รายการอ้างอิง

1. Anderson, J. D. Jr. Computational Fluid Dynamics. Singapore : McGrawHill Inc., 1995.

2. Incropera, F. P., and De Witt, D. P. Fundamentals of Heat and Mass Transfer. 4th ed. Singapore : John Wiley & Sons, 1996.

3. Timoshenko, S. P. and Goodier, J. N. Theory of Elasticity. 3rd ed. Singapore : McGrawHill International Editions.

4. Gnoffo, P. A. Application of program LAURA to three-dimensional AOTV flowfields. AIAA-86-0565 (January 1986) : 1-12.

5. ปราโมทย์ เดชะอำไพ. ไฟไนต์เอลิเมนต์ในงานวิศวกรรม. พิมพ์ครั้งที่ 2 กรุงเทพฯ : สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, 2542.

6. ปัญญา จันท์ไพแสง. ระเบียบวิธีไฟไนต์เอลิเมนต์สำหรับการไหลความเร็วสูงแบบอัดตัวได้. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2541.

7. Wylen, G. V.; Sonntag, R.; Borgnakke, C. Fundamentals of classical Thermodynamics. 4th ed. Singapore : John Wiley & Sons, 1997.

8. Hirsch, C. Numerical Computation of Internal and External Flows. 1 & 2 (1988) Singapore : John Wiley & Sons.

9. Wendt, J. F. Computational Fluid Dynamics An Introduction. Springer-Verlag, 1992.

10. Lapidus, A. A detached shock calculation by second-order finite difference. Journal of Computational Physics. 2 (1967) : 154-177.

11. Tannehill, J. C.; Holst, T.L.; and Rakich, J.V. Numerical computation of two-dimensional viscous blunt body flows with an Impinging Shock. AIAA Paper 75-154 (1975), Pasadena, California.

12. White, F. M. Viscous Fluid Flow. 2nd ed. New York : McGraw-Hill, 1991.

13. Huebner, K. H.; Thornton, E. A.; Byrom, T. G. The Finite Element Method for Engineers. 3rd ed. New York : John Wiley & Sons, 1995.

14. Anderson, D. A.; Tannehill, J. C.; and Pletcher, R. H. Computational Fluid Mechanics and Heat Transfer. Newyork : McGraw-Hill, 1984.

15. Schlichting, H. Boundary-layer theory. 7th ed. New York : McGraw-Hill, 1979.

16. Billig, F. S. Shock-Wave Shapes around Spherical- and Cylindrical-Nosed Bodies. Journal of Spacecraft 4, No. 6 (June 1967) : 822-823.

17. สุพัฒนพงศ์ สิขาบัณฑิต. เทคนิคการปรับขนาดไฟไนต์เอลิเมนต์เพื่อการวิเคราะห์การไหลแบบหนืด. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2541.

18. Hodge, B. K., and Koenig, K. Compressible fluid dynamics with personal computer application. Prentice-Hall International, 1995.

19. Myers, G. E. Analytical methods in conduction heat transfer. McGraw-Hill, 1971.

20. Wieting, A. R. Experimental study of shock wave interference heating on a cylindrical leading edge. NASA TM-100484 May 1987.

21. ปราโมทย์ เดชะอำไพ. ระเบียบวิธีไฟไนต์เอลิเมนต์ เพื่อการคำนวณพลศาสตร์ของไหล. กรุงเทพ ฯ : สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, 2544.

22. Gnoffo, P. A. CFD validation studies for hypersonic flow prediction. AIAA-01-1025 ,2001.

23. Zienkiewicz, O. C. and Taylor, R. L. The finite element method. 5th ed. Butterworth-Heinemann, 2000.

24. Dechaumphai, P. and Limtrakarn, W. Adaptive cell-centered finite element technique for compressible flows. Journal of energy, heat and mass transfer 21, No. 2 (1999) : 57–65.

25. Limtrakarn, W. and Dechaumphai, P. Computations of high-speed compressible flows with adaptive cell–centered finite element method. Journal of the Chinese institute engineers 26, 2003.

26. Limtrakarn, W. and Dechaumphai, P. An explicit finite element method for inviscid compressible flow. ME-NETT 15, 2001.

27. Dechaumphai, P. Improved Finite Element Methodology for Integrated Thermal-Structure Analysis. NASA CR 3635, 1982.

28. Dechaumphai, P. Adaptive Finite Element Technique for Heat Transfer Problems. Journal of Energy, Heat & Mass Transfer. 17, No. 2, 1995.

29. Dechaumphai, P. Evaluation of an adaptive unstructured remeshing technique for integrated fluid-thermal-structural analysis. J. Thermophysics 5, No. 4, 1991.

30. Thornton, E.A. and Dechaumphai, P. Coupled flow, thermal, and structural analyses of aerodynamically heated panels. Journal of Aircraft 25, No. 11, 1988.

31. Moss, J.N. DSMC simulations of shock interactions about sharp double cones. NASA TM-210318, 2000.

32. Thibert, J.J. An overview of supersonic aerodynamics research at ONERA. ECCOMASS 2000, 2000.

33. Gnoffo, P.A. Computational aerodynamics in aeroassist applications. AIAA-01-2632, 2001.

34. Walpot, L. Grid convergence studies on laminar shock wave boundary layer interaction. ECCOMAS 2000, 2000.

35. Chanetz, B. and et al. Experimental and numerical study of the laminar separation in hypersonic flow. ECCOMAS 2000, 2000.

36. Lohner, R., Morgan, K., and Zienkiewicz, O.C. Adaptive grid refinement for compressible euler and navier-stokes equations. The international conference on accuracy estimates and adaptive refinements in finite element computations, 1984.

37. Lohner, R. and et al. Fluid-structure-thermal interaction using a loose coupling algorithm and adaptive unstructured grids. AIAA-98-2419, 1998.

38. Morishita, E. Spreadsheet fluid dynamics of a blunt body problem. JSME International journal series B 45, No. 4, 2002.

39. Kennedy, C.A. and Carpenter, M.H. Comparison of several numerical methods for simulation of compressible shear layers. NASA TP-3484, 1997.

40. Johnston, I.A. Simulation of flow around hypersonic blunt-nosed vehicles for the calibration of air data systems. Ph.D. dissertation, University of Queensland, 1999.

41. Bonhaus, D.L. A higher order accurate finite element method for viscous compressible flows. Ph.D. dissertation, Virginia polytechnic institute, 1998.

42. Giraldo, F.X. A space marching adaptive remeshing technique applied to the 3D euler equations for supersonic flow. Ph.D. dissertation, University of Virginia, 1995.

43. Anderson, J.D. Jr. Modern compressible flow with historical prospective. New York : McGrawHill Inc., 1982.

44. Anderson, J.D. Jr. Hypersonic and high temperature gas dynamics. Singapore : McGrawHill Inc., 1989.

45. Bertin, J.J. Hypersonic aerothermodynamics. AIAA education series, 1994.

46. Lohner, R. Applied computational fluid dynamics techniques. John Wiley & Sons, 2001.

47. Oden, J. T. and Carey, G. F., 1981, Finite Elements: Mathematical Aspects. Prentice-Hall, Englewood Cliffs, New Jersey.

48. Toro, P.G.P. and et al. Self-similar compressible laminar boundary layers. AIAA-97-0767, 1997.

49. Power, G.G. and Barber, T.J. Analysis of complex hypersonic flows with strong viscous/inviscid interaction. AIAA-paper 26, No. 7, 1988.

50. Usab, W.J. Jr. and Murman, E.M. Embedded mesh solutions of the euler equation using a multiple-grid method. AIAA paper 83-1946.

51. Lohner, R., Morgan, K., Peraire, J. and Vahdati, M. Finite element flux-corrected transport (FEM-FCT) for the euler and navier-stokes equations. ICASE report, No. 87-4.

52. Zienkiewicz, O.C., Lohner, R., and Morgan, K. High-speed inviscid compressible flow by the finite element method. NASA-CR-173556, 1984.

53. Donea, J. A taylor-galerkin method for convective transport problems. International journal for numerical method in engineers 20, 1984.

54. Peraire, J., Vahdati, M., Morgan, K., and Zienkiewicz, O.C. Adaptive remeshing for compressible flow computation. Journal of computational physics 72, 1987.

55. Ramakrishnan, R., Bey, K.S., and Thornton, E.A. Adaptive quadrilateral and triangular finite element scheme for compressible flows. AIAA journal 1, 1990.

56. Hung, C.M. and MacCormack, R.W. Numerical solutions of supersonic and hypersonic laminar compression corner flows. AIAA journal 14, No. 4, 1976

57. Simeonides, G. and Haase, W. Experimental and computational investigations of hypersonic flow about compression ramps. J. Fluid Mech. 283, 1995.

58. Zoby, E.V. and Thompson, R.A. Flowfield and vehicle parameter influence on hypersonic heat transfer and drag. J. Spacecraft 27, No, 4, 1990.

59. Davis, R.T., Barnett, M. and Rakich, J.V. The calculation of supersonic viscous flows using the parabolized navier-stokes equations. Computers & Fluids 14, No. 3, 1986.

60. Peroomian, O., Chakravarthy, S. and Goldberg, U.C. A grid-transparent methodlogy for CFD. AIAA paper, 1996.

61. Grondin, G. and Thivet, F. Computation of laminar supersonic flows with conservative multi-domain technique for steady parabolized navier-stokes equations. ECCOMAS 98, 1998.

62. Giraldo, F. A finite volume high resolution 2D euler solver with adaptive grid generation on high performance computers. Ninth international conference on finite elements in fluids, 1995.

63. Brueckner, F.P. and Heinrich, J.C. Petrov-Galerkin finite element model for compressible flows. International journal for numerical methods in engineering 32, 1991.

64. Thareja, R.R., Stewart, J.R., Hassan, O., Morgan, K., and Peraire, J. A point implicit unstructured grid solver for the euler and navier-stokes equations. International journal for numerical methods in fluids 9, 1989.

65. Zuber, M.E. Flowfield structure for crossing shocks intersecting a Mach 6 laminar boundary layer. AIAA paper 2000-4515, 2000.

66. Harvey, J.K., Holden, M.S., and Wadhams, T.P. Code Validation Study of Laminar Shock/Boundary Layer and Shock/Shock Interactions in Hypersonic Flow. Part B: Comparisons with Navier-Stokes and DSMC Solutions. AIAA Paper 2001-1031, 2001.

67. Holden, M.S., Wadhams, T.P., Harvey, J.K., and Candler, G.V. Comparisons Between DSMC and Navier-Stokes Solutions on Measurements in Regions of Laminar Shock Wave Boundary Layer Interaction in Hypersonic Flows. AIAA paper 2002-0435, 2002.

68. Holden, M.S. and Wadhams, T.P. A Database of Aerothermal Measurements in Hypersonic Flows in 'Building Block' Experiments for CFD Validation. AIAA Paper 2003-1137, 2003.

ภาคผนวก

ภาคผนวก ก

รายละเอียดของโปรแกรมคอมพิวเตอร์ CELLHIFLOW

โปรแกรมคอมพิวเตอร์ CELLHIFLOW ที่ได้ประดิษฐ์ขึ้นเพื่อวิเคราะห์ปัญหาที่มีการไหลความเร็วสูงทั้งแบบหนืดและไม่หนืดดังที่ได้กล่าวไว้ในบทที่ 5 มีรายละเอียดดังนี้

```
C     PROGRAM CELLHIFLOW
C
C     A FINITE ELEMENT COMPUTER PROGRAM FOR SOLVING THE N-S EQUATIONS
C     FOR TWO-DIMENSIONAL VISCOUS COMPRESSIBLE HIGH SPEED FLOW.
C
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 ELEMENT SIDES IN THE MODEL
C              >= (3*NELEM+NBOUN)/2
C
      USE MSFLIB
      PARAMETER (MXPOI=1000, MXELE=2000, MXBOU=200, MXSID=5000)
      PARAMETER (NNODE=4, NAMAT=4)
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 AREA(MXELE), SLEN(MXSID,2), SUMSQ(4), ERRU(4)
      DIMENSION SIDERX(MXPOI,3), SIDERY(MXPOI,3), AMLP(MXPOI)
      DIMENSION DNDX(MXELE,4), DNDY(MXELE,4), SUMSQ1(4)
C
      INTEGER INTMAT(MXELE,NNODE), BSIDO(MXBOU,4)
      INTEGER ISIDE(MXSID,4), JESID(MXELE,4)
C
      INTEGER(2) tmpday, tmpmonth, tmpyear
      INTEGER(2) tmphour, tmpminute, tmpsecond, tmphund
C
      CHARACTER NAME1*20, NAME2, NAME3, NAME4, CW*4
C
C     INPUT FILE NAME AND VERSION
C
   10 WRITE(6,20)
   20 FORMAT(/,'PLEASE ENTER THE INPUT FILE NAME:')
      READ(5,'(A)',ERR=10) NAME1
      L = NAMLEN(NAME1)
      IF (L.EQ.0) GO TO 10
      WRITE(6,'(A,$)') ' CURRENT VERSION : '
      READ(5,'(A)') CW
C
C     UNIT  7 ==> DATA FILE
C     UNIT 25 ==> ERROR NORM PLOT & CPU TIME
C
      OPEN(UNIT=7,FILE=NAME1(1:L)//'.DD'//CW,STATUS='OLD',ERR=10)
      OPEN(UNIT=24,FILE=NAME1(1:L)//'.X'//CW,STATUS='UNKNOWN')
      OPEN(UNIT=25,FILE=NAME1(1:L)//'.C'//CW,STATUS='UNKNOWN')
C
      WRITE(6,2)
      WRITE(25,2)
      CALL GETDAT(tmpyear, tmpmonth, tmpday)
      CALL GETTIM(tmphour, tmpminute, tmpsecond, tmphund)
```

```
      WRITE(6,3) tmpday, tmpmonth, tmpyear
      WRITE(6,4) tmphour, tmpminute, tmpsecond, tmphund
      WRITE(25,3) tmpday, tmpmonth, tmpyear
      WRITE(25,4) tmphour, tmpminute, tmpsecond, tmphund
      WRITE(25,*)
    2 FORMAT(/,' START TIME :')
    3 FORMAT(3X,I2.2,'/',I2,'/',I4.4)
    4 FORMAT(3X,I2,':',I2.2,':',I2.2,':',I2.2)
    6 FORMAT(/,' STOP TIME :')
    7 FORMAT(I6,5E12.5,2X,I2,':',I2.2,':',I2.2,':',I2.2)
C
C
C     READ INPUT DATA AND ADJUST TO ELEMENT QUANTITIES
C
C
      CALL GETNPUT(MXELE, MXPOI, MXBOU, NNODE, NELEM, NPOIN,
     *             NBOUN, GAMMA, EPSLAM, CSAFE, DT, NITER,
     *             NERR, NPLOT, INTMAT, COORD, UNKNP, BSIDO,
     *             IVISC, IDISS, CV, AMUF, TREFF, S, PR, CSAFV,
     *             NTRI, NQUAD)
C
C     COMPUTE TOTAL NUMBER OF SIDES AND CONVERGE CRITERIA
C
      NSIDE = (3*(NTRI + 2*NQUAD) + NBOUN)/2 - NQUAD
      NORD = 6
      NSHOW = NPLOT
      TOL = 1.
      DO 30 I=1,NORD
      TOL = 0.1*TOL
   30 CONTINUE
C
      IF(NSIDE.GT.MXSID)  WRITE(6,50)  NSIDE
   50 FORMAT(' PLEASE INCREASE MXSID TO', I6)
      IF(NSIDE.GT.MXSID)  STOP
C
      WRITE(6,55)
   55 FORMAT(/, '  THE FINITE ELEMENT MODEL CONSISTS OF:')
      WRITE(6,60) NPOIN
   60 FORMAT('      NUMBER OF NODES                  =', I12)
      WRITE(6,70) NELEM
   70 FORMAT('      NUMBER OF ELEMENTS               =', I12)
      WRITE(6,80) NBOUN
   80 FORMAT('      NUMBER OF BOUNDARY CONDITIONS    =', I12)
      WRITE(6,90) NITER
   90 FORMAT('      NUMBER OF ITERATIONS             =', I12,/)
C
C     SAVE THESE NODAL INITIAL CONDITIONS
C
      DO 100  J=1,4
      DO 100  I=1,NPOIN
      UNKNP1(I,J) = UNKNP(I,J)
  100 CONTINUE
C
C     OBTAIN NODAL CONSERVATION VARIABLES
C
      DO 150  J=2,4
      DO 150  I=1,NPOIN
      UNKNP(I,J) = UNKNP(I,1)*UNKNP(I,J)
  150 CONTINUE
C
C     COMPUTE ELEMENT QUANTITIES
C
      C13 = 1./3.
      DO 200  IE=1,NELEM
      NCOUNT = 4
      FACTOR = 0.25
      IL = INTMAT(IE,4)
      IF(IL.EQ.0) NCOUNT = 3
```

```
      IF(IL.EQ.0) FACTOR = C13
      DO 200  IA=1,NAMAT
      UNKNO(IE,IA) = 0.
      DO 200  IN=1,NCOUNT
      IP = INTMAT(IE,IN)
      UNKNO(IE,IA) = UNKNO(IE,IA) + FACTOR*UNKNP(IP,IA)
  200 CONTINUE
C
C     IDENTIFY THE SIDE: DETERMINE ARRAYS ISIDE(NSIDE,4),
C     JESID(NELEM,4) AND RSIDO(NSIDE,3)
C
      CALL GETSIDE(MXELE, MXPOI, MXBOU, MXSID, NNODE, NELEM,
     *             NPOIN, NBOUN, NSIDE, INTMAT, COORD, BSIDO,
     *             ISIDE, JESID, RSIDO)
C
C     COMPUTE ELEMENT AREAS
C
      CALL GETMAT(MXELE, MXPOI, NELEM, INTMAT, COORD,
     *            NPOIN, DNDX, DNDY, AREA, AMLP)
C
C     DETERMINE REPRESENTATIVE ELEMENT LENGTHS FOR LOCAL TIME STEP
C     COMPUTATION
C
      CALL GETLEN(MXELE, MXPOI, MXSID, NNODE, NELEM, INTMAT,
     *            COORD, ISIDE, JESID, SLEN)
  205 CONTINUE
C
      ISHOW = 0
      IERR  = 0
      IPLOT = 0
      WRITE(6,400)
  400 FORMAT('  PERFORMING ITERATION COMPUTATION ')
      WRITE(6,*)
      WRITE(6,450)
  450 FORMAT(2X,'ITER',5X,'DENSITY',6X,'U-VELOCITY',4X,'V-VELOCITY',
     *       2X,'TOTAL ENERGY')
C
C     ENTER ITERATION LOOP:
C
      DO 650  ITER=1,NITER
C
C     STORE ELEMENT UNKNOWNS OF PREVIOUS ITERATION IN UNKN1(NELEM,4)
C
      DO 500  IA=1,4
      DO 500  IE=1,NELEM
      UNKN1(IE,IA) = UNKNO(IE,IA)
  500 CONTINUE
C
C     COMPUTE NODAL DERIVATIVES OF  U, V, T  WRT.  X & Y
C
      IF(IVISC.EQ.1)
     *CALL NDER(MXPOI, MXELE, MXBOU, NNODE, NAMAT, NBOUN,
     *    NPOIN, NTRI, NQUAD, NELEM, INTMAT, COORD, UNKNO,
     *    BSIDO, GAMMA,    CV, UNKNP,  AREA,  DNDX,  DNDY,
     *    AMLP, SIDERX, SIDERY                          )
C
      CALL COMPUTE(MXELE, MXPOI, MXSID, NELEM, NSIDE, UNKNP,
     *             GAMMA, EPSLAM, CSAFE, DT, ISIDE, RSIDO,
     *             AREA, SLEN, UNKN1, UNKNO,
     *             NNODE, INTMAT, IVISC, IDISS, SIDERX, SIDERY,
     *             PR, CSAFV, TREFF, S, AMUF, CV)
C
C     CHECK FOR CONVERGENCE:
C
C
      DO 550  IA=1,4
      SUMSQ(IA)  = 0.
      SUMSQ1(IA) = 0.
```

```
      DO 600  IE=1,NELEM
      DIFF = UNKNO(IE,IA) - UNKN1(IE,IA)
      IF(UNKNO(IE,IA).NE.0.) DIFF1 = DIFF/UNKNO(IE,IA)
      IF(UNKNO(IE,IA).EQ.0.) DIFF1 = 0.
      SUMSQ(IA) = SUMSQ(IA) + DIFF*DIFF
      SUMSQ1(IA) = SUMSQ1(IA) + DIFF1*DIFF1
  600 CONTINUE
      SUMSQ(IA)  = SQRT(SUMSQ(IA))
      SUMSQ1(IA) = SQRT(SUMSQ1(IA))
      IF(ITER.EQ.1) ERRU(IA) = SUMSQ(IA)
  550 CONTINUE
C
      IF(SUMSQ(1).LT.ERRU(1)*TOL) ISHOW = ITER
      IF(SUMSQ(1).LT.ERRU(1)*TOL) IERR  = ITER
C
      IF(ITER.EQ.1) THEN
         WRITE(6,620)  ITER, (SUMSQ(IA), IA=1,4), SUMSQ1(1)
          WRITE(25,620) ITER, (SUMSQ(IA), IA=1,4), SUMSQ1(1)
         ISHOW = NSHOW
         IERR  = NERR
         IPLOT = NPLOT
      ENDIF
  620 FORMAT(I6, 5(2X,E12.5))
C
      IF(SUMSQ(1).LT.ERRU(1)*TOL)
     * WRITE(25,620) ITER, (SUMSQ(IA), IA=1,4), SUMSQ1(1)
      IF(SUMSQ(1).LT.ERRU(1)*TOL)
     * WRITE(6,620) ITER, (SUMSQ(IA), IA=1,4), SUMSQ1(1)
      IF(SUMSQ(1).LT.ERRU(1)*TOL) GOTO 700
C
      IF(ITER.EQ.ISHOW) WRITE(6,620)  ITER,(SUMSQ(IA),IA=1,4), SUMSQ1(1)
      IF(ITER.EQ.ISHOW) ISHOW = ISHOW + NSHOW
      IF(IERR.EQ.ITER) WRITE(25,620) ITER, (SUMSQ(IA),IA=1,4), SUMSQ1(1)
      IF(IERR.EQ.ITER) IERR  = IERR  + NERR
      IF(IPLOT.EQ.ITER) IPLOT = IPLOT + NPLOT
C
  650 CONTINUE
C
  700 CONTINUE
C
C     PRINT OUT SOLUTIONS OF DENSITY, U&V VELOCITY AND TOTAL ENERGY
C
  660 WRITE(6,670)
  670 FORMAT(/,'PLEASE ENTER FILE NAME FOR DEN-U-V-TE SOLUTIONS:')
      READ(5,'(A)') NAME2
      OPEN(UNIT= 8,FILE=NAME2,STATUS='NEW', ERR=660)
C
      CALL GETPLOT(MXELE, MXPOI, MXBOU, NNODE, NELEM, NPOIN,
     *            NBOUN, INTMAT, COORD, BSIDO, UNKNP, UNKNP1,
     *            UNKNO)
C
      CALL GETDAT(tmpyear, tmpmonth, tmpday)
      CALL GETTIM(tmphour, tmpminute, tmpsecond, tmphund)
      WRITE(6,6)
      WRITE(6,3) tmpday, tmpmonth, tmpyear
      WRITE(6,4) tmphour, tmpminute, tmpsecond, tmphund
      WRITE(25,6)
      WRITE(25,3) tmpday, tmpmonth, tmpyear
      WRITE(25,4) tmphour, tmpminute, tmpsecond, tmphund
C
      STOP
      END
```

```
C
C-----------------------------------------------------------------------
C
C     [NAMLEN] COUNTS THE NUMBER OF CHARACTERS IN FILNAM.
C
      INTEGER FUNCTION NAMLEN(FILNAM)
C
      CHARACTER*20 FILNAM
C
      NAMLEN = 0
      DO 10 I = 20,1,-1
      IF (FILNAM(I:I).EQ.' ') GO TO 10
      NAMLEN = I
      GO TO 20
   10 CONTINUE
   20 RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE GETNPUT(MXELE, MXPOI, MXBOU, NNODE, NELEM, NPOIN,
     *                   NBOUN, GAMMA, EPSLAM, CSAFE, DT, NITER,
     *                   NERR, NPLOT, INTMAT, COORD, UNKNP, BSIDO,
     *                   IVISC, IDISS, CV, AMUF, TREFF, S, PR, CSAFV,
     *                   NTRI, NQUAD)
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION COORD(MXPOI,2), UNKNP(MXPOI,4)
C
      INTEGER INTMAT(MXELE,NNODE), BSIDO(MXBOU,4)
C
C     READ TITLE OF COMPUTATION:
C
      READ(7,*) NLINES
      DO 5 ILINE=1,NLINES
      READ(7,1) TEXT
    5 CONTINUE
C
C     READ INPUT DATA:
C
      READ(7,1)  TEXT
    1 FORMAT(10A8)
      READ(7,*)  NTRI, NQUAD, NPOIN, NBOUN, NITER, NERR, NPLOT
      NELEM = NTRI + NQUAD
C
      IF(NPOIN.GT.MXPOI)  WRITE(6,2)  NPOIN
    2 FORMAT(' PLEASE INCREASE MXPOI TO', I6)
      IF(NPOIN.GT.MXPOI)  STOP
      IF(NELEM.GT.MXELE)  WRITE(6,3)  NELEM
    3 FORMAT(' PLEASE INCREASE MXELE TO', I6)
      IF(NELEM.GT.MXELE)  STOP
      IF(NBOUN.GT.MXBOU)  WRITE(6,4)  NBOUN
    4 FORMAT(' PLEASE INCREASE MXBOU TO', I6)
      IF(NBOUN.GT.MXBOU)  STOP
C
      EPSLAM = 0.5
      CSAFE  = 0.5
C
C     READ IVISC, GAMMA, CV, AMUF, TREFF, S, PR, DELT:
C
      READ(7,1)  TEXT
      READ(7,*)  IVISC, GAMMA, CV, AMUF, TREFF, S, PR, DT
C
C     READ ELEMENT NODAL COORDINATES AND INITIAL CONDITIONS:
C
      READ(7,1)  TEXT
      DO 10  I=1,NPOIN
      READ(7,*)  N, (COORD(I,J), J=1,2), (UNKNP(I,J), J=1,4)
```

```
   10 CONTINUE
C
C     READ  ELEMENT NODAL CONNECTIONS :
C
      READ(7,1)  TEXT
      DO 40  I=1,NELEM
      READ(7,*)  IE, (INTMAT(I,J), J=1,NNODE)
      IF(I.NE.IE) WRITE(6,20) IE
   20 FORMAT(/, ' ELEMENT NO.', I5,' IN DATA FILE IS MISSING')
      IF(I.NE.IE) STOP
   40 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. NO. OF THAT SIDE
C                              = 1 : INFLOW  CONDITION
C                              = 2 : OUTFLOW CONDITION
C                              = 3 : INVISCID WALL CONDITION
C                              = 4 : VISCOUS  WALL CONDITION
C
      READ(7,1)  TEXT
      DO 70  IB=1,NBOUN
      READ(7,*)  (BSIDO(IB,J), J=1,4)
   70 CONTINUE
C
      RETURN
      END
C
C--------------------------------------------------------------------
C
      SUBROUTINE GETSIDE(MXELE, MXPOI, MXBOU, MXSID, NNODE, NELEM,
     *                   NPOIN, NBOUN, NSIDE, INTMAT, COORD, BSIDO,
     *                   ISIDE, JESID, RSIDO)
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION ISIDE(MXSID,4), COORD(MXPOI,2)
      DIMENSION LWHER(NPOIN), LHOWM(NPOIN), ICONE(4*NELEM)
      DIMENSION JESID(MXELE,NNODE), RSIDO(MXSID,3)
C
      INTEGER BSIDO(MXBOU,4), INTMAT(MXELE,NNODE)
C
C     FILL IN LHOWM: NO. OF ELEMENTS PER NODE
C
      DO 110 IS=1,MXSID
      DO 100 I=1,4
      ISIDE(IS,I) = 0
  100 CONTINUE
      DO 110 I=1,3
      RSIDO(IS,I) = 0.
  110 CONTINUE
C
      DO 115 IP=1,NPOIN
      LHOWM(IP)=0
  115 CONTINUE
C
      DO 120 IE=1,NELEM
      IL = INTMAT(IE,4)
      IF(IL.EQ.0) NCOUNT=3
      IF(IL.NE.0) NCOUNT=4
      DO 120 IN=1,NCOUNT
      IP=INTMAT(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
      IL = INTMAT(IE,4)
      IF(IL.EQ.0) NCOUNT=3
      IF(IL.NE.0) NCOUNT=4
      DO 135 IN=1,NCOUNT
      IP=INTMAT(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)
      IL = INTMAT(IE,4)
      IF(IL.EQ.0) NCOUNT=3
      IF(IL.NE.0) NCOUNT=4
C
C     FIND OUT POSITION OF IP IN THE CONECTIVITY MATRIX
C
      DO 150 IN=1,NCOUNT
      IN1=IN
      IPT=INTMAT(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=INTMAT(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
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
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)
      IL = INTMAT(IEL,4)
      IF(IL.EQ.0) NCOUNT=3
      IF(IL.NE.0) NCOUNT=4
      DO 205 IN=1,NCOUNT
      I1=INTMAT(IEL,IN)
      IN1=IN+1
      IF(IN1.GT.NCOUNT)IN1=1
      I2=INTMAT(IEL,IN1)
      IF(INODE.EQ.I1.AND.JNODE.EQ.I2)
     .JESID(IEL,IN)=IS
  205 CONTINUE
      IF(IER.GT.0) THEN
```

```
      IL = INTMAT(IER,4)
      IF(IL.EQ.0) NCOUNT=3
      IF(IL.NE.0) NCOUNT=4
      DO 210 IN=1,NCOUNT
      I1=INTMAT(IER,IN)
      IN1=IN+1
      IF(IN1.GT.NCOUNT)IN1=1
      I2=INTMAT(IER,IN1)
      IF(INODE.EQ.I2.AND.JNODE.EQ.I1)
     .JESID(IER,IN)=IS
  210 CONTINUE
      ENDIF
C
  200 CONTINUE
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-------------------------------------------------------------------
C
      SUBROUTINE GETMAT(MXELE, MXPOI, NELEM, INTMAT, COORD,
     *                  NPOIN, DNDX, DNDY, AREA, AMLP)
C
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION COORD(MXPOI,2), AREA(MXELE), X(4), Y(4), AML(4)
      DIMENSION AMLP(MXPOI), DNDX(MXELE,4), DNDY(MXELE,4)
C
      INTEGER INTMAT(MXELE,4)
C
      DO 100  IP=1,NPOIN
      AMLP(IP) = 0.
  100 CONTINUE
C
C     LOOP OVER ALL ELEMENTS
C
      DO 1000  IE=1,NELEM
      IL = INTMAT(IE,4)
      IF(IL.NE.0)  GO TO 500
C
C     TRIANGULAR ELEMENT:
C
      DO 110  IA=1,3
      N = INTMAT(IE,IA)
      X(IA) = COORD(N,1)
      Y(IA) = COORD(N,2)
  110 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)
      AMLT     = AREA(IE)/3.
      DO 120  IA=1,3
      N = INTMAT(IE,IA)
      AMLP(N) = AMLP(N) + AMLT
  120 CONTINUE
C
      DNDX(IE,1) = 0.5*B1
      DNDX(IE,2) = 0.5*B2
      DNDX(IE,3) = 0.5*B3
      DNDX(IE,4) = 0.
      DNDY(IE,1) = 0.5*C1
      DNDY(IE,2) = 0.5*C2
      DNDY(IE,3) = 0.5*C3
      DNDY(IE,4) = 0.
C
      GO TO 1000
C
C     QUADRILATERAL ELEMENT:
C
  500 CONTINUE
      DO 510  IA=1,4
      N = INTMAT(IE,IA)
      X(IA) = COORD(N,1)
      Y(IA) = COORD(N,2)
  510 CONTINUE
C
      X21 = X(2) - X(1)
      Y41 = Y(4) - Y(1)
      X41 = X(4) - X(1)
      Y21 = Y(2) - Y(1)
      Y32 = Y(3) - Y(2)
      X32 = X(3) - X(2)
      Y43 = Y(4) - Y(3)
      X43 = X(4) - X(3)
C
      V1 = 0.25*(X21*Y41 - X41*Y21)
      V2 = 0.25*(X21*Y32 - X32*Y21)
      V3 = 0.25*(X32*Y43 - X43*Y32)
      V4 = 0.25*(X41*Y43 - X43*Y41)
C
      AREA(IE) = V1 + V2 + V3 + V4
C
      AML(1) = (2.*V4+V3+2.*V2+4.*V1)/9.
      AML(2) = (V4+2.*V3+4.*V2+2.*V1)/9.
      AML(3) = (2.*V4+4.*V3+2.*V2+V1)/9.
      AML(4) = (4.*V4+2.*V3+V2+2.*V1)/9.
C
      DO 520  IA=1,4
      N = INTMAT(IE,IA)
      AMLP(N) = AMLP(N) + AML(IA)
  520 CONTINUE
C
C     COMPUTE INTEGRAL OVER AREA OF  DNDX & DNDY:
C
      DNDX(IE,1) = -0.5*(Y(4)-Y(2))
      DNDX(IE,2) =  0.5*(Y(3)-Y(1))
      DNDX(IE,3) = -DNDX(IE,1)
      DNDX(IE,4) = -DNDX(IE,2)
      DNDY(IE,1) =  0.5*(X(4)-X(2))
      DNDY(IE,2) = -0.5*(X(3)-X(1))
      DNDY(IE,3) = -DNDY(IE,1)
      DNDY(IE,4) = -DNDY(IE,2)
C
 1000 CONTINUE
C
      RETURN
      END
```

```
C
C-------------------------------------------------------------------
C
      SUBROUTINE GETLEN(MXELE, MXPOI, MXSID, NNODE, NELEM, INTMAT,
     *                  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)
      DIMENSION COORD(MXPOI,2), SLEN(MXSID,2), X(4), Y(4)
C
      INTEGER INTMAT(MXELE,NNODE), ISIDE(MXSID,4), JESID(MXELE,NNODE)
C
      TOL = 1.E-10
C
C     LOOP OVER NUMBER OF ELEMENTS
C
      DO 1000  IE=1,NELEM
      IL = INTMAT(IE,4)
      IF(IL.NE.0)  GO TO 500
C
C     TRIANGULAR ELEMENT:
C
      XC = 0.0
      YC = 0.0
      DO 320  IN=1,3
      IP = INTMAT(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.INTMAT(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
  330 CONTINUE
      GO TO 1000
C
C     QUADRILATERAL ELEMENT:
C
  500 CONTINUE
      DO 510  IN=1,4
      IP = INTMAT(IE,IN)
      X(IN) = COORD(IP,1)
      Y(IN) = COORD(IP,2)
  510 CONTINUE
C
C     FIRST TRIANGLE
C
      X21 = X(3) - X(1)
      X31 = X(4) - X(1)
      Y21 = Y(3) - Y(1)
      Y31 = Y(4) - Y(1)
      AREA1 = 0.5*(X21*Y31 - X31*Y21)
      XCG1  = (X(1) + X(3) + X(4))/3.0
      YCG1  = (Y(1) + Y(3) + Y(4))/3.0
C
C     SECOND TRIANGLE
C
      X21 = X(2) - X(1)
      X31 = X(3) - X(1)
      Y21 = Y(2) - Y(1)
      Y31 = Y(3) - Y(1)
      AREA2 = 0.5*(X21*Y31 - X31*Y21)
      XCG2  = (X(1) + X(2) + X(3))/3.0
      YCG2  = (Y(1) + Y(2) + Y(3))/3.0
C
C     COORDINATE OF CENTROID
C
      AREA = AREA1 + AREA2
      XC   = (AREA1*XCG1 + AREA2*XCG2)/AREA
      YC   = (AREA1*YCG1 + AREA2*YCG2)/AREA
C
C     LOOP OVER ELEMENT SIDES
C
      DO 600  IN=1,4
      IS = JESID(IE,IN)
      IL = ISIDE(IS,1)
      IR = ISIDE(IS,2)
      IF(IL.EQ.INTMAT(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 520
      SLEN(IS,1) = SLEN(IS,2)
      GO TO 530
  520 IF(IER.GT.0)  GO TO 530
      SLEN(IS,2) = SLEN(IS,1)
  530 CONTINUE
C
  600 CONTINUE
C
 1000 CONTINUE
C
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE NDER(MXPOI, MXELE, MXBOU, NNODE, NAMAT, NBOUN,
     *     NPOIN, NTRI, NQUAD, NELEM, INTMAT, COORD, UNKNO, BSIDO,
     *     GAMMA,    CV, UNKNP,  AREA,  DNDX,  DNDY,  AMLP,
     *     SIDERX, SIDERY                                  )
C
C     COMPUTE NODAL DERIVATIVES OF  U, V, T  WRT  X & Y
C
      IMPLICIT  REAL*8(A-H,O-Z)
C
      DIMENSION  COORD(MXPOI,2), UNKNO(MXELE,NAMAT)
      DIMENSION  AREA(MXELE), DNDX(MXELE,4), DNDY(MXELE,4)
      DIMENSION  AMLP(MXPOI), SIDERX(MXPOI,3), SIDERY(MXPOI,3)
      DIMENSION  UNKNP(MXPOI,NAMAT), VAR(3)
C
      INTEGER  INTMAT(MXELE,NAMAT), BSIDO(MXBOU,4)
C
C     ZERO OUT THESE NODAL DERIVATIVES
C
      DO 10  IV=1,3
      DO 10  IP=1,NPOIN
      SIDERX(IP,IV) = 0.
      SIDERY(IP,IV) = 0.
   10 CONTINUE
C
C     COMPUTE INTEGRAL OVER AREA TERMS
C
C     LOOP OVER NUMBER OF ELEMENTS
C
      DO 100  IE=1,NELEM
      VAR(1)  = UNKNO(IE,2)/UNKNO(IE,1)
      VAR(2)  = UNKNO(IE,3)/UNKNO(IE,1)
      VEL2    = VAR(1)*VAR(1) + VAR(2)*VAR(2)
      TOTALE  = UNKNO(IE,4)/UNKNO(IE,1)
      VAR(3)  = (TOTALE - 0.5*VEL2)/CV
```

```
C
      DO 110  IN=1,NNODE
      IP = INTMAT(IE,IN)
      IF(IP.EQ.0)  GO TO 110
      DO 120  IV=1,3
      SIDERX(IP,IV) = SIDERX(IP,IV) - VAR(IV)*DNDX(IE,IN)
      SIDERY(IP,IV) = SIDERY(IP,IV) - VAR(IV)*DNDY(IE,IN)
  120 CONTINUE
  110 CONTINUE
  100 CONTINUE
C
C     COMPUTE INTEGRAL OVER BOUNDARY TERMS
C
      DO 200  IS=1,NBOUN
      II = BSIDO(IS,1)
      JJ = BSIDO(IS,2)
      IE = BSIDO(IS,3)
      IB = BSIDO(IS,4)
      DX = COORD(JJ,1) - COORD(II,1)
      DY = COORD(JJ,2) - COORD(II,2)
      DL = SQRT(DX*DX + DY*DY)
      RNX= DY/DL
      RNY=-DX/DL
C
C     THE QUANTITIES  U, V, T  IN THIS BOUNDARY INTEGRAL DEPEND ON
C     THE TYPES OF BOUNDARY
C
      IF(IB.EQ.1)  GO TO 210
      IF(IB.EQ.2)  GO TO 220
      IF(IB.EQ.3)  GO TO 230
      IF(IB.EQ.4)  GO TO 240
C
C     SUPERSONIC INFLOW
C
  210 CONTINUE
      VAR(1) = 0.5*(UNKNP(II,2)/UNKNP(II,1) + UNKNP(JJ,2)/UNKNP(JJ,1))
      VAR(2) = 0.5*(UNKNP(II,3)/UNKNP(II,1) + UNKNP(JJ,3)/UNKNP(JJ,1))
      VEL2   = VAR(1)*VAR(1) + VAR(2)*VAR(2)
      TOTALE = 0.5*(UNKNP(II,4)/UNKNP(II,1) + UNKNP(JJ,4)/UNKNP(JJ,1))
      VAR(3) = (TOTALE - 0.5*VEL2)/CV
      GO TO 250
C
C     SUPERSONIC OUTFLOW
C
  220 CONTINUE
      VAR(1) = UNKNO(IE,2)/UNKNO(IE,1)
      VAR(2) = UNKNO(IE,3)/UNKNO(IE,1)
      VEL2   = VAR(1)*VAR(1) + VAR(2)*VAR(2)
      TOTALE = UNKNO(IE,4)/UNKNO(IE,1)
      VAR(3) = (TOTALE - 0.5*VEL2)/CV
      GO TO 250
C
C     INVISCID (INSULATED)  OR  SYMMETRY BOUNDARY
C
  230 CONTINUE
      UXL    = UNKNO(IE,2)/UNKNO(IE,1)
      VYL    = UNKNO(IE,3)/UNKNO(IE,1)
      UNL    = 0.
      VTL    =-UXL*RNY + VYL*RNX
      VAR(1) = UNL*RNX - VTL*RNY
      VAR(2) = UNL*RNY + VTL*RNX
      VEL2   = UNL*UNL + VTL*VTL
      TOTALE = UNKNO(IE,4)/UNKNO(IE,1)
      VAR(3) = (TOTALE - 0.5*VEL2)/CV
      GO TO 250
```

```
C
C     VISCOUS WITH SPECIFIED WALL TEMPERATURE
C
  240 CONTINUE
      VAR(1) = 0.
      VAR(2) = 0.
      TOTALE = 0.5*(UNKNP(II,4)/UNKNP(II,1) + UNKNP(JJ,4)/UNKNP(JJ,1))
      VAR(3) = TOTALE/CV
C
  250 CONTINUE
C
      DO 260  IN=1,2
      IP = BSIDO(IS,IN)
      DO 270  IV=1,3
      SIDERX(IP,IV) = SIDERX(IP,IV) + 0.5*RNX*DL*VAR(IV)
      SIDERY(IP,IV) = SIDERY(IP,IV) + 0.5*RNY*DL*VAR(IV)
  270 CONTINUE
  260 CONTINUE
  200 CONTINUE
C
C     DIVIDE THRU BY LUMPED MASS AT NODES
C
      DO 300  IP=1,NPOIN
      DO 310  IV=1,3
      SIDERX(IP,IV) = SIDERX(IP,IV)/AMLP(IP)
      SIDERY(IP,IV) = SIDERY(IP,IV)/AMLP(IP)
  310 CONTINUE
  300 CONTINUE
C
      RETURN
      END
C
C-------------------------------------------------------------------
C
      SUBROUTINE COMPUTE(MXELE, MXPOI, MXSID, NELEM, NSIDE, UNKNP,
     *                   GAMMA, EPSLAM, CSAFE, DT, ISIDE, RSIDO,
     *                   AREA, SLEN, UNKN1, UNKNO,
     *                   NNODE, INTMAT, IVISC, IDISS, SIDERX, SIDERY,
     *                   PR, CSAFV, TREFF, S, AMUF, CV)
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION RHS0(NELEM,4), UNKNO(MXELE,4), UNKN1(MXELE,4)
      DIMENSION RSIDO(MXSID,3), DELTE(NELEM), AREA(MXELE)
      DIMENSION UNKNP(MXPOI,4), SLEN(MXSID,2)
      DIMENSION RLAM(4), R(4,4), RI(4,4), DU(4)
      DIMENSION DISS(4), FSUM(4), FLUX(4), AVROE(4,4)
      DIMENSION SIDERX(MXPOI,3), SIDERY(MXPOI,3)
      DIMENSION VFLUX(4)
C
      INTEGER ISIDE(MXSID,4), INTMAT(MXELE,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=1,4
      DO 30  IE=1,NELEM
      RHS0(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)
      IF (IER.EQ.0) THEN
      WRITE(*,*)
      WRITE(*,'(A)') ' !!  DATA  ERROR  !!'
      WRITE(*,'(A)') '***  PLEASE CHECK BOUNDARY DATA  ***'
      WRITE(*,*)
      STOP
      END IF
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))
      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
      IF(IER.EQ.-4)  GO TO 140
      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))
      IF(IVISC.EQ.1)  TEMPR = (TER - 0.5*(UXR*UXR+VYR*VYR))/CV
      GO TO 200
C
C     SUPERSONIC OUTFLOW
C
  120 CONTINUE
      RHOR = RHOL
```

```
      UXR  = UXL
      VYR  = VYL
      TER  = TEL
      PRESR= PRESL
      IF(IVISC.EQ.1)  TEMPR = (TER - 0.5*(UXR*UXR+VYR*VYR))/CV
      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)
      IF(IVISC.EQ.1)  TEMPR = (TER - 0.5*(UXR*UXR+VYR*VYR))/CV
      GO TO 200
C
C     VISCOUS WALL WITH SPECIFIED TEMPERATURE
C
  140 CONTINUE
      II     = ISIDE(IS,1)
      JJ     = ISIDE(IS,2)
      PRESR  = PRESL
      UXR    =-UXL
      VYR    =-VYL
      TEWALL = 0.5*(UNKNP(II,4)/UNKNP(II,1) + UNKNP(JJ,4)/UNKNP(JJ,1))
      TER    = TEWALL
      IF(TER.LE.0.)  WRITE(*,180)  IEL
  180 FORMAT(' *** WARNING ***  COMPUTED WALL TEMP. NEXT TO ELEMENT',I5,
     &     ' IS NEGATIVE', /, ' TRY ANOTHER STATEMENT COMMENTED IN CODE')
      RHOR   = PRESR/(GAM1*(TER - 0.5*(UXR*UXR+VYR*VYR)))
       IF(RHOR.LT.0.0) RHOR = -RHOR
      IF(IVISC.EQ.1)  TEMPR = TER/CV
      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))
      IF(IVISC.EQ.1)  TEMPR = (TER - 0.5*(UXR*UXR+VYR*VYR))/CV
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     ALSO COMPUTE AVERAGE VALUES OF VISCOUS FLUXES IF NEEDED
C     (HIGH-SPEED VISOUS COMPRESSIBLE FLOW)
C
      IF(IVISC.EQ.0)  GO TO 210
      II   = ISIDE(IS,1)
      JJ   = ISIDE(IS,2)
      DUML = TEMPL/TREFF
      AMUL = SQRT(DUML*DUML*DUML)
      DUML = (TREFF+S)/(TEMPL+S)
      AMUL = AMUF*AMUL*DUML
      DUMR = TEMPR/TREFF
      AMUR = SQRT(DUMR*DUMR*DUMR)
      DUMR = (TREFF+S)/(TEMPR+S)
```

```
      AMUR = AMUF*AMUR*DUMR
      AMU  = 0.5*(AMUL+AMUR)
      TK   = GAMMA*CV*AMU/PR
      UVEL = 0.5*(UXL+UXR)
      VVEL = 0.5*(VYL+VYR)
      DUDX = 0.5*(SIDERX(II,1) + SIDERX(JJ,1))
      DUDY = 0.5*(SIDERY(II,1) + SIDERY(JJ,1))
      DVDX = 0.5*(SIDERX(II,2) + SIDERX(JJ,2))
      DVDY = 0.5*(SIDERY(II,2) + SIDERY(JJ,2))
      DTDX = 0.5*(SIDERX(II,3) + SIDERX(JJ,3))
      DTDY = 0.5*(SIDERY(II,3) + SIDERY(JJ,3))
      SXX  = (2./3.)*AMU*(2.*DUDX - DVDY)
      SXY  =         AMU*(   DUDY + DVDX)
      SYY  = (2./3.)*AMU*(2.*DVDY - DUDX)
      QX   = -TK*DTDX
      QY   = -TK*DTDY
C
      VFLUX(1) =  0.
      VFLUX(2) = -SXX*RNX - SXY*RNY
      VFLUX(3) = -SXY*RNX - SYY*RNY
      VFLUX(4) = -(UVEL*SXX + VVEL*SXY - QX)*RNX
     &           -(UVEL*SXY + VVEL*SYY - QY)*RNY
C
  210 CONTINUE
C
C     COMPUTE INTERFACE VALUES (SEE APPENDIX B, GNOFFO'S PAPER)
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     SET EPSLAM TO BE VERY SMALL FOR ALL QUADS WITH SIDES PARALLEL
C     TO THE WALL IF NEEDED
C
      IF(IDISS.NE.0)  GO TO 230
      LLL = INTMAT(IEL,4)
      IF(LLL.EQ.0)  GO TO 230
      NNI = INTMAT(IEL,1)
```

```
      NNJ = INTMAT(IEL,2)
      NNK = INTMAT(IEL,3)
      NNL = INTMAT(IEL,4)
      II  = ISIDE(IS,1)
      JJ  = ISIDE(IS,2)
      IF((NNI.EQ.II).AND.(NNJ.EQ.JJ))  EPSACT = 0.001*EPSLAM
      IF((NNI.EQ.JJ).AND.(NNJ.EQ.II))  EPSACT = 0.001*EPSLAM
      IF((NNK.EQ.II).AND.(NNL.EQ.JJ))  EPSACT = 0.001*EPSLAM
      IF((NNK.EQ.JJ).AND.(NNL.EQ.II))  EPSACT = 0.001*EPSLAM
  230 CONTINUE

C
C     RESET THESE EIGENVALUES IF THEY ARE LESS THAN EPSLAM
C
      DO 235  IR=1,4
      IF(RLAM(IR).GE.EPSACT)  GO TO 235
      RLAM(IR) = 0.5*(RLAM(IR)*RLAM(IR)/EPSACT + EPSACT)
  235 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.
      IF(IVISC.EQ.0)  GO TO 255
      RHOAVG = 0.5*(RHOL+RHOR)
      AUX    = CSAFV*2.*AMU/(RHOAVG*PR*EIGMAX*REPLEN)
  255 CONTINUE
      IF(CSAFE.EQ.0) THEN
       DTL = DT
       ELSE
       DTL = CSAFE*(REPLEN/EIGMAX)/(1. + AUX)
       ENDIF
      DELTE(IEL) = MIN(DELTE(IEL),DTL)
      IF(IER.LE.0) GOTO 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) = RHOL - RHOR
      DU(2) = RHOL*UXL - RHOR*UXR
      DU(3) = RHOL*VYL - RHOR*VYR
      DU(4) = RHOL*TEL - RHOR*TER
C
C     COMPUTE  DISS  =  [AS] UL-UR :
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     ADD VISCOUS FLUX COMPONENTS FOR VISCOUS ANALYSIS
C
      IF(IVISC.EQ.0)  GO TO 335
      DO 333  I=1,4
      FLUX(I) = FLUX(I) + VFLUX(I)
  333 CONTINUE
  335 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=1,4
      UNKNO(IE,IA) = UNKN1(IE,IA) + DELTE(IE)*RHS0(IE,IA)/AREA(IE)
 1100 CONTINUE
C
      RETURN
      END
C
C----------------------------------------------------------------------
C
      SUBROUTINE GETPLOT(MXELE, MXPOI, MXBOU, NNODE, NELEM, NPOIN,
     *                   NBOUN, INTMAT, COORD, BSIDO, UNKNP, UNKNP1,
     *                   UNKNO)
      IMPLICIT  REAL*8(A-H,O-Z)
      DIMENSION UNKNP(MXPOI,4), UNKNP1(MXPOI,4), UNKNO(MXELE,4)
      DIMENSION COORD(MXPOI,2), INTMAT(MXELE,NNODE)
C
      INTEGER IDUM1(NPOIN), BSIDO(MXBOU,4)
C
C     CONVERT ELEMENT QUANTITIES TO NODAL QUANTITIES
C
      DO 50 IA=1,4
      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 = INTMAT(IE,IN)
      IF(IP.EQ.0)  GOTO 103
      IDUM1(IP) = IDUM1(IP) + 1
      DO 102  IA=1,4
      UNKNP(IP,IA) = UNKNP(IP,IA) + UNKNO(IE,IA)
  102 CONTINUE
  101 CONTINUE
  103 CONTINUE
  100 CONTINUE
```

```
C
      DO 105  IA=1,4
      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=2,4
      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=1,4
      UNKNP(II,IA) = UNKNP1(II,IA)
      UNKNP(JJ,IA) = UNKNP1(JJ,IA)
  125 CONTINUE
  120 CONTINUE
C
C     CREATE RESULT FILE:
C
      WRITE(8,130) NPOIN
  130 FORMAT(I8,' NODES')
      WRITE(8,140)
  140 FORMAT(4X, 'NODE', 4X, 'DENSITY', 3X, 'U-VELOCITY',
     *       2X, 'V-VELOCITY', 1X, 'TOTAL ENERGY')
C
      DO 170  IP=1,NPOIN
      WRITE(8,173)  IP, (UNKNP(IP,IA), IA=1,4)
  173 FORMAT(I8, 4(2X,E10.4))
  170 CONTINUE
C
      RETURN
      END
```

ภาคผนวก ข

รายละเอียดของโปรแกรมคอมพิวเตอร์ TGHIFLOW

โปรแกรมคอมพิวเตอร์ TGHIFLOW ที่ได้ประดิษฐ์ขึ้นเพื่อวิเคราะห์ปัญหาที่มีการไหลความเร็วสูงแบบไม่หนืดดังที่ได้กล่าวไว้ในบทที่ 5 มีรายละเอียดดังนี้

```
      PROGRAM TGHIFLOW
C
C     FE PROGRAM FOR HYPERSONIC INVISCID COMPRESSIBLE FLOW WITH CST ELEMENT
C     BY USING TAYLOR-GALERKIN APPROACH
C
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        MXFRF  = MAXIMUM NUMBER OF OUTFLOW BOUNDARIES IN THE MODEL
C        MXWALL = MAXIMUM NUMBER OF WALL BOUNDARIES IN THE MODEL
C
      USE MSFLIB
      PARAMETER ( MXPOI=5000,  MXELE=5000,   MXFRF=1000, MXWALL = 500)
C
      REAL*8 CONSTF(3), COORD(MXPOI,2), UNKNOF(MXPOI,4)
      REAL*8 DNDXA(MXELE,3), DNDYA(MXELE,3), AMLE(MXELE,3)
      REAL*8 DTEF(MXELE), DTPF(MXPOI), USIDE(MXFRF,4)
      REAL*8 EF(MXPOI,4), FF(MXPOI,4), UHALF(MXELE,4)
      REAL*8 EHALF(MXELE,4), FHALF(MXELE,4), AMLF(MXPOI)
      REAL*8 ESIDE(MXFRF,4), FSIDE(MXFRF,4), RHSF7(MXPOI,4)
      REAL*8 RHSF5(MXPOI,4), RHSF6(MXPOI,4), DELUNF(MXPOI,4)
      REAL*8 RHSF(MXPOI,4), UNKF(MXPOI,4), ERR(4)
      REAL*8 DELX(MXFRF), DELY(MXFRF), AREA(MXELE)
      REAL*8 DNDX(MXELE,3), DNDY(MXELE,3), DT(1)
      CHARACTER FILNAM*12, TEXT*20, CV*4
C
      INTEGER INTMAT(MXELE,3), INTBF(MXFRF,4), IBCW(MXWALL,3)
      INTEGER IOUT(MXFRF,5)
C
      INTEGER(2) tmpday, tmpmonth, tmpyear
      INTEGER(2) tmphour, tmpminute, tmpsecond, tmphund
C
    9 WRITE(6,'(A,$)') ' PLEASE ENTER THE INPUT FILE NAME : '
      READ(5,'(A)') FILNAM
      L = NAMLEN(FILNAM)
      IF (L.EQ.0) GO TO 9
      WRITE(6,'(A,$)') ' CURRENT VERSION : '
      READ(5,'(A)') CV
C
C     UNIT 7 ==> DATA FILE
C     UNIT 8 ==> RESULT OUTPUT FILE
C
      OPEN(UNIT= 7,FILE=FILNAM(1:L)//'.D'//CV,STATUS='OLD')
      OPEN(UNIT= 8,FILE=FILNAM(1:L)//'.S'//CV,STATUS='UNKNOWN')
C
      WRITE(6,2)
      CALL GETDAT(tmpyear, tmpmonth, tmpday)
      CALL GETTIM(tmphour, tmpminute, tmpsecond, tmphund)
      WRITE(6,3) tmpday, tmpmonth, tmpyear
      WRITE(6,4) tmphour, tmpminute, tmpsecond, tmphund
      WRITE(8,3) tmpday, tmpmonth, tmpyear
      WRITE(8,4) tmphour, tmpminute, tmpsecond, tmphund
      WRITE(8,*)
```

```
    2 FORMAT(/,' START TIME :')
    3 FORMAT(3X,I2.2,'/',I2,'/',I4.4)
    4 FORMAT(3X,I2,':',I2.2,':',I2.2,':',I2.2)
    6 FORMAT(/,' STOP TIME :')
    7 FORMAT(I6,5E12.5,2X,I2,':',I2.2,':',I2.2,':',I2.2)
C
      NAMATF = 4
      TOL = 1.0E-06
C
C     READ INPUT DATA TITLE:
C
      READ(7,*)  NLINES
      DO 10 ILINE=1,NLINES
      READ(7,1)  TEXT
   10 CONTINUE
C
C     READ ALL PARAMETERS FOR THE PROBLEM CONSIDERED:
C
      READ(7,1)  TEXT
      READ(7,*)  NEF, NPF, NFRF, NWALL
C
C     READ INPUT DATA:
C
      CALL GETNPUT(NEF, NPF, MXELE, MXPOI, NFRF, MXFRF, MXWALL, NWALL,
     *  IBCW, INTBF, CONSTF, DT, NSTEPS, NSHOW, INTMAT, COORD, UNKNOF)
C
C     FIND ELEMENT NO. FOR OUTFLOW EDGE:
C
      DO 12 I=1,NFRF
      DO 12 J=1,5
      IOUT(I,J) = 0
   12 CONTINUE
      J  = 1
      JJ = 1
      II = 1
      DO 15 I=1,NFRF
      IF(INTBF(I,4).NE.2) GOTO 15
      IF(J.EQ.1) THEN
      IOUT(II,1) = INTBF(I,1)
      IOUT(II,2) = INTBF(I,3)
      N1 = INTBF(I,1)
      J = 2
      JJ = 2
      ENDIF
      DO 14 K=1,NEF
      DO 14 L=1,3
      N = INTMAT(K,L)
      IL = 0
      IF(N.EQ.N1) THEN
      DO 13 IJ=1,4
      IF(K.EQ.IOUT(II,IJ+1)) IL = IL + 1
   13 CONTINUE
      IF(IL.EQ.1) GOTO 14
      JJ =  JJ + 1
      IOUT(II,JJ) = K
      J = 1
      ENDIF
   14 CONTINUE
      IF(IOUT(II,JJ).NE.0) THEN
      II = II + 1
      J = 1
      ENDIF
   15 CONTINUE
C
C     TRANSFORM PRIMATIVE VARIABLES INTO CONSERVATIVE VARIABLES:
C
      CALL TRNSFOR(NPF, MXPOI,      UNKNOF)
C
```

```
C     COMPUTE ALL ELEMENT MATRICES:
C
      CALL GETMAT(NEF, MXELE, NPF, MXPOI, COORD, INTMAT,
     *             AREA,  AMLE, DNDXA, DNDYA, DNDX, DNDY, AMLF )
C
C     COMPUTE DIRECTION COSINES OF THE OUTFLOW NORMAL VECTORS:
C
      CALL GETLM(MXFRF, NFRF, MXPOI, COORD,
     *            INTBF, DELX, DELY)
C
C
C     #####  TRANSIENT  FLUID  LOOP  #####
C
      ISHOW = 1
      DO 9000  ISTEP=1,NSTEPS
C
C---------------------------------------------------------------------
C
C     F L U I D   E Q U A T I O N S
C
C
      DO 20  I=1,NAMATF
      DO 20  K=1,NPF
      DELUNF(K,I) = 0.
      UNKF(K,I) = UNKNOF(K,I)
   20 CONTINUE
C
C     COMPUTE FLOW ELEMENT AND NODAL LOCAL TIME STEPS:
C
      IF(CONSTF(3).EQ.0.)  GO TO 50
      CALL LCDTF(NEF, MXELE, NPF, MXPOI, COORD, INTMAT, CONSTF,
     *     UNKNOF,    DTEF, DTPF )
      GOTO 90
   50 CONTINUE
      DO 70  K=1,NEF
      DTEF(K) = DT(1)
   70 CONTINUE
      DO 80  K=1,NPF
      DTPF(K) = DT(1)
   80 CONTINUE
   90 CONTINUE
C
C     COMPUTE INVISCID FLUXES  EF  AND  FF  FOR FLUID NODES:
C
      CALL GETEFF(NPF, MXPOI, NAMATF, CONSTF,       UNKNOF, EF, FF)
C
C     COMPUTE ELEMENT QUANTITIES AT HALF STEP:
C
      CALL GETHALF(NAMATF, NEF, MXELE, MXPOI, UNKNOF,
     * INTMAT, EF, FF, AMLE, DNDXA, DNDYA, AREA, DTEF,     UHALF)
C
C     COMPUTE ELEMENT FLUX QUANTITIES AT HALF STEP:
C
      CALL GETEFF(NEF, MXELE, NAMATF, CONSTF,   UHALF, EHALF, FHALF)
C
C     COMPUTE SIDE QUANTITIES AT HALF STEP:
C
      CALL GETSIDE(NAMATF, MXPOI, MXFRF, NFRF, UNKNOF, INTBF,
     *              DELX, DELY, EF, FF, DTEF, MXELE, IOUT, NEF,
     *              DNDXA, DNDYA, COORD, INTMAT, AMLE,    USIDE)
C
C     COMPUTE ELEMENT SIDE FLUX QUANTITIES AT HALF STEP:
C
      CALL GETEFF(NFRF, MXFRF, NAMATF, CONSTF,  USIDE, ESIDE, FSIDE)
C
C     *** FLUID SECOND STXELE, NPF, MXPOI
C     COMPUTE INVISCID VECTOR  RHSF5:
C     (INTEGRAL OVER ELEMENT AREA)
```

```
C
      CALL GRHSF5(NAMATF, NEF, MXELE, NPF, MXPOI,
     *     EHALF, FHALF, DNDXA, DNDYA, INTMAT, DTPF,      RHSF5 )
C
C     COMPUTE INVISCID VECTOR  RHSF6:
C     (INTEGRAL OVER OUTFLOW ELEMENT EDGE)
C
      CALL GRHSF6(NAMATF, MXFRF, NFRF, NPF, MXPOI, ESIDE, FSIDE,
     *     DELX, DELY, INTBF, DTPF,                      RHSF6 )
C
C-----------------------------------------------------------------------
C
C     F L U I D   S O L U T I O N S
C
C     ADD ALL FLUID LOAD VFCTORS:
C
      DO 100  I=1,NAMATF
      DO 100  K=1,NPF
      RHSF(K,I) = RHSF5(K,I) + RHSF6(K,I)
  100 CONTINUE
C
C     SOLVE FOR FLUID NODAL INCREMENTS AND APPLY APPROPRIATE
C     BOUNDARY CONDITIONS:
C
      CALL GETDELF(NAMATF, MXPOI, NPF, MXFRF, NFRF, INTBF, RHSF, AMLF,
     *             DELUNF)
C
C     OBTAIN NEW FLUID SOLUTION:
C
      DO 110  I=1,NAMATF
      DO 110  K=1,NPF
      UNKNOF(K,I) = UNKNOF(K,I) + DELUNF(K,I)
  110 CONTINUE
C
C     APPLY INVISCID WALL
C
      CALL INVWALL(MXPOI, NPF, MXFRF, NFRF, INTBF,
     *             MXWALL, NWALL, IBCW, DELX, DELY,    UNKNOF )
C
C     APPLY LAPIDUS SMOOTHING TO THE FLUID SOLUTION.  FIRST,
C     OBTAIN LOAD VECTOR DUE TO ARTIFICIAL VISCOSITY:
C
      CALL LAPIDUS(NAMATF, NEF, MXELE, NPF, MXPOI, UNKNOF, DNDX,
     *  DNDY, DNDXA, DNDYA, CONSTF, DTPF, INTMAT, AREA,  RHSF7 )
C
C     SOLVE FOR FLUID NODAL INCREMENTS AND APPLY APPROPRIATE
C     BOUNDARY CONDITIONS:
C
      CALL GETDELF(NAMATF, MXPOI, NPF, MXFRF, NFRF, INTBF, RHSF7, AMLF,
     *             DELUNF)
C
C     OBTAIN FINAL FLUID SOLUTION AT THIS TIME STEP:
C
      DO 140  I=1,NAMATF
      DO 140  K=1,NPF
      UNKNOF(K,I) = UNKNOF(K,I) + DELUNF(K,I)
  140 CONTINUE
C
C     UPDATE OUTFLOW QUANTITIES
C
      CALL UPOUTF(NAMATF, MXELE, MXPOI, MXFRF, NFRF, INTMAT,
     *            INTBF, UNKNOF, IOUT, AREA)
C
C     APPLY INVISCID WALL
C
      CALL INVWALL(MXPOI, NPF, MXFRF, NFRF, INTBF,
     *             MXWALL, NWALL, IBCW, DELX, DELY,    UNKNOF )
C
```

```
C     OBTAIN TOTAL FLUID SOLUTION INCREMENTS AT THIS TIME STEP:
C
      DO 180  I=1,NAMATF
      ERROR = 0.
      DO 170  K=1,NPF
      DIFF = UNKNOF(K,I) - UNKF(K,I)
      ERROR = ERROR + DIFF*DIFF
  170 CONTINUE
      ERROR = ABS(ERROR)
      ERR(I) = SQRT(ERROR)
  180 CONTINUE
      IF(ISTEP.EQ.1) ER1 = ERR(1)
      IF(ISTEP.EQ.1) WRITE(6,25) ISTEP, (ERR(I),I=1,4)
      IF(ERR(1).LT.TOL*ER1) WRITE(6,25) ISTEP, (ERR(I),I=1,4)
      IF(ISHOW.EQ.NSHOW) WRITE(6,25) ISTEP, (ERR(I),I=1,4)
      IF(ISHOW.EQ.NSHOW) ISHOW = 0
      ISHOW = ISHOW + 1
       IF(ERR(1).LT.TOL*ER1) GOTO 250
 9000 CONTINUE
  250 CONTINUE
C
C
C     PRINT OUT FLUID SOLUTION:
C
      CALL GETPLOT(NAMATF, NPF, MXPOI, CONSTF, UNKNOF, DELUNF)
C
      CALL GETDAT(tmpyear, tmpmonth, tmpday)
      CALL GETTIM(tmphour, tmpminute, tmpsecond, tmphund)
      WRITE(6,6)
      WRITE(6,3) tmpday, tmpmonth, tmpyear
      WRITE(6,4) tmphour, tmpminute, tmpsecond, tmphund
      WRITE(8,6)
      WRITE(8,3) tmpday, tmpmonth, tmpyear
      WRITE(8,4) tmphour, tmpminute, tmpsecond, tmphund
C
    1 FORMAT(10A8)
   25 FORMAT(I8,4E12.5)
C
      STOP
      END
C
C ---------------------------------------------------------------------
C
C     [NAMLEN] COUNTS THE NUMBER OF CHARACTERS IN FILNAM.
C
      INTEGER FUNCTION NAMLEN(FILNAM)
C
      CHARACTER*12 FILNAM
C
      NAMLEN = 0
      DO 10 I = 12,1,-1
      IF (FILNAM(I:I).EQ.' ') GO TO 10
      NAMLEN = I
      GO TO 20
   10 CONTINUE
   20 RETURN
      END
C
C ---------------------------------------------------------------------
C
      SUBROUTINE GETNPUT(NEF, NPF, MXELE, MXPOI, NFRF, MXFRF, MXWALL,
     * NWALL, IBCW, INTBF, CONSTF, DT, NSTEPS, NSHOW, INTMAT, COORD,
     * UNKNOF)
C
C     READ INPUT DATA
C
C
      IMPLICIT  REAL*8(A-H,O-Z)
```

```
C
      REAL*8 CONSTF(3), COORD(MXPOI,2), UNKNOF(MXPOI,4), DT(1)
C
      INTEGER INTMAT(MXELE,3), INTBF(MXFRF,4), IBCW(MXWALL,3)
C
C
C     READ FLUID PROPERTIES & PARAMETERS:
C
C          CONSTF(1) - SPECIFIC HEAT RATIO (GAMMA)
C          CONSTF(2) - LAPIDUS CONSTANT (ALAP)
C          CONSTF(3) - SAFETY FACTOR FOR LOCAL TIME STEP (CSAFE)
C
      READ(7,1)  TEXT
      READ(7,*)  CONSTF(1)
      CONSTF(2) = -1.
      CONSTF(3) = 1.0
C
C     READ TIME STEPS:
C
      READ(7,1)  TEXT
      READ(7,*)  DT(1), NSTEPS, NPRINT, NSHOW
C
C     READ NODAL COORDINATES:
C
      READ(7,1)  TEXT
      DO 300  I=1,NPF
      READ(7,*)  IDUM, (COORD(I,J), J=1,2), (UNKNOF(I,J), J=1,4)
  300 CONTINUE
C
C     READ ELEMENT NODAL CONNECTIONS FOR ENTIRE DOMAIN
C
      READ(7,1)  TEXT
      DO 150  I=1,NEF
      READ(7,*)   IDUM, (INTMAT(I,J), J=1,3)
  150 CONTINUE
C
C     READ BOUNDARY CONDITIONS FOR FLUID NODES:
C
      READ(7,1) TEXT
      DO 1000 I=1,NFRF
      READ(7,*) II, (INTBF(I,J),J=1,4)
 1000 CONTINUE
C
C     READ BOUNDARY CONDITIONS FOR INVISCID WALL :
C
      READ(7,1) TEXT
      DO 1100 I=1,NWALL
      READ(7,*) (IBCW(I,J),J=1,3)
 1100 CONTINUE
C
    1 FORMAT(10A8)
C
      RETURN
      END
C
C ---------------------------------------------------------------------
C
      SUBROUTINE TRNSFOR(NPF, MXPOI,      UNKNOF)
C
C     TRANSFORM PRIMATIVE VARIABLES INTO CONSERVATIVE VARIABLES:
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  UNKNOF(MXPOI,4)
C
      DO 30  J=2,4
      DO 30  I=1,NPF
      UNKNOF(I,J) = UNKNOF(I,1)*UNKNOF(I,J)
   30 CONTINUE
```

```
C
      RETURN
      END
C
C -----------------------------------------------------------------------
C
      SUBROUTINE GETMAT(NEF, MXELE, NPF, MXPOI, COORD, INTMAT,
     *                  AREA, AMLE, DNDXA, DNDYA, DNDX, DNDY, AMLF)
C
C     COMPUTE ALL ELEMENT MATRICES (INTEGRALS OVER AREA)
C
      IMPLICIT  REAL*8(A-H,O-Z)
C
      REAL*8 COORD(MXPOI,2), DNDXA(MXELE,3), DNDYA(MXELE,3)
      REAL*8 X(3), Y(3), AMLE(MXELE,3), AMLF(MXPOI), AREA(MXELE)
      REAL*8 DNDX(MXELE,3), DNDY(MXELE,3)
C
      INTEGER INTMAT(MXELE,3)
C
      DO 50 I=1,NPF
      AMLF(I) = 0.0
   50 CONTINUE
C
C     LOOP OVER ALL ELEMENTS "I"
C
      DO 1000  I=1,NEF
C
      DO 110  J=1,3
      N = INTMAT(I,J)
      X(J) = COORD(N,1)
      Y(J) = COORD(N,2)
  110 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)
      AREA(I) = 0.5*(X(1)*B1 + X(2)*B2 + X(3)*B3)
      if(AREA(I).lt.0.) write(*,*) I
      XP = (X(1) + X(2) + X(3))/3.
      YP = (Y(1) + Y(2) + Y(3))/3.
      DELX1 = (X(1) - XP)
      DELX2 = (X(2) - XP)
      DELX3 = (X(3) - XP)
      DELY1 = (Y(1) - YP)
      DELY2 = (Y(2) - YP)
      DELY3 = (Y(3) - YP)
      A1 = (DELX2)*(DELY3) - (DELX3)*(DELY2)
      A2 = (DELX3)*(DELY1) - (DELX1)*(DELY3)
      A3 = (DELX1)*(DELY2) - (DELX2)*(DELY1)
      AMLT = AREA(I)/3.
C
C     COMPUTE ELEMENT MATRICES  DXMAT  AND  DYMAT
C
      DO 130  J=1,3
      DNDXA(I,J) = 0.
      DNDYA(I,J) = 0.
      DNDX(I,J)  = 0.
      DNDY(I,J)  = 0.
  130 CONTINUE
C
C     COMPUTE ELEMENT MATRICES
C
      AMLE(I,1)  = AMLT
      AMLE(I,2)  = AMLT
      AMLE(I,3)  = AMLT
```

```
      DNDXA(I,1) = 0.5*B1
      DNDXA(I,2) = 0.5*B2
      DNDXA(I,3) = 0.5*B3
      DNDYA(I,1) = 0.5*C1
      DNDYA(I,2) = 0.5*C2
      DNDYA(I,3) = 0.5*C3
      DNDX(I,1)  = DNDXA(I,1)/AREA(I)
      DNDX(I,2)  = DNDXA(I,2)/AREA(I)
      DNDX(I,3)  = DNDXA(I,3)/AREA(I)
      DNDY(I,1)  = DNDYA(I,1)/AREA(I)
      DNDY(I,2)  = DNDYA(I,2)/AREA(I)
      DNDY(I,3)  = DNDYA(I,3)/AREA(I)
C
C     OBTAIN SYSTEM NODAL LUMPED MASS VECTOR FOR
C     FLUID REGION :
C
      DO 150 J=1,3
      N = INTMAT(I,J)
      AMLF(N) = AMLF(N) + AMLE(I,J)
  150 CONTINUE
C
 1000 CONTINUE
C
      RETURN
      END
C
C ----------------------------------------------------------------------
C
      SUBROUTINE GETLM(MXFRF, NFRF, MXPOI, COORD,
     *                 INTBF, DELX, DELY)
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  COORD(MXPOI,2), DELX(MXFRF), DELY(MXFRF)
      REAL*8  X(2), Y(2)
      INTEGER  INTBF(MXFRF,4)
C
      DO 20  K=1,NFRF
      DO 10  J=1,2
      N = INTBF(K,J)
      X(J) = COORD(N,1)
      Y(J) = COORD(N,2)
   10 CONTINUE
      DELX(K) = X(2) - X(1)
      DELY(K) = Y(2) - Y(1)
   20 CONTINUE
C
      RETURN
      END
C
C ----------------------------------------------------------------------
C
      SUBROUTINE LCDTF(NEF, MXELE, NPF, MXPOI, COORD, INTMAT, CONSTF,
     *     UNKNOF,                                     DTEF, DTPF )
C
C     COMPUTE FLUID ELEMENT AND NODAL LOCAL TIME STEPS
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  CONSTF(3), COORD(MXPOI,2)
      REAL*8  UNKNOF(MXPOI,4), DTEF(MXELE), DTPF(MXPOI)
      REAL*8  X(MXELE,3), Y(MXELE,3)
      REAL*8  DXM(MXELE), DYM(MXELE), AELE(MXELE)
      REAL*8  UELE(MXELE), VELE(MXELE)
C
      INTEGER  INTMAT(MXELE,3)
C
C     OBTAIN ELEMENT NODAL COORDINATES:
C
      DO 10  J=1,3
```

```
      DO 10  I=1,NEF
      N = INTMAT(I,J)
      X(I,J) = COORD(N,1)
      Y(I,J) = COORD(N,2)
   10 CONTINUE
C
C     COMPUTE AVERAGE ELEMENT LENGTHS IN  X  AND  Y  DIRECTIONS:
C
      DO 20  I=1,NEF
      DEL12 = ABS(X(I,1)-X(I,2))
      DEL23 = ABS(X(I,2)-X(I,3))
      DEL13 = ABS(X(I,1)-X(I,3))
      DXM(I) = (DEL12+DEL23+DEL13)/2.
   20 CONTINUE
C
      DO 30  I=1,NEF
      DEL12 = ABS(Y(I,1)-Y(I,2))
      DEL23 = ABS(Y(I,2)-Y(I,3))
      DEL13 = ABS(Y(I,1)-Y(I,3))
      DYM(I) = (DEL12+DEL23+DEL13)/2.
   30 CONTINUE
C
C     COMPUTE AVERAGE ELEMENT VELOCITY COMPONENTS:
C     AND ABSOLUTE VELOCITY COMPONENTS IN  X & Y  DIR:
C
      DO 90  I=1,NEF
      DUMX = 0.0
      DUMY = 0.0
      DO 80  J=1,3
      N = INTMAT(I,J)
      U = UNKNOF(N,2)/UNKNOF(N,1)
      V = UNKNOF(N,3)/UNKNOF(N,1)
      DUMX = DUMX + U
      DUMY = DUMY + V
   80 CONTINUE
C
C     COMPUTE ABSOLUTE VELOCITY COMPONENTS IN  X & Y  DIR:
C
      UELE(I) = ABS(DUMX)/3.
      VELE(I) = ABS(DUMY)/3.
   90 CONTINUE
C
C     COMPUTE AVERAGE ELEMENT SPEED OF SOUND:
C     LET'S DENOTE  DUM1  AS ELEMENT TEMPERATURES;
C
      GAMMA = CONSTF(1)
      DO 120 I=1,NEF
      DUM = 0.0
      DO 100 J=1,3
      N = INTMAT(I,J)
      UNODE = UNKNOF(N,2)/UNKNOF(N,1)
      VNODE = UNKNOF(N,3)/UNKNOF(N,1)
      TNODE = (UNKNOF(N,4)/UNKNOF(N,1) -
     &        0.5*(UNODE*UNODE + VNODE*VNODE))
      DUM = DUM + TNODE/3.
  100 CONTINUE
      DUM = GAMMA*(GAMMA-1.)*DUM
      IF(DUM.LT.0.) DUM = 0.0
      AELE(I) = SQRT(DUM)
  120 CONTINUE
C
C     COMPUTE ELEMENT TIME STEPS BASED ON  CFL  CONDITIONS:
C
      CSAFE = CONSTF(3)
      DO 150  I=1,NEF
      DUMX = 1./(DXM(I)*DXM(I)) + 1./(DYM(I)*DYM(I))
      DUMY = SQRT(DUMX)
      AELE(I) = AELE(I)*DUMY
```

```
      DUMX = UELE(I)/DXM(I)
      DUMY = VELE(I)/DYM(I)
      DTEF(I) = 1./(DUMX + DUMY + AELE(I))*CSAFE
  150 CONTINUE
C
C     COMPUTE ALL NODAL TIME STEPS:
C
C     NOTE:  NODAL TIME STEP IS THE MINIMUM TIME STEP OF
C            ALL ELEMENTS SURROUNDING THAT NODE.
C
      CBIG = 1.E+10
      DO 260  I=1,NPF
      DTPF(I) = CBIG
  260 CONTINUE
C
      DO 270  J=1,3
      DO 270  I=1,NEF
      N = INTMAT(I,J)
      IF(DTEF(I).LT.DTPF(N))  DTPF(N) = DTEF(I)
  270 CONTINUE
C
      RETURN
      END
C
C ----------------------------------------------------------------------
C
      SUBROUTINE GETEFF(NPF, MXPOI, NAMATF, CONSTF,    UNKNOF, EF, FF)
C
C     COMPUTE INVISCID FLUXES  EF  AND  FF  FOR FLUID NODES
C
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  UNKNOF(MXPOI,4), P(MXPOI,4)
      REAL*8  EF(MXPOI,4), FF(MXPOI,4)
      REAL*8  CONSTF(3)
C
C
      DO 10  J=1,NAMATF
      DO 10  I=1,NPF
      EF(I,J) = 0.
      FF(I,J) = 0.
   10 CONTINUE
C
      GAM  = CONSTF(1)
      GAM1 = GAM - 1.
      GAM3 = 0.5*(3.-GAM)


C
      DO 20  I=1,NPF
      EF(I,1) = UNKNOF(I,2)
      EF(I,3) = UNKNOF(I,2)*UNKNOF(I,3)/
     1            UNKNOF(I,1)
      FF(I,1) = UNKNOF(I,3)
      FF(I,2) =     EF(I,3)
   20 CONTINUE
C
      DO 30  I=1,NPF
      P(I,1) =     UNKNOF(I,2)/UNKNOF(I,1)
      P(I,2) =     UNKNOF(I,3)/UNKNOF(I,1)
      P(I,3) = GAM*UNKNOF(I,4)/UNKNOF(I,1)
      P(I,4) =-0.5*GAM1*(P(I,1)*P(I,1) + P(I,2)*P(I,2))
   30 CONTINUE
C
      DO 40  I=1,NPF
      EF(I,2) = GAM3*UNKNOF(I,2)*P(I,1)
     1   + GAM1*(UNKNOF(I,4) - 0.5*UNKNOF(I,3)*P(I,2))
      EF(I,4) = UNKNOF(I,2)*P(I,4) + UNKNOF(I,2)*P(I,3)
```

```
      FF(I,3) = GAM3*UNKNOF(I,3)*P(I,2)
     1    + GAM1*(UNKNOF(I,4) - 0.5*UNKNOF(I,2)*P(I,1))
      FF(I,4) = UNKNOF(I,3)*P(I,4) + UNKNOF(I,3)*P(I,3)
   40 CONTINUE
C
      RETURN
      END
C
C ----------------------------------------------------------------------
C
      SUBROUTINE GETHALF(NAMATF, NEF, MXELE, MXPOI, UNKNOF,
     * INTMAT, EF, FF, AMLE, DNDXA, DNDYA, AREA, DTEF,    UHALF)
C
C     COMPUTE ELEMENT QUANTITIES AT HALF STEP
C
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  UNKNOF(MXPOI,4)
      REAL*8  EF(MXPOI,4), FF(MXPOI,4)
      REAL*8  AMLE(MXELE,3), AREA(MXELE)
      REAL*8  DNDXA(MXELE,3), DNDYA(MXELE,3)
      REAL*8  UHALF(MXELE,4), DTEF(MXELE)
C
      INTEGER  INTMAT(MXELE,3)
C
C     COMPUTE ELEMENT QUANTITIES AT HALF STEP:
C
      DO 40  J=1,NAMATF
      DO 40  K=1,NEF
      UHALF(K,J) = 0.
   40 CONTINUE
C
      DO 50  J=1,NAMATF
      DO 50  I=1,3
      DO 50  K=1,NEF
      N = INTMAT(K,I)
      UELE = UNKNOF(N,J)
      EELE = EF(N,J)
      FELE = FF(N,J)
      UHALF(K,J) = UHALF(K,J) + AMLE(K,I)*UELE
     &          -0.5*DTEF(K)*( DNDXA(K,I)*EELE +
     &                         DNDYA(K,I)*FELE )
   50 CONTINUE
C
      DO 60  J=1,NAMATF
      DO 60  K=1,NEF
      UHALF(K,J) = UHALF(K,J)/AREA(K)
   60 CONTINUE
C
      RETURN
      END
C
C ----------------------------------------------------------------------
C
      SUBROUTINE UPOUTF(NAMATF, MXELE, MXPOI, MXFRF, NFRF, INTMAT,
     *                  INTBF, UNKNOF, IOUT, AREA)
C
C     UPDATE OUTFLOW QUANTITIES
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8 UNKNOF(MXPOI,4), AREA(MXELE)
C
      INTEGER  INTMAT(MXELE,3), IOUT(MXFRF,5), INTBF(MXFRF,4)
C
      USUM = 0.
      DO 20  J=1,NAMATF
      DO 20  I=1,NFRF
      IONE = 0
```

```
      ASUM = 0.
      IF(IOUT(I,1).EQ.0) GOTO 20
      DO 10  K=1,2
      IE = IOUT(I,K+1)
      IF(IE.EQ.0) GOTO 10
      N1 = INTMAT(IE,1)
      N2 = INTMAT(IE,2)
      N3 = INTMAT(IE,3)
      TEM = (UNKNOF(N1,J) + UNKNOF(N2,J) + UNKNOF(N3,J))/3.
      USUM = USUM + AREA(IE)*TEM
      ASUM = ASUM + AREA(IE)
      IONE = IONE + 1
   10 CONTINUE
      IF(IONE.EQ.1) USUM = 0.
      IF(IONE.EQ.1) GOTO 20
      UF = USUM/ASUM
      NI = IOUT(I,1)
      UNKNOF(NI,J) = UF
      USUM = 0.
   20 CONTINUE
C
      RETURN
      END
C
C ----------------------------------------------------------------------
C
      SUBROUTINE GETSIDE(NAMATF, MXPOI, MXFRF, NFRF, UNKNOF, INTBF,
     *                   DELX, DELY, EF, FF, DTEF, MXELE, IOUT, NEF,
     *                   DNDXA, DNDYA, COORD, INTMAT, AMLE,    USIDE)
C
C     COMPUTE ELEMENT SIDE QUANTITIES AT HALF STEP
C
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  UNKNOF(MXPOI,4), USIDE(MXFRF,4), UHALF(MXELE,4)
      REAL*8  US(MXFRF,2,4), DELX(MXFRF), DELY(MXFRF)
      REAL*8  EF(MXPOI,4), FF(MXPOI,4), DTEF(MXELE)
      REAL*8  UHS(MXELE,4), AME(3), X(3), Y(3)
      REAL*8  DNDXA(MXELE,3), DNDYA(MXELE,3)
      REAL*8  COORD(MXPOI,2), AMLE(MXELE,3)
      REAL*8  EHALF(MXELE,4), FHALF(MXELE,4)
C
      INTEGER  INTBF(MXFRF,4), INTMAT(MXELE,3), IOUT(MXFRF,5)
C
C     OBTAIN ELEMENT SIDE NODAL QUANTITIES:
C
      DO 30  I=1,2
      DO 30  J=1,NAMATF
      DO 30  K=1,NFRF
      N = INTBF(K,I)
      US(K,I,J) = UNKNOF(N,J)
   30 CONTINUE
C
C     COMPUTE ELEMENT SIDE QUANTITIES (USING SIMPLEST FORM):
C
      DO 40  J=1,NAMATF
      DO 40  I=1,NFRF
      IE = INTBF(I,3)
      NI = INTBF(I,1)
      NJ = INTBF(I,2)
      N1 = INTMAT(IE,1)
      N2 = INTMAT(IE,2)
      N3 = INTMAT(IE,3)
      SLEN = DELX(I)*DELX(I) + DELY(I)*DELY(I)
      SLEN = SQRT(SLEN)
      TEMP = (-EF(NI,J) + EF(NJ,J))*DELX(I)/SLEN
      TEMP = (-FF(NI,J) + FF(NJ,J))*DELY(I)/SLEN + TEMP
      RDUM = 0.5*DTEF(IE)*TEMP/SLEN
```

```
      USIDE(I,J) = 0.5*(US(I,1,J) + US(I,2,J)) - RDUM
   40 CONTINUE
C
      RETURN
      END
C
C -----------------------------------------------------------------------
C
      SUBROUTINE GRHSF5(NAMATF, NEF, MXELE, NPF, MXPOI,
     *     EHALF, FHALF, DNDXA, DNDYA, INTMAT, DTPF,      RHSF5 )
C
C     COMPUTE INVISCID LOAD VECTOR  RHSF5
C
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  EHALF(MXELE,4), FHALF(MXELE,4)
      REAL*8  DNDXA(MXELE,3), DNDYA(MXELE,3)
      REAL*8  RHSF5(MXPOI,4), DTPF(MXPOI)
      REAL*8  RDUM(MXELE,3,4)
C
      INTEGER  INTMAT(MXELE,3)
C
C     OBTAIN ELEMENT NODAL LOAD VECTORS:
C
      DO 20  I=1,NAMATF
      DO 20  J=1,3
      DO 20  K=1,NEF
      RDUM(K,J,I) = DNDXA(K,J)*EHALF(K,I)
     *            + DNDYA(K,J)*FHALF(K,I)
   20 CONTINUE
C
C     OBTAIN SYSTEM NODAL LOAD VECTOR:
C
      DO 30  I=1,NAMATF
      DO 30  K=1,NPF
      RHSF5(K,I) = 0.
   30 CONTINUE
C
      DO 40  I=1,NAMATF
      DO 40  J=1,3
      DO 40  K=1,NEF
      N = INTMAT(K,J)
      RHSF5(N,I) = RHSF5(N,I) + RDUM(K,J,I)
   40 CONTINUE
C
      DO 50  I=1,NAMATF
      DO 50  K=1,NPF
      RHSF5(K,I) = DTPF(K)*RHSF5(K,I)
   50 CONTINUE
C
      RETURN
      END
C
C -----------------------------------------------------------------------
C
      SUBROUTINE GRHSF6(NAMATF, MXFRF, NFRF, NPF, MXPOI, ESIDE, FSIDE,
     *     DELX, DELY, INTBF, DTPF,                              RHSF6 )
C
C     COMPUTE OUTFLOW INVISCID VECTOR  RHSF6
C
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  ESIDE(MXFRF,4), FSIDE(MXFRF,4)
      REAL*8  DELX(MXFRF), DELY(MXFRF)
      REAL*8  RHSF6(MXPOI,4), DTPF(MXPOI), RSIDE(NFRF,4)
C
      INTEGER  INTBF(MXFRF,4)
C
```

```
      DO 10  I=1,NAMATF
      DO 10  K=1,NFRF
      RSIDE(K,I) = 0.
   10 CONTINUE
C
C     OBTAIN SIDE INVISCID QUANTITIES FOR THE NODES ON THE ELEMENT
C     OUTFLOW EDGES:
C
      DO 20  I=1,NAMATF
      DO 20  K=1,NFRF
      RSIDE(K,I) = 0.5*(  DELY(K)*ESIDE(K,I)
     &                  - DELX(K)*FSIDE(K,I) )
   20 CONTINUE
C
C     OBTAIN SYSTEM NODAL LOAD VECTOR:
C
      DO 30  I=1,NAMATF
      DO 30  K=1,NPF
      RHSF6(K,I) = 0.
   30 CONTINUE
C
      DO 40  I=1,NAMATF
      DO 40  J=1,2
      DO 40  K=1,NFRF
      IF(INTBF(K,4).NE.2) GOTO 40
      N = INTBF(K,J)
      RHSF6(N,I) = RHSF6(N,I) + RSIDE(K,I)
   40 CONTINUE
C
      DO 50  I=1,NAMATF
      DO 50  K=1,NPF
      RHSF6(K,I) = -DTPF(K)*RHSF6(K,I)
   50 CONTINUE
C
      RETURN
      END
C
C ----------------------------------------------------------------------
C
      SUBROUTINE GETDELF(NAMATF, MXPOI, NPF, MXFRF, NFRF, INTBF, RHSF,
     *                   AMLF,                                   DELUNF)
C
C     SOLVE FLUID SYSTEM EXPLICIT EQUATIONS FOR INCREMENTS OF
C     UNKNOWNS AND APPLY APPROPRIATE BOUNDARY CONDITIONS
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  AMLF(MXPOI), RHSF(MXPOI,4)
      REAL*8  DELUNF(MXPOI,4)
C
      INTEGER  INTBF(MXFRF,4)
C
C     SOLVE SYSTEM EXPLICIT EQS. FOR INCREMENTS OF  DELUNF:
C
      DO 10  I=1,NAMATF
      DO 11  J=1,NPF
      DELUNF(J,I) = RHSF(J,I)/AMLF(J)
   11 CONTINUE
   10 CONTINUE
C
C     APPLY BOUNDARY CONDITIONS (INTBF(I,4) = 1 => FIXED INLET FLOW):
C
      DO 20  I=1,NFRF
      DO 20  J=1,2
      DO 20  K=1,NAMATF
      N = INTBF(I,J)
      IF(INTBF(I,4).EQ.1)  DELUNF(N,K) = 0.
   21 CONTINUE
   20 CONTINUE
```

```
C
      RETURN
      END
C
C ------------------------------------------------------------------------
C
      SUBROUTINE LAPIDUS(NAMATF, NEF, MXELE, NPF, MXPOI, UNKNOF, DNDX,
     *     DNDY, DNDXA, DNDYA, CONSTF, DTPF, INTMAT, AREA,     RHSF7 )
C
C
C     COMPUTE LAPIDUS ARTIFICIAL VISCOSITY LOAD VECTOR USING
C     GAUSS POINT NUMERICAL INTEGRATION
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  UNKNOF(MXPOI,4), CONSTF(3), RHSEL(MXELE,3,4)
      REAL*8  RHSF7(MXPOI,4), DTPF(MXPOI), AREA(MXELE), DUME(NEF,4)
      REAL*8  DUM(NEF,3,4), DNDX(MXELE,3), DNDY(MXELE,3)
      REAL*8  DNDXA(MXELE,3), DNDYA(MXELE,3)
C
      INTEGER  INTMAT(MXELE,3)
C
      NNODE = 3
      ALAP = -CONSTF(2)
C
C     COMPUTE MAGNITUDE OF TOTAL NODAL VELOCITIES:
C
      DO 15  I=1,3
      DO 15  J=1,NAMATF
      DO 15  K=1,NEF
      RHSEL(K,I,J) = 0.
   15 CONTINUE
C
      DO 45  I=1,4
      DO 45  K=1,NEF
      DUME(K,I) = 0.
   45 CONTINUE
C
C     COMPUTE ELEMENT  U,X  AND  V,Y:
C
      DO 50  I=1,NNODE
      DO 50  K=1,NEF
      N = INTMAT(K,I)
      UE = UNKNOF(N,2)/UNKNOF(N,1)
      VE = UNKNOF(N,3)/UNKNOF(N,1)
      DUME(K,1) = DUME(K,1) + DNDX(K,I)*UE
      DUME(K,2) = DUME(K,2) + DNDY(K,I)*VE
   50 CONTINUE
C
C     THEN, THEIR ABSOLUTE VALUES:
C
      DO 60  K=1,NEF
      DUME(K,3) = ABS(DUME(K,1))
      DUME(K,4) = ABS(DUME(K,2))
   60 CONTINUE
C
C     COMPUTE ELEMENT  CAP U,X  AND  CAP U,Y:
C
      DO 70  I=1,NAMATF
      DO 80  K=1,NEF
      DUME(K,1) = 0.
      DUME(K,2) = 0.
   80 CONTINUE
C
      DO 90  J=1,NNODE
      DO 90  K=1,NEF
      IN = INTMAT(K,J)
      DUME(K,1) = DUME(K,1) + DNDX(K,J)*UNKNOF(IN,I)
      DUME(K,2) = DUME(K,2) + DNDY(K,J)*UNKNOF(IN,I)
```

```
   90 CONTINUE
C
      DO 100  K=1,NEF
      DUM(K,1,I) = AREA(K)*DUME(K,3)*DUME(K,1)
      DUM(K,2,I) = AREA(K)*DUME(K,4)*DUME(K,2)
  100 CONTINUE
   70 CONTINUE
C
      DO 110  I=1,NAMATF
      DO 110  J=1,NNODE
      DO 110  K=1,NEF
      RHSEL(K,J,I) = RHSEL(K,J,I) +
     &     (DNDXA(K,J)*DUM(K,1,I) +
     &      DNDYA(K,J)*DUM(K,2,I) )
  110 CONTINUE
C
C     OBTAIN SYSTEM NODAL LOAD VECTOR:
C
      DO 120  I=1,NAMATF
      DO 120  K=1,NPF
      RHSF7(K,I) = 0.
  120 CONTINUE
C
      DO 130  I=1,NAMATF
      DO 130  J=1,NNODE
      DO 130  K=1,NEF
      N = INTMAT(K,J)
      RHSF7(N,I) = RHSF7(N,I) + RHSEL(K,J,I)
  130 CONTINUE
C
      DO 140  I=1,NAMATF
      DO 140  K=1,NPF
      RHSF7(K,I) = -ALAP*DTPF(K)*RHSF7(K,I)
  140 CONTINUE
C
      RETURN
      END
C
C ---------------------------------------------------------------------
C
      SUBROUTINE INVWALL(MXPOI, NPF, MXFRF, NFRF, INTBF,
     *              MXWALL, NWALL, IBCW, DELX, DELY,    UNKNOF )
C
C     APPLY INVISCID WALL
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8 UNKNOF(MXPOI,4)
      REAL*8 DELX(MXFRF), DELY(MXFRF)
      REAL*8 RL(NPF), RM(NPF)
C
      INTEGER INTBF(MXFRF,4), ONE(NPF)
      INTEGER IBCW(MXWALL,3)
C
      DO 100 I=1,NPF
      RL(I) = 0.
      RM(I) = 0.
      ONE(I) = 0
  100 CONTINUE
C
      DO 500 I=1,NFRF
C
      IC = INTBF(I,4)
      IF(IC.NE.3) GOTO 500
      A  = DELX(I)*DELX(I) + DELY(I)*DELY(I)
      A  = SQRT(A)
      TEMRL = DELX(I)/A
      TEMRM = DELY(I)/A
      DO 400 K=1,2
```

```
      IN = INTBF(I,K)
      RL(IN) = RL(IN) + TEMRL
      RM(IN) = RM(IN) + TEMRM
      ONE(IN) = ONE(IN) + 1
  400 CONTINUE
  500 CONTINUE
C
      DO 600 I=1,NWALL
      IN = IBCW(I,2)
      RL(IN) = RL(IN)/ONE(IN)
      RM(IN) = RM(IN)/ONE(IN)
      UOLD = UNKNOF(IN,2)/UNKNOF(IN,1)
      VOLD = UNKNOF(IN,3)/UNKNOF(IN,1)
      UNEW = UOLD*RL(IN)*RL(IN) + VOLD*RL(IN)*RM(IN)
      VNEW = UOLD*RL(IN)*RM(IN) + VOLD*RM(IN)*RM(IN)
      UNKNOF(IN,2) = UNEW*UNKNOF(IN,1)
      UNKNOF(IN,3) = VNEW*UNKNOF(IN,1)
  600 CONTINUE
C
      DO 700 I=1,NWALL
      II = IBCW(I,3)
      IF(II.EQ.0) GOTO 700
      N1 = IBCW(I-1,2)
      N  = IBCW(I,2)
      N2 = IBCW(I+1,2)
      U1 = UNKNOF(N1,2)/UNKNOF(N1,1)
      U2 = UNKNOF(N2,2)/UNKNOF(N2,1)
      V1 = UNKNOF(N1,3)/UNKNOF(N1,1)
      V2 = UNKNOF(N2,3)/UNKNOF(N2,1)
      UNEW = 0.5*(U1 + U2)
      VNEW = 0.5*(V1 + V2)
      UNKNOF(N,2) = UNEW*UNKNOF(N,1)
      UNKNOF(N,3) = VNEW*UNKNOF(N,1)
  700 CONTINUE
C
      RETURN
      END
C
C ---------------------------------------------------------------------
C
      SUBROUTINE GETPLOT(NAMATF, NPF, MXPOI, CONSTF, UNKNOF, DELUNF)
C
C     PRINT OUT FLOW SOLUTION
C
      IMPLICIT  REAL*8(A-H,O-Z)
      REAL*8  UNKNOF(MXPOI,4), DELUNF(MXPOI,4), CONSTF(3)
C
C     TRANSFORM CONSERVATIVE VARIABLES BACK TO PRIMATIVE VARIABLES:
C
      GAM = CONSTF(1)
      DO 10  I=1,NPF
      DELUNF(I,1) = UNKNOF(I,1)
   10 CONTINUE
      DO 15  J=2,NAMATF
      DO 15  I=1,NPF
      DELUNF(I,J) = UNKNOF(I,J)/UNKNOF(I,1)
   15 CONTINUE
C
      WRITE(8,'(I9)') NPF
      WRITE(8,100)
      WRITE(8,'(A)')
      DO 20  I=1,NPF
      WRITE(8,200)  I, (DELUNF(I,K), K=1,NAMATF)
   20 CONTINUE
  100 FORMAT('     NODE         RHO         U          V           E')
  200 FORMAT(I8, 4E16.5)
      RETURN
      END
```

ภาคผนวก ค

รายละเอียดของโปรแกรมคอมพิวเตอร์ HEAT2DTS

โปรแกรมคอมพิวเตอร์ HEAT2DTS ที่ได้ประดิษฐ์ขึ้นเพื่อวิเคราะห์ปัญหาการถ่ายเทความร้อนที่ขึ้นกับเวลาดังที่ได้กล่าวไว้ในบทที่ 5 มีรายละเอียดดังนี้

```
C     PROGRAM  HEAT2DTS
C
C     A FINITE ELEMENT THERMAL ANALYSIS PROGRAM FOR TWO-DIMENSIONAL
C     TRANSIENT HEAT CONDUCTION WITH INTERNAL HEAT GENERATION,
C     APPLIED SURFACE HEATING, AND SURFACE CONVECTION.
C
C
C     THE VALUES DECLARED IN THE PARAMETER STATEMENT BELOW SHOULD
C     BE ASSIGNED ACCORDING TO THE SIZE OF THE PROBLEMS
C
C        MXPOI = MAXIMUM NUMBER OF NODES IN THE MODEL
C        MXELE = MAXIMUM NUMBER OF ELEMENTS IN THE MODEL
C
      PARAMETER (MXPOI=1000, MXELE=1000)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  COORD(MXPOI,2), TEMP(MXPOI), TEXT(20)
      DIMENSION  SYSK(MXPOI), SYSQ(MXPOI), TN(MXPOI)
      DIMENSION  SYSC(MXPOI), TK(4), FLUX(MXPOI)
      DIMENSION  DEN(4), SH(4), TH(4)
      CHARACTER*10  NAME1, NAME2, NAME3, NAME4
C
      INTEGER  INTMAT(MXELE,3), IBC(MXPOI), LTYPE(MXELE,3)
      INTEGER  ID(MXPOI), IDR(10*MXPOI)
C
   10 WRITE(6,20)
   20 FORMAT(/, ' PLEASE ENTER THE INPUT FILE NAME:')
      READ(5, '(A)', ERR=10)  NAME1
      OPEN(UNIT=7, FILE=NAME1, STATUS='OLD', ERR=10)
C
C     PRINT OUT NODAL TEMPERATURE SOLUTIONS:
C
   50 WRITE(6,60)
   60 FORMAT(/, ' PLEASE ENTER FILE NAME FOR TEMPERATURE'
     *        ' SOLUTIONS:'                                    )
      READ(5, '(A)', ERR=50)  NAME2
      OPEN(UNIT=8, FILE=NAME2, STATUS='NEW', ERR=50)
C
      WRITE(6,*)
      WRITE(6,*) ' PLEASE ENTER NO. OF ITERATION TO SHOW ERROR :'
      READ(5,*) NSHOW
C
C     READ TITLE OF COMPUTATION:
C
      READ(7,*)  NLINES
      DO 100  ILINE=1,NLINES
      READ(7,1)  TEXT
    1 FORMAT(20A4)
  100 CONTINUE
C
C     READ INPUT DATA:
C
      READ(7,1)  TEXT
      READ(7,*)  NPOIN, NELEM
      IF(NPOIN.GT.MXPOI)  WRITE(6,110)  NPOIN
```

```
  110 FORMAT(/,' PLEASE INCREASE THE PARAMETER MXPOI TO ', I5)
      IF(NPOIN.GT.MXPOI)  STOP
      IF(NELEM.GT.MXELE)  WRITE(6,120)  NELEM
  120 FORMAT(/,' PLEASE INCREASE THE PARAMETER MXELE TO ', I5)
      IF(NELEM.GT.MXELE)  STOP
      WRITE(8,124)  NPOIN
  124 FORMAT('  NODAL TEMPERATURE SOLUTIONS [', I5,']:',
     *       //, 2X, 'NODE', 3X, 'TEMPEARTURE'     )
      READ(7,*)  NMAT
      DO 125 I=1,NMAT
      READ(7,*)  TK(I), DEN(I), SH(I), TH(I)
  125 CONTINUE
      READ(7,1)  TEXT
      READ(7,*)  Q, QS, H, TI, TS, NTIME, NPRINT
      READ(7,1)  TEXT
      DO 130  IP=1,NPOIN
      READ(7,*) I, IBC(IP), (COORD(IP,K),K=1,2), TEMP(IP), FLUX(IP)
      ID(IP) = I
      IDR(I) = IP
  130 CONTINUE
      LT1 = 0
      LT2 = 0
      LT3 = 0
      READ(7,1)  TEXT
      DO 140  IE=1,NELEM
      READ(7,*)  II, (INTMAT(II,J), J=1,3), (LTYPE(II,K), K=1,3)
      IF(II.NE.IE)  WRITE(6,150)  IE
  150 FORMAT(/, ' ELEMENT NO.', I5, ' IN DATA FILE IS MISSING')
      IF(II.NE.IE)  STOP
  140 CONTINUE
      IF(Q.NE.0.)   LT1 = 1
      IF(QS.NE.0.)  LT2 = 1
      WRITE(6,160)
  160 FORMAT(/,' THE F.E. MODEL INCLUDES THE FOLLOWING',
     *          ' HEAT TRANSFER MODE(S):',
     *       /,'    --  HEAT CONDUCTION                 ')
      IF(LT1.EQ.1)  WRITE(6,170)
  170 FORMAT(  '    --  INTERNAL HEAT GENERATION       ')
      IF(LT2.EQ.1)  WRITE(6,180)
  180 FORMAT(  '    --  SPECIFIED SURFACE HEATING      ')
C
      WRITE(6,190)
  190 FORMAT(/,' *** ESTABLISHING ELEMENT MATRICES AND',
     *          ' ASSEMBLING ELEMENT EQUATIONS ***',/    )
      WRITE(6,'(A)') '        ITER       TIME             T-L2-NORM'
      WRITE(6,*)
C
      ISHOW  = 1
      IPRINT = NPRINT
      DO 1000 I=1,NTIME
C
      DO 200 J=1,NPOIN
      TN(J) = TEMP(J)
  200 CONTINUE
C
C     ESTABLISH ALL ELEMENT MATRICES ASSOCIATED WITH THE SPECIFIED
C     HEAT TRANSFER MODES AND ASSEMBLE THEM FOR SYSTEM MATRICES IN
C     THE FORM NEEDED FOR MINIMUM MEMORY REQUIREMENT:
C
      CALL TRI(NELEM, INTMAT, COORD,    TK,    DEN,   SH,
     *    FLUX,  Q,     QS,     TH,  LTYPE, SYSC, SYSQ,  H,
     *    TI, TS,   TIME,     TN,  NPOIN, MXPOI, MXELE, IDR )
C
      CALL SOLVE(NPOIN, NELEM, MXPOI, MXELE, SYSC, SYSQ, TS,IBC,TEMP,TN)
C
      IF(ISHOW.EQ.I) THEN
      SUM = 0.
      DO 300 J=1,NPOIN
```

```
      DIFF = TEMP(J) - TN(J)
      SUM = SUM + DIFF*DIFF
  300 CONTINUE
      SUM = SQRT(SUM)
      WRITE(6,'(I12,2X,E12.4,2X,E16.6)') I, REAL(I)*TS, SUM
      IF(I.EQ.1) ISHOW = NSHOW
      IF(I.GT.1) ISHOW = ISHOW + NSHOW
     ENDIF
C
      IF(I.EQ.IPRINT) THEN
      WRITE(8,505) I
  505 FORMAT(/,' **** NO. OF ITERATION = ', I8,' ****')
      WRITE(8,*)
      DO 500  IP=1,NPOIN
      WRITE(8,510)  ID(IP), TEMP(IP)
  510 FORMAT(I6, E14.6)
  500 CONTINUE
      IPRINT = IPRINT + NPRINT
      ENDIF
C
 1000 CONTINUE
C
      STOP
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE TRI(NELEM, INTMAT, COORD,    TK,    DEN,   SH,
     *       FLUX,   Q,      QS,     TH, LTYPE, SYSC, SYSQ,  H,
     *       TI, TS,   TIME,    TN,  NPOIN, MXPOI, MXELE, IDR )
C
C     ESTABLISH ELEMENT MATRICES ACCORDING TO THE SPECIFIED HEAT
C     TRANSFER MODES AND ASSEMBLE THEM FOR SYSTEM EQUATIONS
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  COORD(MXPOI,2), SYSC(MXPOI), SYSQ(MXPOI)
      DIMENSION  AKC(3,3), AKE(3,3), B(2,3), BT(3,2), TN(MXPOI)
      DIMENSION  AKH(3,3), QH(3), TK(4), FLUX(MXPOI)
      DIMENSION  DEN(4), SH(4), TH(4)
C
      INTEGER  INTMAT(MXELE,3), LTYPE(MXELE,3), ND(3)
      INTEGER  IDR(10*MXPOI)
C
      DO 2 I=1,NPOIN
      SYSC(I) = 0.
      SYSQ(I) = 0.
    2 CONTINUE
C
C     LOOP OVER THE NUMBER OF ELEMENTS:
C
      DO 5000  IE=1,NELEM
C
C     FIND ELEMENT LOCAL COORDINATES:
C
      ND(1) = IDR(INTMAT(IE,1))
      ND(2) = IDR(INTMAT(IE,2))
      ND(3) = IDR(INTMAT(IE,3))
C
      XG1 = COORD(ND(1),1)
      XG2 = COORD(ND(2),1)
      XG3 = COORD(ND(3),1)
      YG1 = COORD(ND(1),2)
      YG2 = COORD(ND(2),2)
      YG3 = COORD(ND(3),2)
      AREA= 0.5*(XG2*(YG3-YG1) + XG1*(YG2-YG3) + XG3*(YG1-YG2))
      IF(AREA.LE.0.)  WRITE(6,5)  IE
    5 FORMAT(/,'  !!! ERROR !!!  ELEMENT NO.', I5,
     *          ' HAS NEGATIVE OR ZERO AREA ', /,
```

```
     *          '  --- CHECK F.E. MODEL FOR NODAL COORDINATES',
     *          ' AND ELEMENT NODAL CONNECTIONS ---'            )
      IF(AREA.LE.0.)  STOP
C
      B1 = YG2 - YG3
      B2 = YG3 - YG1
      B3 = YG1 - YG2
      C1 = XG3 - XG2
      C2 = XG1 - XG3
      C3 = XG2 - XG1
C
      DO 10  I=1,2
      DO 10  J=1,3
      B(I,J) = 0.
   10 CONTINUE
C
      B(1,1) = B1
      B(1,2) = B2
      B(1,3) = B3
      B(2,1) = C1
      B(2,2) = C2
      B(2,3) = C3
C
      DO 20  I=1,2
      DO 30  J=1,3
      B(I,J)  = B(I,J)/(2.*AREA)
      BT(J,I) = B(I,J)
   30 CONTINUE
   20 CONTINUE
C
C     ZERO ALL COEFFICIENTS OF THE FINAL ELEMENT MATRICES:
C
      DO 60  I=1,3
      DO 50  J=1,3
      AKE(I,J) = 0.
      AKH(I,J) = 0.
   50 CONTINUE
      QH(I) = 0.
   60 CONTINUE
C
C     ELEMENT CONDUCTION MATRIX:
C
      II  = LTYPE(IE,3)
      TKR = TK(II)
      THICK = TH(II)
      DENST = DEN(II)
      SHEAT = SH(II)
C
      DO 100  I=1,3
      DO 100  J=1,3
      AKC(I,J) = 0.
      DO 110  K=1,2
      AKC(I,J) = AKC(I,J) + BT(I,K)*B(K,J)
  110 CONTINUE
      AKC(I,J) = TKR*AREA*THICK*AKC(I,J)
  100 CONTINUE
      DO 120  I=1,3
      DO 120  J=1,3
      AKE(I,J) = AKE(I,J) + AKC(I,J)
  120 CONTINUE
      DO 140 I=1,3
      DUMP = 0.
      DO 130 J=1,3
      DUMP = DUMP + AKE(I,J)*TN(ND(J))
  130 CONTINUE
      SYSQ(ND(I)) = SYSQ(ND(I)) - DUMP
  140 CONTINUE
C
```

```
C     ELEMENT CONVECTION MATRICES:
C
      DO 150 I=1,3
      DO 150 J=1,3
      AKE(I,J) = 0.
  150 CONTINUE
      IF(LTYPE(IE,3).EQ.0)  GO TO 300
      FAC = H*AREA/12.
      DO 230  I=1,3
      DO 230  J=1,3
      AKH(I,J) = FAC
  230 CONTINUE
      DO 240  I=1,3
      AKH(I,I) = 2.*FAC
  240 CONTINUE
      IF(LTYPE(IE,3).EQ.4) THEN
      FAC = H*AREA*TI/3.
      DO 250  I=1,3
      QH(I) = FAC
  250 CONTINUE
      ENDIF
      IF(LTYPE(IE,3).EQ.1) THEN
      SL = SQRT((XG1-XG2)*(XG1-XG2)+(YG1-YG2)*(YG1-YG2))
      FAC = 0.5*H*TI*SL*THICK
      QH(1) = FAC
      QH(2) = FAC
      QH(3) = 0.
      ENDIF
      IF(LTYPE(IE,3).EQ.2) THEN
      SL = SQRT((XG2-XG3)*(XG2-XG3)+(YG2-YG3)*(YG2-YG3))
      FAC = 0.5*H*TI*SL*THICK
      QH(1) = 0.
      QH(2) = FAC
      QH(3) = FAC
      ENDIF
      IF(LTYPE(IE,3).EQ.3) THEN
      SL = SQRT((XG1-XG3)*(XG1-XG3)+(YG1-YG3)*(YG1-YG3))
      FAC = 0.5*H*TI*SL*THICK
      QH(1) = FAC
      QH(2) = 0.
      QH(3) = FAC
      ENDIF
      DO 260  I=1,3
      DO 260  J=1,3
      AKE(I,J) = AKE(I,J) + AKH(I,J)
  260 CONTINUE
      DO 280 I=1,3
      DUMP = 0.
      DO 270 J=1,3
      DUMP = DUMP + AKE(I,J)*TN(ND(J))
  270 CONTINUE
      SYSQ(ND(I)) = SYSQ(ND(I)) - DUMP + QH(I)
  280 CONTINUE
  300 CONTINUE
C
C     ELEMENT HEAT LOAD DUE TO INTERNAL HEAT GENERATION:
C
      IF(LTYPE(IE,1).NE.1)  GO TO 400
      FAC = Q*AREA*THICK/3.
      DO 320  I=1,3
      SYSQ(ND(I)) = SYSQ(ND(I)) + FAC
  320 CONTINUE
  400 CONTINUE
C
C     ELEMENT HEAT LOAD DUE TO SPECIFIED EDGE HEATING:
C
      IF(LTYPE(IE,2).EQ.0)  GO TO 500
      IF (LTYPE(IE,2).EQ.1) THEN
```

```
      SL = SQRT((XG1-XG2)*(XG1-XG2)+(YG1-YG2)*(YG1-YG2))
      FAC = 0.5*QS*SL*THICK
      SYSQ(ND(1)) = SYSQ(ND(1)) + FAC
      SYSQ(ND(2)) = SYSQ(ND(2)) + FAC
      ENDIF
      IF (LTYPE(IE,2).EQ.2) THEN
      SL = SQRT((XG2-XG3)*(XG2-XG3)+(YG2-YG3)*(YG2-YG3))
      FAC = 0.5*QS*SL*THICK
      SYSQ(ND(2)) = SYSQ(ND(2)) + FAC
      SYSQ(ND(3)) = SYSQ(ND(3)) + FAC
      ENDIF
      IF (LTYPE(IE,2).EQ.3) THEN
      SL = SQRT((XG1-XG3)*(XG1-XG3)+(YG1-YG3)*(YG1-YG3))
      FAC = 0.5*QS*SL*THICK
      SYSQ(ND(3)) = SYSQ(ND(3)) + FAC
      SYSQ(ND(1)) = SYSQ(ND(1)) + FAC
      ENDIF
  500 CONTINUE
C
      DO 600 I=1,3
      FAC = DENST*SHEAT*AREA*THICK/3.
      SYSC(ND(I)) = SYSC(ND(I)) + FAC
  600 CONTINUE
C
 5000 CONTINUE
C
C     NODAL HEAT FLUX (W/m):
C
      DO 800 I=1,NPOIN
      IF(FLUX(I).GT.0.) SYSQ(I) = SYSQ(I) + THICK*FLUX(I)
  800 CONTINUE
C
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE SOLVE(NPOIN, NELEM, MXPOI, MXELE, SYSC, SYSQ,
     *                 TS, IBC, TEMP, TN)
C
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION  SYSC(MXPOI), SYSQ(MXPOI)
      DIMENSION  TEMP(MXPOI), TN(MXPOI)
      INTEGER    IBC(MXPOI)
C
      DO 100 I=1,NPOIN
      IF(IBC(I).EQ.0) THEN
      DELT = SYSQ(I)*TS/SYSC(I)
      ENDIF
      IF(IBC(I).EQ.1) THEN
      DELT = 0.
      ENDIF
      TEMP(I) = DELT + TN(I)
  100 CONTINUE
C
      RETURN
      END
```

ภาคผนวก ง

รายละเอียดของโปรแกรมคอมพิวเตอร์ STRESS2DTH

โปรแกรมคอมพิวเตอร์ STRESS2DTH ที่ได้ประดิษฐ์ขึ้นเพื่อวิเคราะห์ปัญหาที่เกิดการยืดหดตัวและความเค้นเนื่องจากอุณหภูมิดังที่ได้กล่าวไว้ในบทที่ 5 มีรายละเอียดดังนี้

```
C     PROGRAM  STRESS2DTH
C
C     A FINITE ELEMENT MECHANICAL/THERMAL STRESS ANALYSIS PROGRAM
C     FOR TWO-DIMENSIONAL PLANE STRUCTURES.
C
C
C     THE VALUES DECLARED IN THE PARAMETER STATEMENT BELOW SHOULD
C     BE ASSIGNED ACCORDING TO THE SIZE OF THE PROBLEMS
C
C        MXPOI = MAXIMUM NUMBER OF NODES IN THE MODEL
C        MXELE = MAXIMUM NUMBER OF ELEMENTS IN THE MODEL
C        MXHBW = MAXIMUM NUMBER OF HALF-BANDWIDTH
C
      PARAMETER (MXPOI=1000, MXELE=2000, MXHBW=3000)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  COORD(MXPOI,2), TEMP(MXPOI), TEXT(20)
      DIMENSION  SYSK(MXPOI*2,MXHBW), SYSF(MXPOI*2)
      DIMENSION  SXX(MXPOI), SYY(MXPOI), SXY(MXPOI), ONE(MXPOI)
      CHARACTER*12  NAME1, NAME2, NAME3, NAME4
C
      INTEGER  INTMAT(MXELE,3), IBC(MXPOI,2)
C
   10 WRITE(6,15)
   15 FORMAT(/, ' PLEASE ENTER THE INPUT FILE NAME:')
      READ(5, '(A)', ERR=10)  NAME1
      OPEN(UNIT=7, FILE=NAME1, STATUS='OLD', ERR=10)
C
C     READ TITLE OF COMPUTATION:
C
      READ(7,*)  NLINES
      DO 100  ILINE=1,NLINES
      READ(7,1)  TEXT
    1 FORMAT(20A4)
  100 CONTINUE
C
C     READ INPUT DATA:
C
      READ(7,1)  TEXT
      READ(7,*)  NPOIN, NELEM, NFORCE, IANA
      IF(NPOIN.GT.MXPOI)  WRITE(6,110)  NPOIN
  110 FORMAT(/,' PLEASE INCREASE THE PARAMETER MXPOI TO ', I5)
      IF(NPOIN.GT.MXPOI)  STOP
      IF(NELEM.GT.MXELE)  WRITE(6,120)  NELEM
  120 FORMAT(/,' PLEASE INCREASE THE PARAMETER MXELE TO ', I5)
      IF(NELEM.GT.MXELE)  STOP
      READ(7,1)  TEXT
      READ(7,*)  ELAS, PR, ALPHA, TREF, THICK
      READ(7,1)  TEXT
      DO 130  IP=1,NPOIN
      READ(7,*)  I, (IBC(I,J), J=1,2), (COORD(I,K), K=1,2), TEMP(I)
      IF(I.NE.IP)  WRITE(6,135)  IP
  135 FORMAT(/, ' NODE NO.', I5, ' IN DATA FILE IS MISSING')
      IF(I.NE.IP)  STOP
  130 CONTINUE
```

```
      READ(7,1)  TEXT
      DO 140  IE=1,NELEM
      READ(7,*)  I, (INTMAT(I,J), J=1,3)
      IF(I.NE.IE)  WRITE(6,150)  IE
  150 FORMAT(/, ' ELEMENT NO.', I5, ' IN DATA FILE IS MISSING')
      IF(I.NE.IE)  STOP
  140 CONTINUE
C
      NDF  = 2
      NDOF = 6
      NEQ  = NPOIN*NDF
      DO 300  I=1,NEQ
      SYSF(I) = 0.
  300 CONTINUE
      READ(7,1)  TEXT
      DO 310  II=1,NFORCE
      READ(7,*)  N, FX, FY
      IEQ = (N-1)*NDF
      SYSF(IEQ+1) = FX
      SYSF(IEQ+2) = FY
  310 CONTINUE
C
C     COMPUTE HALF-BANDWIDTH:
C
      NHBW = 0
      DO 400  IE=1,NELEM
      MIN = 100000
      MAX = 0
      DO 410  IN=1,3
      II = INTMAT(IE,IN)
      IF(II.GT.MAX)  MAX = II
      IF(II.LT.MIN)  MIN = II
  410 CONTINUE
      NDIF = MAX - MIN + 1
      IF(NDIF.GT.NHBW)  NHBW = NDIF
  400 CONTINUE
C
      NHBW = NHBW*NDF
      IF(NHBW.GT.MXHBW)  WRITE(6,420)  NHBW
  420 FORMAT(/,' PLEASE INCREASE THE PARAMETER MXHBW TO ', I5)
      IF(NHBW.GT.MXHBW)  STOP
C
      DO 430  I=1,NEQ
      DO 430  J=1,NHBW
      SYSK(I,J) = 0.
  430 CONTINUE
      WRITE(6,435)  NPOIN, NELEM
  435 FORMAT(/,' *** THE FINITE ELEMENT MODEL CONSISTS OF', I5,
     *         ' NODES AND', I5,' ELEMENTS ***')
C
C     LOOP OVER ALL ELEMENTS TO COMPUTE ELEMENT MATRICES AND ASSEMBLE
C     THEM FOR SYSTEM MATRICES IN THE FORM NEEDED FOR MINIMUM MEMORY
C     REQUIREMENT:
C
      WRITE(6,440)
  440 FORMAT(/,' *** ESTABLISHING ELEMENT MATRICES AND',
     *         ' ASSEMBLING ELEMENT EQUATIONS ***'     )
      CALL CST(NELEM, INTMAT, COORD, ELAS,    PR, ALPHA, THICK,
     *    TREF,   TEMP,  SYSK, SYSF, MXPOI, MXELE, MXHBW, IANA)
C
      WRITE(6,450)
  450 FORMAT(/,' *** APPLYING BOUNDARY CONDITIONS ***')
      CALL APPLYBC(NHBW, NPOIN, IBC, SYSK, SYSF, MXPOI, MXHBW)
C
      WRITE(6,460)
  460 FORMAT(/,' *** SOLVING A SET OF SIMULTANEOUS EQUATIONS',
     *         ' FOR DISPLACEMENT SOLUTIONS ***'              )
      WRITE(6,465)  NEQ, NHBW
```

```
  465 FORMAT(5X,'( TOTAL OF', I5,' EQUATIONS WITH HALF-BANDWIDTH OF',
     *        I4, ' )')
      CALL SOLVE(NEQ, NHBW, SYSK, SYSF, MXPOI, MXHBW)
C
C     COMPUTE NODAL STRESSES:
C
      CALL STRESS(NPOIN, NELEM, INTMAT, COORD, SYSF, ELAS,  PR,
     *            ALPHA,  TREF,   TEMP,   SXX,  SYY,  SXY, ONE,
     *            MXPOI, MXELE, IANA                          )
C
C     PRINT OUT NODAL DISPLACEMENT & STRESS SOLUTIONS:
C
  470 WRITE(6,480)
  480 FORMAT(/, ' PLEASE ENTER FILE NAME FOR DISPLACEMENT'
     *          ' AND STRESS SOLUTIONS:'                    )
      READ(5, '(A)', ERR=470)  NAME2
      OPEN(UNIT=8, FILE=NAME2, STATUS='NEW', ERR=470)
      WRITE(8,490)  NPOIN
  490 FORMAT('  NODAL DISPLACEMENT & STRESS SOLUTIONS [', I5,']:',
     *        //, 2X, 'NODE', 13X, 'U', 13X, 'V', 11X, 'SXX',
     *           11X, 'SYY', 11X, 'SXY', /       )
      I1 = 1
      DO 500  IP=1,NPOIN
      I2 = IP*NDF
      WRITE(8,510)  IP, (SYSF(I), I=I1,I2), SXX(IP), SYY(IP), SXY(IP)
  510 FORMAT(I6, 5E14.6)
      I1 = I2 + 1
  500 CONTINUE
C
C     CREATE FILE FOR 2-D CONTOUR PLOTTING PROGRAM:
C
      IDEF = 0
      NVAR = 3
      IT = 9
  550 WRITE(6,560)
  560 FORMAT(/, ' PLEASE ENTER FILE NAME FOR GRAPHIC DISPLAY:')
      READ(5, '(A)', ERR=550)  NAME3
      OPEN(UNIT=9, FILE=NAME3, STATUS='NEW', ERR=550)
  600 WRITE(IT,610)  NPOIN, NELEM, NVAR
  610 FORMAT('   NPOIN   NELEM    NVAR', /, 3I8)
      WRITE(IT,620)  NPOIN
  620 FORMAT(' NODAL COORDINATES & SOLUTIONS [', I5, ']:')
      DO 630  I=1,NPOIN
      WRITE(IT,640)  I, (COORD(I,J), J=1,2), SXX(I), SYY(I), SXY(I)
  640 FORMAT(I8, 5E12.5)
  630 CONTINUE
      WRITE(IT,650)  NELEM
  650 FORMAT(' ELEMENT NODAL CONNECTIONS [', I5, ']:')
      DO 660  IE=1,NELEM
      WRITE(IT,670)  IE, (INTMAT(IE,J), J=1,3)
  670 FORMAT(4I8)
  660 CONTINUE
      IF(IDEF.EQ.1)  STOP
C
      WRITE(6,700)
  700 FORMAT(/,' CREATE A PLOT FILE WITH DEFORMED SHAPE ?', /,
     *         ' ( 1 = YES,  0 = NO )'                        )
      READ(5,*)  IDEF
      IF(IDEF.NE.1)  STOP
  710 WRITE(6,720)
  720 FORMAT(/,' PLEASE ENTER PLOTTING FILE NAME WITH DEFORMED SHAPE:')
      READ(5, '(A)', ERR=710)  NAME4
      IT = 10
      OPEN(UNIT=IT, FILE=NAME4, STATUS='NEW', ERR=710)
      DISMAX = 0.
      DO 730  IP=1,NPOIN*2
      DISP = SYSF(IP)
      DISP = ABS(DISP)
```

```
      IF(DISP.GT.DISMAX)  DISMAX = DISP
  730 CONTINUE
      XMIN = 1.E20
      XMAX = -XMIN
      YMIN =  XMIN
      YMAX = -YMIN
      DO 740  IP=1,NPOIN
      IF(COORD(IP,1).GT.XMAX)  XMAX = COORD(IP,1)
      IF(COORD(IP,1).LT.XMIN)  XMIN = COORD(IP,1)
      IF(COORD(IP,2).GT.YMAX)  YMAX = COORD(IP,2)
      IF(COORD(IP,2).LT.YMIN)  YMIN = COORD(IP,2)
  740 CONTINUE
      XL = XMAX - XMIN
      YL = YMAX - YMIN
      AL = XL
      IF(YL.GT.AL)  AL = YL
C
C     ASSIGN MAXIMUM PLOTTING DEFORMATION 10% OF MAXIMUM SHAPE:
C
      FAC = 0.1*AL/DISMAX
      DO 750  IP=1,NPOIN
      J1 = 2*IP - 1
      J2 =   J1 + 1
      COORD(IP,1) = COORD(IP,1) + FAC*SYSF(J1)
      COORD(IP,2) = COORD(IP,2) + FAC*SYSF(J2)
  750 CONTINUE
      GO TO 600
C
      STOP
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE APPLYBC(NHBW, NPOIN, IBC, SYSK, SYSF, MXPOI, MXHBW)
C
C     APPLY DISPLACEMENT BOUNDARY CONDITIONS WITH CONDITION CODES OF:
C           0 = FREE TO MOVE
C           1 = FIXED
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  SYSK(MXPOI*2,MXHBW), SYSF(MXPOI*2)
C
      INTEGER  IBC(MXPOI,2)
C
      NDF = 2
      DO 100  IN=1,NPOIN
      DO 200  ID=1,NDF
      IF(IBC(IN,ID).NE.1)  GO TO 200
C
      IEQ = (IN-1)*NDF + ID
      SYSF(IEQ)    = 0.
C
      SYSK(IEQ,1) = 1.
      DO 300  I=2,NHBW
      SYSK(IEQ,I) = 0.
  300 CONTINUE
C
      IF(IEQ.EQ.1)  GO TO 450
      DO 400  N=1,IEQ-1
      IROW = IEQ - N
      ICOL =    N + 1
      IF(ICOL.GT.NHBW)  GO TO 450
      SYSK(IROW,ICOL) = 0.
  400 CONTINUE
  450 CONTINUE
C
  200 CONTINUE
  100 CONTINUE
```

```
C
      RETURN
      END
C
C-------------------------------------------------------------------
C
      SUBROUTINE ASSMBLE(   IE, INTMAT,  SGBL, FGBL, SYSK, SYSF,
     *                   MXPOI,  MXELE, MXHBW                 )
C
C     ASSEMBLE ELEMENT EQUATIONS INTO SYSTEM EQUATIONS
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  SGBL(6,6), FGBL(6)
      DIMENSION  SYSK(MXPOI*2,MXHBW), SYSF(MXPOI*2)
C
      INTEGER  INTMAT(MXELE,3)
C
      NNODE = 3
      NDF   = 2
C
      DO 100  NR=1,NNODE
      NODR = INTMAT(IE,NR)
      DO 100  MR=1,NDF
C
C     DENOTE:  NSR = ROW POSITION IN THE SYSTEM  EQS.
C              NER = ROW POSITION IN THE ELEMENT EQS.
C
      NSR = (NODR-1)*NDF + MR
      NER = (NR  -1)*NDF + MR
      SYSF(NSR) = SYSF(NSR) + FGBL(NER)
C
      DO 200  NC=1,NNODE
      NODC = INTMAT(IE,NC)
      DO 200  MC=1,NDF
C
C     DENOTE:  NSC = COLUMN POSITION IN THE SYSTEM  EQS.
C                    (AFTER ROTATION - READY FOR BANDED SOLVER)
C              NEC = COLUMN POSITION IN THE ELEMENT EQS.
C
      NSC = (NODC-1)*NDF + MC - NSR + 1
      NEC = (NC  -1)*NDF + MC
      IF(NSC.GT.0)
     &     SYSK(NSR,NSC) = SYSK(NSR,NSC) + SGBL(NER,NEC)
  200 CONTINUE
C
  100 CONTINUE
C
      RETURN
      END
C
C-------------------------------------------------------------------
C
      SUBROUTINE CST(NELEM, INTMAT, COORD, ELAS,   PR, ALPHA, THICK,
     *          TREF,   TEMP,  SYSK, SYSF, MXPOI, MXELE, MXHBW, IANA)
C
C     COMPUTE ELEMENT MATRICES AND ASSEMBLE THEM FOR SYSTEM EQUATIONS
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  COORD(MXPOI,2), TEMP(MXPOI)
      DIMENSION  SYSK(MXPOI*2,MXHBW), SYSF(MXPOI*2)
      DIMENSION  SCST(6,6), FCST(6), C(3,3), B(3,6), BT(6,3)
      DIMENSION  DUMA(3,6), DUMB(3), AL(3)
C
      INTEGER  INTMAT(MXELE,3)
C
C     LOOP OVER THE NUMBER OF ELEMENTS:
C
      DO 5000  IE=1,NELEM
```

```
C
C     FIND ELEMENT LOCAL COORDINATES:
C
      II = INTMAT(IE,1)
      JJ = INTMAT(IE,2)
      KK = INTMAT(IE,3)
C
      XG1 = COORD(II,1)
      XG2 = COORD(JJ,1)
      XG3 = COORD(KK,1)
      YG1 = COORD(II,2)
      YG2 = COORD(JJ,2)
      YG3 = COORD(KK,2)
      AREA= 0.5*(XG2*(YG3-YG1) + XG1*(YG2-YG3) + XG3*(YG1-YG2))
      IF(AREA.LE.0.)  WRITE(6,5)  IE
    5 FORMAT(/,'  !!! ERROR !!!  ELEMENT NO.', I5,
     *          ' HAS NEGATIVE OR ZERO AREA ', /,
     *          '  --- CHECK F.E. MODEL FOR NODAL COORDINATES',
     *          ' AND ELEMENT NODAL CONNECTIONS ---'           )
      IF(AREA.LE.0.)  STOP
C
      B1 = YG2 - YG3
      B2 = YG3 - YG1
      B3 = YG1 - YG2
      C1 = XG3 - XG2
      C2 = XG1 - XG3
      C3 = XG2 - XG1
C
      DO 10  I=1,3
      DO 10  J=1,6
      B(I,J) = 0.
   10 CONTINUE
C
      B(1,1) = B1
      B(1,3) = B2
      B(1,5) = B3
      B(2,2) = C1
      B(2,4) = C2
      B(2,6) = C3
      B(3,1) = C1
      B(3,2) = B1
      B(3,3) = C2
      B(3,4) = B2
      B(3,5) = C3
      B(3,6) = B3
C
      DO 20  I=1,3
      DO 30  J=1,6
      B(I,J)  = B(I,J)/(2.*AREA)
      BT(J,I) = B(I,J)
   30 CONTINUE
   20 CONTINUE
C
C     ELASTICITY MATRIX:
C
C     IANA = 1 FOR PLANE STRESS
C          = 2 FOR PLANE STRAIN
C
      IF(IANA.EQ.1) EXPV = 0.
      IF(IANA.EQ.2) EXPV = PR
      IF(IANA.EQ.1) FAC = ELAS/(1.-PR*PR)
      IF(IANA.EQ.2) FAC = ELAS/(1+PR)/(1-2*PR)
      C(1,1) = FAC*(1-EXPV)
      C(1,2) = FAC*PR
      C(1,3) = 0.
      C(2,1) = C(1,2)
      C(2,2) = C(1,1)
      C(2,3) = 0.
```

```
      C(3,1) = 0.
      C(3,2) = 0.
      C(3,3) = FAC*(1.-PR-EXPV)/2.
C
C     ELEMENT STIFFNESS MATRIX:
C
      DO 100  I=1,3
      DO 100  J=1,6
      DUMA(I,J) = 0.
      DO 200  K=1,3
      DUMA(I,J) = DUMA(I,J) + C(I,K)*B(K,J)
  200 CONTINUE
  100 CONTINUE
C
      DO 300  I=1,6
      DO 300  J=1,6
      SCST(I,J) = 0.
      DO 400  K=1,3
      SCST(I,J) = SCST(I,J) + BT(I,K)*DUMA(K,J)
  400 CONTINUE
  300 CONTINUE
C
      DO 500  I=1,6
      DO 500  J=1,6
      SCST(I,J) = SCST(I,J)*THICK*AREA
  500 CONTINUE
C
C     ELEMENT NODAL FORCE DUE TO IN-PLANE THERMAL EXPANSION:
C
      AL(1) = ALPHA*(1+EXPV)
      AL(2) = ALPHA*(1+EXPV)
      AL(3) = 0.
      DO 600  I=1,3
      DUMB(I) = 0.
      DO 700  J=1,3
      DUMB(I) = DUMB(I) + C(I,J)*AL(J)
  700 CONTINUE
  600 CONTINUE
C
      DO 800  I=1,6
      FCST(I) = 0.
      DO 900  J=1,3
      FCST(I) = FCST(I) + BT(I,J)*DUMB(J)
  900 CONTINUE
  800 CONTINUE
C
C     AVERAGE ELEMENT TEMPERATURE:
C
      TAVG = (TEMP(II) + TEMP(JJ) + TEMP(KK))/3.
C
      FAC = (TAVG - TREF)*THICK*AREA
      DO 1000  I=1,6
      FCST(I) = FCST(I)*FAC
 1000 CONTINUE
C
C     ASSEMBLE THESE ELEMENT EQUATIONS INTO THE SYSTEM EQUATIONS:
C
      CALL ASSMBLE(   IE, INTMAT,  SCST, FCST, SYSK, SYSF,
     *               MXPOI,  MXELE, MXHBW                    )
C
 5000 CONTINUE
C
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE SOLVE(NROW, NHBW, GSTIF, XL, MXPOI, MXHBW)
```

```
C
C     SOLVE A SET OF SIMULTANEOUS EQUATIONS USING GAUSS ELIMINATION.
C     THIS SOLVER ROUTINE CAN BE DESCRIBED BY USING AN EXAMPLE OF A
C     SET OF FOUR SIMULTANEOUS EQUATIONS (AFTER APPLYING BOUNDARY
C     CONDITIONS) AS SHOWN BELOW:
C
C         [ A11   A12   A13     0 ]    [ X1 ]          [ F1 ]
C         [                       ]    [    ]          [    ]
C         [ A12   A22   A23   A24 ]    [ X2 ]          [ F2 ]
C         [                       ]    [    ]     =    [    ]
C         [ A13   A23   A33   A34 ]    [ X3 ]          [ F3 ]
C         [                       ]    [    ]          [    ]
C         [   0   A24   A34   A44 ]    [ X4 ]          [ F4 ]
C
C     WHERE THE VARIABLE XL IS THE LOAD VECTOR ON RHS OF THE EQUATIONS.
C     THE GLOBAL STIFFNESS MATRIX ABOVE IS STORED IN THE VARIABLE
C     GSTIF IN THE FORMAT SHOWN BELOW: (HERE  NROW = 4  AND  NHBW = 3)
C
C                             [ A11   A12   A13 ]
C                             [                 ]
C                             [ A22   A23   A24 ]
C          [ GSTIF ]    =     [                 ]
C                             [ A33   A34     0 ]
C                             [                 ]
C                             [ A44     0     0 ]
C
C     AND THE OUTPUT SOLUTIONS WILL BE STORED IN THE VARIABLE XL.
C
      IMPLICIT REAL*8(A-H,O-Z)
C
      DIMENSION  GSTIF(MXPOI*2,MXHBW), XL(MXPOI*2)
C
      NR=NROW
      NC=NHBW
C
C     DIAGONALIZATION THE MATRIX:
C
      DO 10 I=1,NR
      PIVOT1=GSTIF(I,1)
      IF(ABS(PIVOT1).LT.10.E-10) THEN
      WRITE(6,1025) I, PIVOT1
 1025 FORMAT(' EQ. NO.', I5, ' HAS NEARLY ZERO PIVOT OF', E14.6,
     *        '  ** STOP **', //,
     *        ' *** CHECK NODE AND ELEMENT NUMBERING IN F.E. MODEL ***')
      STOP
      ENDIF
C
      XL(I)=XL(I)/PIVOT1
      DO 20 J=1,NC
   20 GSTIF(I,J)=GSTIF(I,J)/PIVOT1
      MM=0
      DO 30 II=I+1,NR
      MM=MM+1
      IF(MM+1.GT.NC) GOTO 30
      PIVOT2=GSTIF(I,MM+1)*PIVOT1
      XL(II)=XL(II)-XL(I)*PIVOT2
      DO 40 JJ=1,NC
      JJJ=JJ+MM
      IF(JJJ.LE.NC)
     &  GSTIF(II,JJ)=GSTIF(II,JJ)-GSTIF(I,JJJ)*PIVOT2
   40 CONTINUE
   30 CONTINUE
   10 CONTINUE
C
C     BACK SUBSTITUTION:
C
      DO 70 I=NR-1,1,-1
      II=1
```

```
      DO 80 J=I+1,NR
      II=II+1
      IF(II.LE.NHBW) XL(I)=XL(I)-GSTIF(I,II)*XL(J)
   80 CONTINUE
   70 CONTINUE
C
      RETURN
      END
C
C-----------------------------------------------------------------
C
      SUBROUTINE STRESS(NPOIN, NELEM, INTMAT, COORD, DISP, ELAS,  PR,
     *                  ALPHA,  TREF,   TEMP,   SXX,  SYY,  SXY, ONE,
     *                  MXPOI, MXELE, IANA                          )
C
C     COMPUTE NODAL STRESS COMPONENTS FOR CST ELEMENTS
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION  COORD(MXPOI,2), TEMP(MXPOI)
      DIMENSION  DISP(MXPOI*2), ONE(MXPOI)
      DIMENSION  SXX(MXPOI), SYY(MXPOI), SXY(MXPOI)
      DIMENSION  C(3,3), B(3,6), EPS(3), UG(3), VG(3)
C
      INTEGER  INTMAT(MXELE,3)
C
      DO 10  I=1,NPOIN
      SXX(I) = 0.
      SYY(I) = 0.
      SXY(I) = 0.
      ONE(I) = 0.
   10 CONTINUE
C
C     LOOP OVER THE NUMBER OF ELEMENTS:
C
      DO 1000  IE=1,NELEM
C
C     FIND ELEMENT LOCAL COORDINATES:
C
      II = INTMAT(IE,1)
      JJ = INTMAT(IE,2)
      KK = INTMAT(IE,3)
C
      XG1 = COORD(II,1)
      XG2 = COORD(JJ,1)
      XG3 = COORD(KK,1)
      YG1 = COORD(II,2)
      YG2 = COORD(JJ,2)
      YG3 = COORD(KK,2)
      AREA= 0.5*(XG2*(YG3-YG1) + XG1*(YG2-YG3) + XG3*(YG1-YG2))
C
      B1 = YG2 - YG3
      B2 = YG3 - YG1
      B3 = YG1 - YG2
      C1 = XG3 - XG2
      C2 = XG1 - XG3
      C3 = XG2 - XG1
C
      DO 110  I=1,3
      DO 110  J=1,6
      B(I,J) = 0.
  110 CONTINUE
C
      B(1,1) = B1
      B(1,3) = B2
      B(1,5) = B3
      B(2,2) = C1
      B(2,4) = C2
      B(2,6) = C3
```

```
      B(3,1) = C1
      B(3,2) = B1
      B(3,3) = C2
      B(3,4) = B2
      B(3,5) = C3
      B(3,6) = B3
C
      DO 120  I=1,3
      DO 130  J=1,6
      B(I,J) = B(I,J)/(2.*AREA)
  130 CONTINUE
  120 CONTINUE
C
C     ELASTICITY MATRIX:
C
C     IANA = 1 FOR PLANE STRESS
C          = 2 FOR PLANE STRAIN
C
      IF(IANA.EQ.1) EXPV = 0.
      IF(IANA.EQ.2) EXPV = PR
      IF(IANA.EQ.1) FAC = ELAS/(1.-PR*PR)
      IF(IANA.EQ.2) FAC = ELAS/(1+PR)/(1-2*PR)
      C(1,1) = FAC*(1-EXPV)
      C(1,2) = FAC*PR
      C(1,3) = 0.
      C(2,1) = C(1,2)
      C(2,2) = C(1,1)
      C(2,3) = 0.
      C(3,1) = 0.
      C(3,2) = 0.
      C(3,3) = FAC*(1.-PR-EXPV)/2.
C
C     GATHER ELEMENT NODAL DISPLACEMENTS:
C
      DO 200  J1=1,3
      I1  = INTMAT(IE,J1)
      IEQ = (I1-1)*2 + 1
      UG(J1) = DISP(IEQ  )
      VG(J1) = DISP(IEQ+1)
  200 CONTINUE
C
C     COMPUTE THE TOTAL STRAINS:
C
      DO 220  I=1,3
      EPS(I) = 0.
      DO 230  J=1,3
      J1 = (J-1)*2 + 1
      J2 = J1 + 1
      EPS(I) = EPS(I) + B(I,J1)*UG(J) + B(I,J2)*VG(J)
  230 CONTINUE
  220 CONTINUE
C
C     COMPUTE THERMAL STRAINS USING AVERAGE ELEMENT NODAL TEMPERATURES:
C
      TAVG = (TEMP(II) + TEMP(JJ) + TEMP(KK))/3.
C
C     COMPUTE THE NET STRAINS:
C
      EPS(1) = EPS(1) - ALPHA*(1+EXPV)*(TAVG - TREF)
      EPS(2) = EPS(2) - ALPHA*(1+EXPV)*(TAVG - TREF)
C
C     COMPUTE THE ELEMENT STRESSES:
C
      SXXE = C(1,1)*EPS(1) + C(1,2)*EPS(2) + C(1,3)*EPS(3)
      SYYE = C(2,1)*EPS(1) + C(2,2)*EPS(2) + C(2,3)*EPS(3)
      SXYE = C(3,1)*EPS(1) + C(3,2)*EPS(2) + C(3,3)*EPS(3)
C
C     COMPUTE NODAL STRESSES FROM ELEMENT STRESSES:
```

```
C
      SXX(II) = SXX(II) + SXXE
      SXX(JJ) = SXX(JJ) + SXXE
      SXX(KK) = SXX(KK) + SXXE
      SYY(II) = SYY(II) + SYYE
      SYY(JJ) = SYY(JJ) + SYYE
      SYY(KK) = SYY(KK) + SYYE
      SXY(II) = SXY(II) + SXYE
      SXY(JJ) = SXY(JJ) + SXYE
      SXY(KK) = SXY(KK) + SXYE
      ONE(II) = ONE(II) + 1.
      ONE(JJ) = ONE(JJ) + 1.
      ONE(KK) = ONE(KK) + 1.
C
 1000 CONTINUE
C
C     COMPUTE NODAL STRESSES:
C
      DO 1100  I=1,NPOIN
      IF(ONE(I).EQ.0.)  WRITE(6,1200)  I
 1200 FORMAT(' *** WARNING ***  NO STRESS CONTRIBUTION AT NODE', I5)
      IF(ONE(I).EQ.0.)  ONE(I) = 1.
      SXX(I) = SXX(I)/ONE(I)
      SYY(I) = SYY(I)/ONE(I)
      SXY(I) = SXY(I)/ONE(I)
 1100 CONTINUE
C
      RETURN
      END
```

ประวัติผู้เขียนวิทยานิพนธ์

นายวิโรจน์ ลิ่มตระการ เกิดเมื่อวันที่ 10 เดือนเมษายน พุทธศักราช 2511 จังหวัดสงขลา สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรมหาบัณฑิตจากภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2539 เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรดุษฎีบัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2541