      READ(17,36) TEXT
      READ(17,*)  GAMMA, EPSLAM, NTIME
C
      READ(17,36) TEXT
      READ(17,*)  P00,U00,V00,E00
```

```
C
      WRITE(10,31) GAMMA, EPSLAM, NTIME
C
      WRITE(10,22)  NELEM
      DO 500  I=1,NELEM
      WRITE(10,23)  I, (INTMAT(J,I), J=1,3)
  500 CONTINUE
      WRITE(10,24)  NPOIN
      DO 510  I=1,NPOIN
      WRITE(10,25)  I, (COORD(J,I), J=1,2)
  510 CONTINUE
      WRITE(10,26)  NPOIN
C
       DO 1002 I=1,NPOIN
C
C      INTERPOLATE THE UNKNOWN
C
       INORM=0
       CALL FINDEL(NPOIG,NELEG,COORG,INTMAG,INTMEG,COORD(1,I),
     1             COORD(2,I),ILAST,AR,I1,I2,I3,IENR)
       INORM=0
       CALL GETVALUE(NPOIG,I1,I2,I3,AR,UNKNG,RO,UV,VV,EN,INORM)
C
       IF(I.GT.NBOUN) GOTO 1202
C**    IF(LPOIN(I).NE.2) GOTO 1202
C**
C**    IF THE BOUNDARY CONDITION CODE = 2 (I.E. THE CONDITION FOR THE ABOVE
C**    STATEMENT IS MET) THE NODE IS ON THE SUPERSONIC EXIT BOUNDARY.  THE
C**    2 STATEMENTS BELOW GIVE WRONG EXIT U- AND V-VELOCITIES.  IF WE DON'T
C**    DO PROJECTION, THE VELOCITIES ARE CORRECT.  THUS, JUMP THE FOLLOWING
C**    5 STATEMENTS AND PRINT OUT NODAL UNKNOWNS DIRECTLY: (20 JULY '88)
C**
      IF(NPOIN.GT.0)  GO TO 1202
C
C      PROJECT
C
       TANX= RCOND(2,I)
       TANY=-RCOND(1,I)
       VELOC=TANX*UV+TANY*VV
       UV=VELOC*TANX
       VV=VELOC*TANY
 1202  WRITE(8,5)I,RO,UV,VV,EN,(COORD(J,I),J=1,2)
C      WRITE(10,27)  I, RO, UV, VV, EN
       WRITE(10,27)  I, P00, U00, V00, E00
 1002 CONTINUE
C
       WRITE(8,2)
       DO 1000 I=1,NELEM
       WRITE(8,3)I,(INTMAT(J,I),J=1,3),0
 1000 CONTINUE
C
       WRITE(8,7)
       DO 1003 I=1,NBOUN
       A=ABS(RCOND(1,I))+ABS(RCOND(2,I))
       IF(A.GT.1.E-5) WRITE(8,8) I,LPOIN(I),(RCOND(J,I),J=1,2)
       IF(A.LT.1.E-5) WRITE(8,9) I
       IF(A.LT.1.E-5) LPOIN(I)=100
 1003 CONTINUE
C
      WRITE(8,10)
      WRITE(10,28)  NBOUN
       DO 2500 IS=1,NSIDE
       IF(ISIDE(4,IS).NE.0) GOTO 2500
       I1=ISIDE(1,IS)
       I2=ISIDE(2,IS)
       IE=ISIDE(3,IS)
       IB=LPOIN(I1)
       IF(IB.EQ.100) IB=LPOIN(I2)
       WRITE(8,11) I1,I2,IE,IB
      WRITE(10,29)  I1, I2, IE, IB
 2500  CONTINUE
C
C     ----FORMATS
C
    1 FORMAT('  NELEM   NPOIN   NBOUN ',/,4I8)
    2 FORMAT(' INTERDEPENDENCY MATRIX INTMAT :')
    3 FORMAT(I10,I15,10I10)
```

```
    5 FORMAT(I8,2X,1P6E15.7)
    6 FORMAT(' LAST VALUES FOR THE UNKNOWNS :')
    7 FORMAT(' BOUNDARY MARKERS AND NORMALS :')
    8 FORMAT(2I8,1P2E15.7)
    9 FORMAT(I8,'     ********************************')
   10 FORMAT('   OUTPUT BSIDO')
   11 FORMAT(5I10)
C
   21 FORMAT(' NELEM   NPOIN   NBOUN ', /, 3I8)
   22 FORMAT(' ELEMENT NODAL CONNECTIONS [', I5, ']:')
   23 FORMAT(4I8)
   24 FORMAT(' NODAL COORDINATES [', I5, ']:')
   25 FORMAT(I8, 2E30.14)
   26 FORMAT(' NODAL INITIAL CONDITIONS [', I5, ']:')
   27 FORMAT(I8, 4E16.8)
   28 FORMAT(' BOUNDARY CONDITIONS [', I4, ']:')
   29 FORMAT(4I8)
C
   31 FORMAT(' INPUT GAMMA,EPSLAM,NTIME', /,F12.6,3X,
     *       F14.7,1X,I12)
   36 FORMAT(10A8)
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      SUBROUTINE REF2(NPOIG,NELEG)
C
C     THIS PROGRAM GENERATES THE REQUIRED PARAMETERS FOR RE-MESHING
C     FROM A PREVIOUSLY OBTAINED SOLUTION
C
C     ----GLOBAL ARRAYS
C
      COMMON/ASTORE/COORG(2,5000),UNKNG(4,5000),DELTA(4,5000),
     1 GEOMG(7,10000),MMATG(5000),DERIP(2,5000),DERI2(4,5000),
     2 HELP(4,5000),DERIA(4,5000),STREC(4,5000),COORN(2,5000),
     3 COOR(2,5000),COOR0(2,5000),RCOND(2,700),TEXT(20),AR(3),
     4 IELEG(3,10000),NPFRONT(10,2000),NQFRONT(10,2000),IEL(3,10000),
     5 LCOOR(5000),LCORE(5000),LCOID(5000),ISIDE(4,30000),
     6 INTMEG(3,10000),LWHER(5000),LKOWN(5000),ICONE(30000),
     7 NBNO(30,500),NREGI(10,2000),LBOU(2,30),ICOND(30),NNN(30),
     8 IBCS(10,30),MBCS(10),NONF(10),NONR(10),LBOUD(700),LPOSI(5000)
C
      REAL MMATG
C
C     -----OBTAIN GLOBALLY THE GEOMETRICAL VALUES NEEDED
C
C      NDIMN=2
       NAMAT=4
       NNODE=3
       NGEOM=7
C
       CALL GETGEO(NELEG,NPOIG,NNODE,NGEOM,
     &            IELEG,COORG,GEOMG,NSTOP)
C
C     ------INITIALIZE
C
      DO 1702 IPOIN=1,NPOIG
      DELTA(1,IPOIN)=1.E+6
      DELTA(2,IPOIN)=1.E+6
 1702 CONTINUE
C
C     -----INQUIRE FOR MAXIMUM AND MINIMUM SPACING
C
      WRITE (6,*) ' ENTER DELKMMAX AND DELKMMIN DELSCA'
      WRITE(19,*) ' ENTER DELKMMAX AND DELKMMIN DELSCA'
      READ (7,*) DELKMMA,DELKMMI,DELSCA
      WRITE(6,*) DELKMMA,DELKMMI,DELSCA
      WRITE(19,*) DELKMMA,DELKMMI,DELSCA
      WRITE (6,*) ' ENTER MAXIMUM STRETCHING'
      WRITE(19,*) ' ENTER MAXIMUM STRETCHING'
      READ (7,*) STR
      WRITE(6,*) STR
      WRITE(19,*) STR
C
C     -----INQUIRE FOR REFINEMENT CRITERIA
```

```
C
      WRITE (6,*) ' WHAT DO YOU WANT TO USE AS KEY VARIABLE ?'
      WRITE (6,*) ' 1 - DENSITY'
      WRITE (6,*) ' 2 - VELOCITY (MODULUS)'
      WRITE (6,*) ' 3 - DENSITY + VELOCITY'
      WRITE (6,*) ' 4 - ENTROPY'
      WRITE (6,*) ' 5 - DENSITY + MACH NO'
      WRITE (6,*) ' 6 - MACH NUMBER'
C
      WRITE(19,*) ' WHAT DO YOU WANT TO USE AS KEY VARIABLE ?'
      WRITE(19,*) ' 1 - DENSITY'
      WRITE(19,*) ' 2 - VELOCITY (MODULUS)'
      WRITE(19,*) ' 3 - DENSITY + VELOCITY'
      WRITE(19,*) ' 4 - ENTROPY'
      WRITE(19,*) ' 5 - DENSITY + MACH NO'
      WRITE(19,*) ' 6 - MACH NUMBER'
      READ (7,*) ICRIT
      WRITE(6,*) ICRIT
      WRITE(19,*) ICRIT
      IBACK=0
      IVARI=1
      IF(ICRIT.EQ.2.OR.ICRIT.EQ.3) IVARI=2
      IF(ICRIT.EQ.4) IVARI=3
      IF(ICRIT.EQ.5.OR.ICRIT.EQ.6) IVARI=4
      IF(ICRIT.EQ.3.OR.ICRIT.EQ.5) IBACK=1
      GOTO 1210
 1209 IF(IBACK.EQ.1) THEN
      IVARI=1
      IBACK=0
      ELSE
      ENDIF
 1210 CONTINUE
C
C     -----OBTAIN THE GRADIENT AT THE NODES
C
      CALL GETVAR(NNODE,NAMAT,NGEOM,2,NELEG,NPOIG,
     &            COORG,IELEG,DERIP,DERI2,DERIA,GEOMG,
     &            MMATG,UNKNG,IVARI)
C     WRITE (9,20001)
20001 FORMAT(' SECOND DERIVATIVES'/)
C     WRITE (9,20002) (IP,(COORG(IJ,IP),IJ=1,2),(DERI2(IJ,IP),
C    .                 IJ=1,4),IP=1,NPOIG)
20002 FORMAT(I8,2X,1P6E15.7)
C
C     -----FIND OUT PRINCIPAL DIRECTIONS AND VARIATIONS
C
      DO 1601 IPOIN=1,NPOIG
      AMEAN=0.5*(DERI2(1,IPOIN)+DERI2(4,IPOIN))
      ADEVI=0.5*(DERI2(1,IPOIN)-DERI2(4,IPOIN))
      TAUXY=DERI2(2,IPOIN)
      SI1=AMEAN+SQRT(ADEVI**2+TAUXY**2)
      SI2=AMEAN-SQRT(ADEVI**2+TAUXY**2)
      IF((ABS(SI1)+ABS(SI2)).LT.1.E-8) THEN
        HELP(1,IPOIN)=1.E-9
        HELP(2,IPOIN)=1.E-9
        HELP(3,IPOIN)=1.0
        HELP(4,IPOIN)=0.0
      ELSE
        IF(AMEAN.GE.0.0) THEN
          HELP(1,IPOIN)=ABS(SI1)
          HELP(2,IPOIN)=ABS(SI2)
          ALPHA=ATAN2(TAUXY,ADEVI)/2.
          HELP(3,IPOIN)=-SIN(ALPHA)
          HELP(4,IPOIN)= COS(ALPHA)
        ELSE
          HELP(1,IPOIN)=ABS(SI2)
          HELP(2,IPOIN)=ABS(SI1)
          ALPHA=ATAN2(TAUXY,ADEVI)/2.
          HELP(3,IPOIN)= COS(ALPHA)
          HELP(4,IPOIN)= SIN(ALPHA)
        ENDIF
      ENDIF
 1601 CONTINUE
C
C     -----SEARCH FOR THE MAXIMUM SPACING AND STRETCHING
C
      AMAX=0.0
```

```
      DO 1201 IPOIN=1,NPOIG
      IF(HELP(1,IPOIN).GT.AMAX) AMAX=HELP(1,IPOIN)
 1201 CONTINUE
C
      AMAXSQ=SQRT(AMAX)
c     RMIN=1.5
      DO 1202 IPOIN=1,NPOIG
c     X=COORG(1,IPOIN)
c     Y=COORG(2,IPOIN)
c     RAD=SQRT(X*X+Y*Y)
c     RMAX=1.7+0.1*ABS(Y)/RAD
c     DELRAJ=DELSCA+(RAD-RMIN)*3.0*DELSCA/(RMAX-RMIN)
      DELKM=DELKMMI*AMAXSQ/SQRT(AMAX1(1.E-9,HELP(1,IPOIN)))
      DELT2=DELKMMI*AMAXSQ/SQRT(AMAX1(1.E-9,HELP(2,IPOIN)))
      IF(DELKM.GT.DELKMMA) DELKM=DELKMMA
      IF(DELT2.GT.DELKMMA) DELT2=DELKMMA
      IF(DELKM.LT.DELSCA) DELKM=DELSCA
c     IF((RAD.LT.RMAX).AND.(DELRAJ.LT.DELKM)) DELKM=DELRAJ
      HELP(1,IPOIN)=DELKM
      HELP(2,IPOIN)=DELT2
 1202 CONTINUE
C
      DO 1302 IPOIN=1,NPOIG
      DELKM=HELP(1,IPOIN)
      DELTA(2,IPOIN)=AMIN1(DELTA(2,IPOIN),HELP(2,IPOIN))
      IF(DELKM.GE.DELTA(1,IPOIN)) GOTO 1302
      DELTA(1,IPOIN)=DELKM
      DELTA(3,IPOIN)=HELP(3,IPOIN)
      DELTA(4,IPOIN)=HELP(4,IPOIN)
 1302 CONTINUE
C
      IF(IBACK.EQ.1) GOTO 1209
C
      DO 1402 IPOIN=1,NPOIG
      DELKM=DELTA(1,IPOIN)
      DELT2=DELTA(2,IPOIN)
      STRE=DELT2/DELKM
c     X=COORG(1,IPOIN)
c     Y=COORG(2,IPOIN)
c     RAD=SQRT(X*X+Y*Y)
c     RMAX=1.7+0.1*ABS(Y)/RAD
c     DELRAJ=1.0+(RAD-RMIN)*2.0/(RMAX-RMIN)
c     IF(DELRAJ.LT.1.0) DELRAJ=1.0
c     IF(DELRAJ.GT.3.0) DELRAJ=3.0
      IF(STRE.GT.STR) STRE=STR
      IF(STRE.LT.1.0) STRE=1.0
c     IF((DELRAJ.GT.1.0).AND.(DELRAJ.LT.3.0)) STRE=DELRAJ
      DELTA(2,IPOIN)=STRE
 1402 CONTINUE
C     WRITE (9,20003)
20003 FORMAT(' DELTA',/)
C     WRITE (9,20002) (IP,(COORG(IJ,IP),IJ=1,2),(DELTA(IJ,IP),
C    .                 IJ=1,4),IP=1,NPOIG)
C
C *** SOME EXTRA REFINEMENT FOR STAGNATION POINTS
C
      WRITE (6,*) ' DO YOU WANT EXTRA REFINEMENT',
     .      ' FOR STAGNATION POINTS ?'
      WRITE(19,*) ' DO YOU WANT EXTRA REFINEMENT',
     .      ' FOR STAGNATION POINTS ?'
      READ (7,*)  ISTAG
      WRITE(6,*)  ISTAG
      WRITE(19,*)  ISTAG
      IF(ISTAG.EQ.0) GOTO 3420
      WRITE (6,*) ' ENTER DELKM, THRESHOLD MACH NUMBER AND GAMMA'
      WRITE(19,*) ' ENTER DELKM, THRESHOLD MACH NUMBER AND GAMMA'
      READ (7,*)  DELST,AMINI,GAMMA
      WRITE(6,*)  DELST,AMINI,GAMMA
      WRITE(19,*)  DELST,AMINI,GAMMA
      DO 3430 IPOIN=1,NPOIG
      RO=UNKNG(1,IPOIN)
      U1=UNKNG(2,IPOIN)
      U2=UNKNG(3,IPOIN)
      EN=UNKNG(4,IPOIN)
      VSQ=U1*U1+U2*U2
      PRE=(GAMMA-1.)*RO*(EN-0.5*VSQ)
      ROPOI=SQRT((VSQ*RO)/(GAMMA*PRE))
```

```
      IF(ROPOI.GT.AMINI) GOTO 3430
      DELKM=DELTA(1,IPOIN)
      DELT2=DELKM*DELTA(2,IPOIN)
      IF(DELKM.GT.DELST) DELKM=DELST
      DELTA(1,IPOIN)=DELKM
 3430 CONTINUE
C
 3420 CONTINUE
C
      RETURN
      END
C
C     ----------------------------------------------------------------
C
      SUBROUTINE GETGEO(NELEM,NPOIN,NNODE,NGEOM,
     &                  INTMAT,COORD,GEOME,NSTOP)
C
      REAL    COORD(2,NPOIN),GEOME(NGEOM,NELEM)
      REAL    X(3),Y(3),NXI(3),NET(3)
C
      INTEGER INTMAT(NNODE,NELEM)
C
      DATA  NXI/-1.0 , 1.0 , 0.0 /
      DATA  NET/-1.0 , 0.0 , 1.0 /
C
C     -----THIS SUB EVALUATES N,X & N,Y FOR EACH ELEMENT
C          SHAPE FUNCTION (LINEAR TRIANGLES) AND THE
C          JACOBIAN (=2*AREA)
C
C     -----LOOP OVER THE ELEMENTS
C
      DO 1000 IELEM=1,NELEM
C
C     ----PICK UP THE VALUES NEEDED
C
      DO 1001 INODE=1,NNODE
      IN=INTMAT(INODE,IELEM)
      X(INODE)=COORD(1,IN)
      Y(INODE)=COORD(2,IN)
 1001 CONTINUE
C
C     ----EVALUATE THE GEOMETRICAL QUANTITIES NEEDED
C
      X21=X(2)-X(1)
      X31=X(3)-X(1)
      Y21=Y(2)-Y(1)
      Y31=Y(3)-Y(1)
C
      RJ =X21*Y31-X31*Y21
      IF(RJ.GT.0.0) GOTO 2030
C     PRINT 12, IELEM
 12   FORMAT('  SOME ELEMENT HAVE GOT NEGATIVE AREA ',I5)
      NSTOP=1
      GO TO 1000
 2030 CONTINUE
      RJ1=1./RJ
C
      XIX= Y31*RJ1
      XIY=-X31*RJ1
      ETX=-Y21*RJ1
      ETY= X21*RJ1
C
C     ----FORM N,X & N,Y
C
      DO 1002 IN=1,3
      RNXI=NXI(IN)
      RNET=NET(IN)
      GEOME(IN  ,IELEM)=XIX*RNXI+ETX*RNET
      GEOME(IN+3,IELEM)=XIY*RNXI+ETY*RNET
 1002 CONTINUE
C
      GEOME(7,IELEM)=RJ
C
C     -----END OF LOOP OVER THE ELEMENTS
C
 1000 CONTINUE
      RETURN
```

```
      END
C
C     ------------------------------------------------
C
      SUBROUTINE GETVAR(NNODE,NAMAT,NGEOM,NDERV,NELEM,
     &                  NPOIN,COORD,INTMAT,DERIP,
     &                  DERI2,DERIA,GEOME,MMAT,UNKNO,
     &                  IVARI)
C
      REAL    UNKNO(NAMAT,NPOIN),MMAT(NPOIN)
      REAL    GEOME(NGEOM,NELEM),COORD(2,NPOIN)
      REAL    DERIP(NDERV,NPOIN),DERI2(4,NPOIN)
      REAL    DERIA(4,NPOIN)
      INTEGER INTMAT(NNODE,NELEM),NODE(3)
      INTEGER NX(4),NY(4)
C
C     -----INITIALIZE
C
      CALL RFILLM(DERIP,NDERV,NPOIN,0.0)
      CALL RFILLM(DERI2,4,NPOIN,0.0)
C
C     -----OBTAIN MMAT (ALREADY INVERTED)
C
      CALL GTMMAT(NELEM,NPOIN,NNODE,NGEOM,INTMAT,
     &            GEOME,MMAT)
C
C     ----- READ GAMMA
C
      IF(IVARI.LT.3) GOTO 22
      WRITE (6,*) ' ENTER THE VALUE OF GAMMA (USUALLY 1.4)'
      WRITE(19,*) ' ENTER THE VALUE OF GAMMA (USUALLY 1.4)'
      READ (7,*) GAMMA
      WRITE(6,*) GAMMA
      WRITE(19,*) GAMMA
   22 CONTINUE
C
C     -----LOOP OVER THE ELEMENTS
C
      DO 7000 IELEM=1,NELEM
C
      ROELX=0.0
      ROELY=0.0
C
C     -----OBTAIN THE VALUES NEEDED
C
      DO 7100 INODE=1,NNODE
      IPOIN=INTMAT(INODE,IELEM)
      RO=UNKNO(1,IPOIN)
      U1=UNKNO(2,IPOIN)
      U2=UNKNO(3,IPOIN)
      EN=UNKNO(4,IPOIN)
      VSQ=U1*U1+U2*U2
      PRE=(GAMMA-1.)*RO*(EN-0.5*VSQ)
      IF(IVARI.EQ.1) ROPOI=RO
      IF(IVARI.EQ.2) ROPOI=SQRT(VSQ)
      IF(IVARI.EQ.3) ROPOI=PRE/(RO**GAMMA)
      IF(IVARI.EQ.4) ROPOI=SQRT((VSQ*RO)/(GAMMA*PRE))
      RNX  =GEOME(INODE      ,IELEM)
      RNY  =GEOME(INODE+NNODE,IELEM)
      ROELX=ROELX+RNX*ROPOI
      ROELY=ROELY+RNY*ROPOI
      NODE(INODE)=IPOIN
      NX(INODE)=RNX
      NY(INODE)=RNY
 7100 CONTINUE
C
C     -----ELEMENT AREA
C
      RJ=GEOME(NGEOM,IELEM)*0.5/3.0
C
C     -----ASSEMBLY
C
      DO 1101 INODE=1,NNODE
      IPOIN=NODE(INODE)
      DERIP(1,IPOIN)=DERIP(1,IPOIN)+RJ*ROELX
      DERIP(2,IPOIN)=DERIP(2,IPOIN)+RJ*ROELY
 1101 CONTINUE
```

```
C
C     -----END OF LOOP OVER THE ELEMENTS
C
 7000 CONTINUE
C
C     -----MULTIPLY BY MMAT(INVERTED) AND DIVIDE BY THE NR. OF EL./PER NODE
C
       DO 7001 IPOIN=1,NPOIN
       DO 7002 J=1,2
       DERIP(J,IPOIN)=DERIP(J,IPOIN)*MMAT(IPOIN)
 7002  CONTINUE
 7001  CONTINUE
C
C     -----OBTAIN THE SECOND VARIATIONS
C
C
C     -----LOOP OVER THE ELEMENTS
C
       DO 8000 IELEM=1,NELEM
C
       ROEXX=0.0
       ROEXY=0.0
       ROEYX=0.0
       ROEYY=0.0
C
C     -----OBTAIN THE VALUES NEEDED
C
       DO 8100 INODE=1,NNODE
       IPOIN=INTMAT(INODE,IELEM)
       ROPOX=DERIP(1,IPOIN)
       ROPOY=DERIP(2,IPOIN)
       RNX  =GEOME(INODE       ,IELEM)
       RNY  =GEOME(INODE+NNODE,IELEM)
       ROEXX=ROEXX+RNX*ROPOX
       ROEXY=ROEXY+RNY*ROPOX
       ROEYX=ROEYX+RNX*ROPOY
       ROEYY=ROEYY+RNY*ROPOY
       NODE(INODE)=IPOIN
       NX(INODE)=RNX
       NY(INODE)=RNY
 8100 CONTINUE
C
C     -----ELEMENT AREA
C
      RJ=GEOME(NGEOM,IELEM)*0.5/3.0
C
C     -----ASSEMBLY
C
       DO 8101 INODE=1,NNODE
       IPOIN=NODE(INODE)
       DERI2(1,IPOIN)=DERI2(1,IPOIN)+RJ*ROEXX
       DERI2(2,IPOIN)=DERI2(2,IPOIN)+RJ*ROEXY
       DERI2(3,IPOIN)=DERI2(3,IPOIN)+RJ*ROEYX
       DERI2(4,IPOIN)=DERI2(4,IPOIN)+RJ*ROEYY
 8101  CONTINUE
C
C     -----END OF LOOP OVER THE ELEMENTS
C
 8000 CONTINUE
C
C     -----MULTIPLY BY MMAT(INVERTED) AND DIVIDE BY THE NR. OF EL./PER NODE
C
       DO 8001 IPOIN=1,NPOIN
       DO 8002 J=1,4
       DERI2(J,IPOIN)=DERI2(J,IPOIN)*MMAT(IPOIN)
 8002  CONTINUE
 8001  CONTINUE
C
C     DO SOME SMOOTHING IF REQUIRED
C
       WRITE (6,*) ' HOW MANY SMOOTHING LOOPS ?'
       WRITE(19,*) ' HOW MANY SMOOTHING LOOPS ?'
       READ (7,*) NSMOO
       WRITE(6,*) NSMOO
       WRITE(19,*) NSMOO
C
       IF(NSMOO.EQ.0) GOTO 7891
```

```
      DO 7890 IS=1,NSNOO
C
      CALL RFILLM(DERIA,4,NPOIN,0.0)
C
      DO 7900 IELEM=1,NELEM
      RJ=GEOME(NGEOM,IELEM)*0.5/3.
      R1=0.0
      R2=0.0
      R3=0.0
      R4=0.0
      DO 7901 INODE=1,NNODE
      IPOIN=INTMAT(INODE,IELEM)
      NODE(INODE)=IPOIN
      R1=R1+DERI2(1,IPOIN)*RJ
      R2=R2+DERI2(2,IPOIN)*RJ
      R3=R3+DERI2(3,IPOIN)*RJ
      R4=R4+DERI2(4,IPOIN)*RJ
 7901 CONTINUE
C
C     DISTRIBUTE
C
      DO 7902 INODE=1,NNODE
      IPOIN=NODE(INODE)
      DERIA(1,IPOIN)=DERIA(1,IPOIN)+R1
      DERIA(2,IPOIN)=DERIA(2,IPOIN)+R2
      DERIA(3,IPOIN)=DERIA(3,IPOIN)+R3
      DERIA(4,IPOIN)=DERIA(4,IPOIN)+R4
 7902 CONTINUE
 7900 CONTINUE
C
C     -----MULTIPLY BY MMAT(INVERTED) AND DIVIDE BY THE NR. OF EL./PER NODE
C
      DO 9001 IPOIN=1,NPOIN
      DO 9002 J=1,4
      DERI2(J,IPOIN)=DERIA(J,IPOIN)*MMAT(IPOIN)
 9002 CONTINUE
 9001 CONTINUE
C
 7890 CONTINUE
 7891 CONTINUE
C
C     SYMMETRIZE !!!!!! (EQUATE CROSS DERIVATIVES)
C
      DO 8060 IPOIN=1,NPOIN
      TAUXY=(DERI2(2,IPOIN)+DERI2(3,IPOIN))/2.
      DERI2(2,IPOIN)=TAUXY
      DERI2(3,IPOIN)=TAUXY
 8060 CONTINUE
C
      RETURN
      END
C
C     ---------------------------------------------------------------
C
      SUBROUTINE GTMMAT(NELEM,NPOIN,NNODE,NGEOM,
     &                  INTMAT,GEOME,MMAT)
C
      REAL GEOME(NGEOM,NELEM),MMAT(NPOIN)
      INTEGER INTMAT(NNODE,NELEM)
C       INTEGER NODEN(9)
C
      C6=1./6.
C
C     ----ZERO MMAT
C
      CALL RFILLV(MMAT,NPOIN,0.0)
C
C     ----LOOP OVER THE ELEMENTS
C
      DO 1000 IELEM=1,NELEM
C
C     ----JACOBIAN OF THE ELEMENT
C
      RJ =GEOME(7,IELEM)
      RJ6=RJ*C6
C      WRITE (19,*) 'IELEM ',IELEM,RJ
C
```

```
C      ----ADD TO MMAT
C
        DO 1020 INODE=1,NNODE
        IN=INTMAT(INODE,IELEM)
        MMAT(IN)=MMAT(IN)+RJ6
C        WRITE (19,*) 'INODE ',INODE,IN,MMAT(IN)
 1020 CONTINUE
C
C      ----END OF LOOP OVER THE ELEMENTS
C
 1000 CONTINUE
C
C      ----CONTROL OUTPUT OF MMAT
C
C       WRITE(6,5)
C       WRITE(19,*)NPOIN
C       WRITE(19,6)(MMAT(I),I=1,NPOIN)
C    5 FORMAT(//,' OUTPUT OF MMAT BEFORE INVERSION',//)
     6 FORMAT(1P10E15.7)
C
C      ----INVERSION OF MMAT
C
        DO 2000 I=1,NPOIN
        MMAT(I)=1./MMAT(I)
C        WRITE (19,*) 'POIN  ',I,MMAT(I)
 2000 CONTINUE
C
        RETURN
        END
C
C       ------------------------------------------------------
C
        SUBROUTINE ADDMAT(A,B,C,N,M)
C
        REAL A(N,M),B(N,M),C(N,M)
C
C      ----MATRIX A + MATRIX B ==> MATRIX C
C
        DO 1000 J=1,M
        DO 2000 I=1,N
        C(I,J)=A(I,J)+B(I,J)
 2000 CONTINUE
 1000 CONTINUE
      RETURN
      END
C
C    ------------------------------------------------------
C
      SUBROUTINE RFILLM(A,N,M,VALUE)
C
      REAL A(N,M)
C
      DO 1000 I=1,N
      DO 1000 J=1,M
 1000 A(I,J)=VALUE
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      SUBROUTINE RFILLIV(IA,N,IVALUE)
C
      INTEGER IA(N)
C
      DO 1000 I=1,N
 1000 IA(I)=IVALUE
C
      RETURN
      END
C
C      ------------------------------------------------------------
C
       SUBROUTINE RFILLV(A,NA,C)
       REAL A(NA)
       DO 1000 I=1,NA
       A(I)=C
```

```
 1000 CONTINUE
      RETURN
      END
c     SUBROUTINE WHOAMI
c     ENTRY JBEGIN
c     ENTRY JDINIT
c     ENTRY JDEVON
c     ENTRY JWINDO
c     ENTRY JOPEN
c     ENTRY JCMARK
c     ENTRY JMARK
c     ENTRY JCLOSE
c     ENTRY JPAUSE
c     ENTRY JDEVOF
c     ENTRY JDEND
c     ENTRY JEND
c     ENTRY JMOVE
c     ENTRY JDRAW
c     RETURN
C     END
C
C     ------------------------------------------------------
C
      REAL FUNCTION FXTR(XQ,YQ,AX,AY,ALPH)
C
      FXTR = AX*ALPH*(AX*XQ+AY*YQ)+AY*(AY*XQ-AX*YQ)
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      REAL FUNCTION FYTR(XQ,YQ,AX,AY,ALPH)
C
      FYTR = AY*ALPH*(AX*XQ+AY*YQ)-AX*(AY*XQ-AX*YQ)
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      REAL FUNCTION FXBA(XQ,YQ,AX,AY,ALPH)
C
      FXBA = (AX*(AX*XQ+AY*YQ)/ALPH)+AY*(AY*XQ-AX*YQ)
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      REAL FUNCTION FYBA(XQ,YQ,AX,AY,ALPH)
C
      FYBA = (AY*(AX*XQ+AY*YQ)/ALPH)-AX*(AY*XQ-AX*YQ)
C
      RETURN
      END
C
C     ------------------------------------------------------
C
      REAL FUNCTION DETER(P1,Q1,P2,Q2,P3,Q3)
C
      DETER = P2*Q3-P3*Q2-P1*Q3+P3*Q1+P1*Q2-P2*Q1
C
      RETURN
      END
C     ------------------------------------------------------
```

# ประวัติผู้วิจัย

นายปัญญา จันทร์ไพแสง เกิดวันที่ 10 กันยายน พ.ศ. 2516 ที่อำเภอชุมแสง จังหวัดนครสวรรค์ สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมเครื่องกล ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2537 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมเครื่องกล ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2538