รายการอ้างอิง

1. ปราโมทย์ เดชะอำไพ.ไฟไนต์เอลิเมนต์ในงานวิศวกรรม.พิมพ์ครั้งที่ 2. กรุงเทพฯ: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2542.
2. Anderson, J. D. Jr. Modern Compressible Flow with Historical Perspective. Third Edition. New York: McGraw-Hill, 2003.
3. Anderson, J. D. Jr. Fundamentals of Aerodynamics. Third Edition. New York: McGraw-Hill, 2001.
4. ปราโมทย์ เดชะอำไพ. ระเบียบวิธีไฟไนต์เอลิเมนต์เพื่อการคำนวณพลศาสตร์ของไหล. พิมพ์ครั้งที่ 1. กรุงเทพฯ: สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2545.
5. Löhner , R., Morgan, K. and Zienkiewicz, O.C. An Adaptive Finite Element Procedure for Compressible High Speed Flows. Computer Methods in Applied Mechanics and Engineering 51 (1985): 441-465.
6. ปัญญา จันทร์ไพแสง. ระเบียบวิธีไฟไนต์เอลิเมนต์สำหรับการไหลความเร็วสูงแบบอัดตัวได้. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2541.
7. Brueckner, F .P. and Heinrich, J. C. Petrov-Galerkin Finite Element Model for Compressible Flows. International Journal for Numerical Methods in Engineering 32 (1991): 255-274.
8. Taghaddosi, F., Habashi, W. G., Guevrremont, G. and Ait-Ali-Yahia, D. An Adaptive Least-Squares Method for The Compressible Euler Equations. International Journal for Numerical Methods in Fluids 31 (1999): 1121-1139.
9. Zienkiewicz, O. C., Nithiarasu, P., Codina, R., Vazquez, M. and Ortiz, P. The Characteristic-Based-Split Procedure: An Efficient and Accurate Algorithm for Fluid Problems. International Journal for Numerical Methods in Fluids 31 (1999): 359-392.
10. Zienkiewicz, O. C. and Taylor, R. L. The Finite Element Method. Fifth Edition, Volume III. Butterworth-Heinemann, 2000.
11. สุพัฒนพงศ์ สิขาบัณฑิต. เทคนิคการปรับขนาดไฟไนต์เอลิเมนต์เพื่อการวิเคราะห์การไหลแบบหนืด. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2541.

12. Zienkiewicz, O.C., Szmelter, J. and Peraire, J. Compressible and Incompressible Flow; An Algorithm for All Seasons. Computer Methods in Applied Mechanics and Engineering 78 (1990): 105-121.

13. Zienkiewicz, O.C. and Wu, J. A General Explicit or Semi-Explicit Algorithm for Compressible and Incompressible Flows. International Journal for Numerical Methods in Engineering 35 (1992): 457-479.

14. Dechaumphai, P. and Limtrakarn, W. Adaptive Cell-Centered Finite Element Technique for Compressible Flows. Journal of Energy, Heat and Mass Transfer 21 (1999): 57-65.

15. สุทธิศักดิ์ พงศ์ธนาพาณิช. การพัฒนาโปรแกรมคอมพิวเตอร์สำหรับการไหลไม่คงตัวความเร็วสูงแบบอัดตัวได้และไร้ความหนืดในสองมิติด้วยระเบียบวิธีไฟไนต์เอลิเมนต์. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2544.

16. Limtrakarn, W. and Dechaumphai, P. Computations of High-Speed Compressible Flows With Adaptive Cell-Centered Finite Element Method. Journal of the Chinese Institute of Engineers 26 (2003): 553-563.

17. วิโรจน์ ลิ่มตระการ. ระเบียบวิธีไฟไนต์เอลิเมนต์สำหรับปฏิสัมพันธ์ระหว่างการไหลความเร็วสูงและโครงสร้าง. วิทยานิพนธ์ปริญญาดุษฎีบัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2545.

18. Anderson, J. D. Jr. Computational Fluid Dynamics, The Basics with Applications. International Edition. Singapore: McGraw-Hill, 1995.

19. Zienkiewicz, O.C. and Codina, R. A General Algorithm for Compressible and Incompressible flow – Part I . The Split, Characteristic-Based Scheme. International Journal for Numerical Methods in Fluids 20 (1995): 869-885.

20. Zienkiewicz, O. C., Morgan, K., Satya Sai, B.V.K. A General Algorithm for Compressible and Incompressible flow – Part II .Tests On The Explicit Form. International Journal for Numerical Methods in Fluids 20 (1995): 887-913.

21. Codina, R., Vazquez, M. and Zienkiewicz, O. C. A General Algorithm for Compressible and Incompressible flow – Part III .The Semi-Implicit Form. International Journal for Numerical Methods in Fluids 27 (1998): 13-32.

22. Nithiarasu, P., Zienkiewicz, O.C., Satya Sai, B.V.K., Morgan, K., Codina, R. and Vazquez, M. Shock Capturing Viscosities for The General Fluid Mechanics Algorithm. International Journal for Numerical Methods in Fluids 28 (1998): 1325-1353.

23. Löhner , R., Morgan, K. and Zienkiewicz, O. C. The Solution of Non-Linear Hyperbolic Equation System by The Finite Element Method International Journal for Numerical Methods in Fluids 4 (1984): 1043-1063

24. Satya Sai, B.V.K., Zienkiewicz, O. C., Manzari, M.T., Lyra, P.R.M. and Morgan, K. General Purpose Versus Special Algorithms for High-Speed Flows with Shocks. International Journal for Numerical Methods in Fluids 27 (1998): 57-80.

25. Nithiarasu, P., Zienkiewicz, O. C. On Stabilization of The CBS Algorithm: Internal and External Time Steps. International Journal for Numerical Methods in Engineering 48 (2000): 875-880.

26. Nithiarasu, P. On Boundary Conditions of The Characteristic Based Split (CBS) Algorithm for Fluid Dynamics. International Journal for Numerical Methods in Engineering 54 (2002): 523-536.

27. Nithiarasu, P. An Efficient Artificial Compressibility (AC) Scheme Based On The Characteristic based split (CBS) Method for Incompressible flows. International Journal for Numerical Methods in Engineering 56 (2003): 1815-1845.

28. Zienkiewicz, O. C. and Wu, J. Incompressibility Without Tears-How to Avoid Restrictions of Mixed Formulation. International Journal for Numerical Methods in Engineering 32 (1991): 1189-1203.

29. Xikui Li , Wenhua Wu and Zienkiewicz, O.C. Implicit Characteristic Galerkin Method for Convection-Diffusion Equations. International Journal for Numerical Methods in Engineering 47 (2000): 1689-1708.

30. Xikui Li , Wenhua Wu and Zienkiewicz, O. C. Implicit Characteristic Galerkin Method for Convection-Diffusion Equations. International Journal for Numerical Methods in Engineering 47 (2000): 1689-1708.

31. Donea, J. A Taylor-Galerkin Method for Convective Transport Problems. International Journal for Numerical Methods in Engineering 20 (1984): 101-119.

32. Hodge, B. K. and Keioth Koenig. Compressible Fluid Dynamics with Personal Computer Applications. International Edition, United States of America: Prentice-Hall, 1995.

33. Oosthuizen, P. H. and Carscallen, W.E. Compressible Fluid Flow. International Edition, Singapore: McGraw-Hill, 1997.

34. Dechaumphai, P. Adaptive Finite Element Technique for Heat Transfer Problems. Journal of Energy, Heat and Mass Transfer 17 (1995): 87-94.

35. Dechaumphai, P. Adaptive Finite Element Technique for Thermal Stress Analysis of Built-Up Structures. JSME International Journal 39 (1996): 223-230.

36. Dechaumphai, P and Phongthanapanich, S High-Speed Compressible Flow Solutions by Adaptive Cell-Centered Upwinding Algorithm with Modified H-Correction Entropy Fix. Advances in Engineering Software 34 (2003): 533-538.

37. Limtrakarn, W. and Dechaumphai, P. An Explicit Finite Element Method for Inviscid Compressible Flow. The Fifteeth Annual National Mechanical Engineering Conference, Bangkok,November 28-30, 2000.

38. Hosseini, R., Rahimian, M. H., Mirzaei, M. Performance of High-Accuracy Schemes in Inviscid Fluxes Calculation. [Online] Available from: http://tetra.mech.ubc.ca/CFD03/papers/paper30AD6.pdf

39. Dena Hendriana and Klaus Jurgen Bathe. On a Parabolic Quadrilateral Finite Element for Compressible Flows. Computer Methods in Applied Mechanics and Engineering 186 (2000): 1-22.

40. Dechaumphai, P. and Wieting, A. R. Couple Fluid-Thermal-Structural Analysis for Aerodynamically Heated Structures. 7th International Conference on Finite Element Methods in Flow Problems, Alabama, April 3-7, 1989.

41. Huebner, H.K., Thronton, E.A. and Byrom, T.G. The Finite Element Method for Engineers. Third Edition. New York: John Wiley & Sons, 1995.

42. Murthy, T.K.S. Computational Methods in Hypersonic Aerodynamics. Great Britain:, Computational Mechanics Publications, 1991.

ภาคผนวก

ภาคผนวก ก

รายละเอียดของโปรแกรมคอมพิวเตอร์ CBSHIFLOW

โปรแกรมคอมพิวเตอร์ CBSHIFLOW ที่ได้ประดิษฐ์ขึ้นเพื่อการวิเคราะห์ปัญหาการไหลความเร็วสูงแบบไร้ความหนืดที่ได้กล่าวไว้ในบทที่ 4 มีรายละเอียดดังนี้

```
C************************************************************************ C
C
C       PROGRAM CBSHIFLOW
C
C       A FINITE ELEMENT COMPUTER PROGRAM FOR SOLVING EULER EQUATIONS
C       FOR TWO-DIMENTIONAL INVISID HIGH-SPEED COMPRESSIBLE FLOWS
C       BASED ON THE CHARACTERISTIC BASED SPLIT (CBS) ALGORITHM
C       FULLY EXPLICIT,LINEAR TRIANGULAR FINITE ELEMENTS
C
C
C
C************************************************************************ C

      IMPLICIT  REAL*8(A-H,O-Z)
      INTEGER   MXBOU,MXELE,MXPOI
      PARAMETER (MXPOI=200000,MXELE=400000,MXBOU=5000)

      INTEGER(2) TMPDAYS    ,TMPMONTHS    ,TMPYEARS
      INTEGER(2) TMPHOURS   ,TMPMINUTES   ,TMPSECONDS   ,TMPHUNDS
      INTEGER(2) TMPDAYE    ,TMPMONTHE    ,TMPYEARE
      INTEGER(2) TMPHOURE   ,TMPMINUTEE   ,TMPSECONDE   ,TMPHUNDE

      INTEGER   IHELP(MXPOI)     ,IWPOIN(3,MXBOU)  ,NCMAX(MXPOI)
      INTEGER   MATCON(MXPOI,20),INTMA(3,MXELE)    ,ISIDO(4,MXBOU)
      INTEGER   IELSI(2,MXELE)   ,ICOUNT(4)        ,IHELP2(MXPOI)

      REAL*8    COORD(2,MXPOI)   ,GEOME(7,MXELE)   ,DMMAT(MXPOI)
      REAL*8    UNKNO(4,MXPOI)   ,UNKN1(4,MXPOI)   ,RSIDO(3,MXBOU)
      REAL*8    PRES(MXPOI)      ,PRES1(MXPOI)     ,SOUND(MXPOI)
      REAL*8    RHS0(4,MXPOI)    ,RHS2(4,MXPOI)    ,RHS1(4,MXPOI)
      REAL*8    FXSEC(4,MXELE)   ,FYSEC(4,MXELE)   ,PSWE(MXPOI)
      REAL*8    DELTP(MXPOI)     ,DELTE(MXELE)     ,WNOR(2,MXBOU)
      REAL*8    CONIN(3)         ,CINF(5)          ,THETA(2)
      REAL*8    TT1(MXPOI)       ,CINF2(5)
      REAL*8    ALEN(MXPOI)      ,ABSV(MXPOI)      ,AMACH(MXPOI)
      REAL*8    HA(4)            ,HB(4)   ,SUMER(4) ,SUMSQ(4)

      CHARACTER*4  CV
      CHARACTER*40 TEXT  ,NAME1

C *** ENTER INPUT FILE NAME

      WRITE(*,1)
    1 FORMAT(/,'BISMINLA   HIJROHMAN   NIDROGHIM',/)
    5 WRITE(*,10)
   10 FORMAT(/, 'PLEASE ENTER THE INPUT FILE NAME:')
      READ(*, '(A)', ERR=5)  NAME1

C *** OPEN FILE

      OPEN( UNIT= 5, FILE=  NAME1    ,STATUS='OLD'    ,ERR=5 )
      OPEN( UNIT= 7, FILE= 'ERROR_AV_1_1.EAV',STATUS='UNKNOWN',ERR=5 )
      OPEN( UNIT= 9, FILE= 'SOLUTION_1_1.OUT',STATUS='UNKNOWN',ERR=5 )

C *** STORE START TIME

      CALL GETDAT(TMPYEARS ,TMPMONTHS  ,TMPDAYS)
      CALL GETTIM(TMPHOURS ,TMPMINUTES ,TMPSECONDS ,TMPHUNDS)
```

```
      WRITE(7,*)'START TIME:'
      WRITE (7, 15) TMPMONTHS, TMPDAYS, TMPYEARS
   15 FORMAT(I2, '/', I2.2, '/', I4.4)
      WRITE (7, 20) TMPHOURS,TMPMINUTES,TMPSECONDS,TMPHUNDS
   20 FORMAT(I2, ':', I2.2, ':', I2.2, ':', I2.2, ' ',A)

C *** READIND NUMBER OF ELEMENTS, NUMBER OF NODES, NUMBER OF BOUNDARY SIDES

      READ(5,*) LINES
      DO L = 1,LINES
               READ(5,2) TEXT
      ENDDO !L
      READ(5,2) TEXT
    2 FORMAT(A)
      READ(5,*) NELEM, NPOIN, NBOUN

C *** CHECKING BASIC ARRAY-SIZE ALLOCATION

      IF(NPOIN.GT.MXPOI)THEN
               WRITE(*,4) NPOIN
    4          FORMAT(/,'PLEASE INCREASE THE PARAMETER MXPOI TO ',I6)
               STOP
      ENDIF
      IF(NELEM.GT.MXELE)THEN
               WRITE(*,6) NELEM
    6          FORMAT(/,'PLEASE INCREASE THE PARAMETER MXELE TO ',I6)
               STOP
      ENDIF
      IF(NBOUN.GT.MXBOU)THEN
               WRITE(*,7) NBOUN
    7          FORMAT(/,'PLEASE INCREASE THE PARAMETER MXBOUN TO ',I6)
               STOP
      ENDIF
C
      WRITE(*,25)
   25 FORMAT(/,'THE FINITE ELEMENT MODEL CONSISTS OF :')
      WRITE(*,30)NPOIN
   30 FORMAT('    NUMBER OF NODES       = ', I6)
      WRITE(*,35)NELEM
   35 FORMAT('    NUMBER OF ELEMENTS    = ', I6)
      WRITE(*,40)NBOUN
   40 FORMAT('    NUMBER OF BOUNDARY    = ', I6,/)

C *** CALL SUBROUTINE READ INPUT DATA

      CALL INPUT(MXPOI ,MXELE ,MXBOU ,NPOIN  ,NELEM ,NBOUN ,IOPT  ,
     &           INTMA ,COORD ,UNKNO ,ISIDO  ,CONIN ,CINF  ,NTIME ,
     &           ISTEP ,TIMT  ,ILOTS ,IWRITE ,DTFIX ,CSAFM ,THETA ,
     &           CC    ,TINF  ,SOUND ,TT1    ,PRES  ,PRES1 ,METHOD,
     &           CSMOO ,NSMOO ,PINF  ,CC2    ,PINF2 ,TINF2 ,CINF2 )
        WRITE(*,*)'*******************************************************'

      GAMMA = CONIN(1)
      GAM1  = GAMMA - 1.0D+00

C *** MARKING THE ELEMENT WITH ONE OR TW0 OR NO BOUNDARY SIDES

      DO J = 1, NELEM
               DO I = 1, 2
                        IELSI(I,J) = 0
               ENDDO !I
      ENDDO !J
      DO I = 1, NBOUN
               IE = ISIDO(3,I)
               IF(IELSI(1,IE).EQ.0) IELSI(1,IE) = I
               IF(IELSI(1,IE).NE.I) IELSI(2,IE) = I
      ENDDO !I

C ***  CONVERT THE VARIABLES TO CONSERVATION FORM

       DO IP = 1, NPOIN
              DO IA = 2, 4
                       UNKNO(IA,IP) = UNKNO(1,IP)*UNKNO(IA,IP)
```

```
              ENDDO !IA
          ENDDO !IP

C *** PRELIMINARY INPUT DATA

      CALL PRELIM(MXPOI,MXELE,MXBOU,NPOIN,NELEM,NBOUN,INTMA,NCMAX,
     &            MATCON,IHELP,ILOTS,COORD ,GEOME,ALEN,DMMAT,ISIDO,
     &            RSIDO,IWPOIN,WNOR ,NWALL)

C *** OBTAIN THE DIRECTION COSINE OF BOUNDARY AND NORMAL WALL VECTOR

      CALL GETBOUN(MXPOI  ,MXBOU ,NPOIN ,NBOUN ,ISIDO ,IHELP ,RSIDO ,
     &             IWPOIN ,WNOR  ,NWALL ,COORD )
      WRITE(*,85)
   85 FORMAT(/,'PREPARED ALL DATA OF FINITE ELEMENT MODEL COMPLETELY',/)
      WRITE(*,*)'*******************************************************'
C
      WRITE(*,90)
      WRITE(7,90)
   90 FORMAT(/,2X,'ITERATION NO.',5X,'DEL-RHO',5X,'DEL RHO-U',5X,
     &'DEL RHO-V',5X,'DEL RHO-E',/)

C **********************TIME ITERATIONS START NOW ***************************C

      NSTEP = 1
      DO ITIME = NSTEP, NTIME
               INTIME = ITIME - NSTEP + 1
               DO IP = 1, NPOIN
                      DO IA = 1, 4
                             UNKN1(IA,IP) = UNKNO(IA,IP)
                      ENDDO !IA
               ENDDO !IP
C ***   <STEP 1>
C ***   CALCULATE OF THE INTEMEDIATE VELOCITIES Ui[*] IN X,Y DIRECTIONS
               CALL STEP1(MXPOI ,MXELE ,MXBOU ,NPOIN ,NELEM ,NBOUN ,
     &                    INTMA ,GEOME ,DMMAT ,UNKNO ,RSIDO ,ISIDO ,
     &                    RHS0  ,RHS1  ,RHS2  ,PSWE  ,IHELP ,DELTP ,
     &                    DELTE ,SOUND ,CONIN ,NCMAX ,ITIME ,INTIME,
     &                    CINF  ,THETA ,FXSEC ,FYSEC ,PRES  ,PRES1 ,
     &                    CC    ,CSAFM ,DTFIX ,ILOTS ,TT1   ,ALEN  ,
     &                    NITER ,MATCON,CC2   ,CINF2 )

C *** <STEP2>
C *** CALCULATE OF THE PRESSURE (EXPLICIT SOLVER ONLY)

               CALL STEP2(MXPOI ,MXELE ,MXBOU ,NPOIN ,NBOUN ,NELEM ,INTMA,
     &                    ISIDO ,RSIDO ,GEOME ,DMMAT ,UNKNO ,UNKN1 ,RHS1 ,
     &                    RHS2  ,IELSI ,PRES  ,DELTP ,DELTE ,THETA ,NITER )

C *** <STEP3>
C *** CALCULATE  OF THE  CORRECT VELOCITES

               CALL STEP3(MXPOI ,MXELE ,MXBOU ,NPOIN ,NELEM ,NBOUN ,INTMA ,
     &                    ISIDO ,RSIDO ,GEOME ,DMMAT ,UNKNO ,UNKN1 ,IELSI ,
     &                    RHS2  ,DELTP ,DELTE ,THETA ,PRES  ,PRES1 )

C *** STEP<4>
C *** CALCULATE ENERGY EQUATION

               CALL STEP4(MXPOI ,MXELE ,MXBOU ,NPOIN ,NELEM ,NBOUN  ,INTMA ,
     &                    GEOME ,UNKN1 ,RHS2  ,DELTE ,FXSEC ,FYSEC  ,PRES  ,
     &                    ISIDO ,RSIDO ,RHS0  ,TT1   ,DELTP ,UNKNO  ,INTIME,
     &                    NITER ,DMMAT ,RHS1 )

C *** CALCULATE PRESSURE
C *** APPLY BOUNDARY CONDITION

               CALL GETPRES(MXPOI ,MXBOU ,NPOIN ,NBOUN ,UNKNO ,SOUND ,CONIN,
     &                      CINF  ,ISIDO ,PRES1 ,CC    ,CC2   ,CINF2 )
               CALL BOUND(MXPOI ,MXBOU ,NPOIN ,NBOUN  ,ISIDO ,CINF ,UNKNO ,
     &                    CINF2 ,NWALL ,WNOR  ,IWPOIN ,UNKN1 ,SOUND ,CONIN,
     &                    PRES1 ,CC    ,CC2   ,TINF   ,TINF2 ,RSIDO )
```

```
C *** ADD ARTIFICIAL VISCOCITY BASED ON SECOND GRADIRENT OF PRESSURE

              IF (CONIN(2).GT.0.0) THEN

                     CALL ARTVIS(MXPOI, MXELE, NPOIN, NELEM, INTMA, PRES ,CONIN ,
     &                           GEOME, DMMAT, UNKNO, RHS1 , SOUND, ALEN )
                     DO IP = 1, NPOIN
                            UNKNO(1,IP)=UNKNO(1,IP)+DELTP(IP)*DMMAT(IP)*RHS1(1,IP)
                            UNKNO(2,IP)=UNKNO(2,IP)+DELTP(IP)*DMMAT(IP)*RHS1(2,IP)
                            UNKNO(3,IP)=UNKNO(3,IP)+DELTP(IP)*DMMAT(IP)*RHS1(3,IP)
                            UNKNO(4,IP)=UNKNO(4,IP)+DELTP(IP)*DMMAT(IP)*RHS1(4,IP)
                     ENDDO !IP
              ENDIF

C ***  APPLY BOUNDARY CONDITION AGAIN

              CALL GETPRES(MXPOI,MXBOU ,NPOIN ,NBOUN ,UNKNO ,SOUND ,CONIN,
     &                     CINF ,ISIDO ,PRES1 ,CC    ,CC2   ,CINF2 )
              CALL BOUND(MXPOI ,MXBOU ,NPOIN ,NBOUN  ,ISIDO ,CINF ,UNKNO,
     &                   CINF2 ,NWALL ,WNOR  ,IWPOIN ,UNKN1 ,SOUND ,CONIN,
     &                   PRES1 ,CC    ,CC2   ,TINF   ,TINF2 ,RSIDO )

C *** CALCULATING THE TEMPERATURES

              DO IP = 1, NPOIN
                     TT1(IP) = GAMMA*( UNKNO(4,IP) - 0.5*( UNKNO(2,IP)**2 +
     &                         UNKNO(3,IP)**2 )/UNKNO(1,IP) )/UNKNO(1,IP)
                     PRES(IP) = PRES1(IP)
              ENDDO !IP

C *** ESTIMATION OF CONVERGENCE
C *** SHOW SUMMATION SQUARE OF ERROR IN FIRST TIME ITERATION

              IF(ITIME.EQ.NSTEP) THEN
                     DO IQ=1,4
                            SUMER(IQ) = 0.0D+00
                                   DO IR=1,NPOIN
                                          DIFF = UNKNO(IQ,IR) - UNKN1(IQ,IR)
                                          SUMER(IQ) = SUMER(IQ) + DIFF*DIFF
                                   ENDDO !IR
                            SUMER(IQ) = DSQRT(SUMER(IQ))
                     ENDDO !IQ
              ENDIF

C *** CHECK SUMMATION SQUARE OF ERROR IN EVERY TIME ITERATION

              DO IQ=1,4
                     SUMSQ(IQ) = 0.
                            DO IR=1,NPOIN
                                   DIFF = UNKNO(IQ,IR) - UNKN1(IQ,IR)
                                   SUMSQ(IQ) = SUMSQ(IQ) + DIFF*DIFF
                            ENDDO !IR
                     SUMSQ(IQ) = DSQRT(SUMSQ(IQ))
              ENDDO !IQ

C *** SHOW ERROR SUM SQUARE EVERY IWRITE

              KKK = MOD(ITIME,IWRITE)
              IF(KKK.EQ.0.OR.ITIME.EQ.ISTEP.OR.INTIME.EQ.1) THEN
                     WRITE(*,105)ITIME, (SUMSQ(IQ),IQ=1,4)
  105                FORMAT(4X,I6,5X,E12.6,3(2X,E12.6))
              ENDIF

C *** CHECK ERROR IN EVERY TIME ITERATION WITH FIRST TIME ITERATION

              IFINAL = 0
              DO IQ=1,4
                     IF(SUMSQ(IQ).LT.SUMER(IQ)*1.E-6) IFINAL = IFINAL + 1
              ENDDO !IQ

              IF(IFINAL.EQ.4) THEN
                     WRITE(*,110)ITIME, (SUMSQ(IQ),IQ=1,4)
  110                FORMAT(I8,4(2X,E12.6))
```

```
                    GOTO 2000
                ENDIF

C *** END OF LOOP TIME ITERATION

        ENDDO !ITIME

 2000 CONTINUE

      CALL GETDAT(TMPYEARE ,TMPMONTHE  ,TMPDAYE)
      CALL GETTIM(TMPHOURE ,TMPMINUTEE ,TMPSECONDE ,TMPHUNDE)
      WRITE(7,*)'END TIME:'
      WRITE (7, 115) TMPMONTHE, TMPDAYE, TMPYEARE
  115 FORMAT(I2, '/', I2.2, '/', I4.4)
      WRITE (7, 120) TMPHOURE,TMPMINUTEE,TMPSECONDE,TMPHUNDE
  120 FORMAT(I2, ':', I2.2, ':', I2.2, ':', I2.2, ' ',A)

C *** PRINT OVERALL TIME USED TO COMPUTE

      WRITE(*,121)
  121 FORMAT(/,'START TIME:')
      WRITE (*, 15) TMPMONTHS, TMPDAYS, TMPYEARS
      WRITE (*, 20) TMPHOURS,TMPMINUTES,TMPSECONDS,TMPHUNDS
C
      WRITE(*,125)
  125 FORMAT(/,'END TIME:')
      WRITE (*, 115) TMPMONTHE, TMPDAYE, TMPYEARE
      WRITE (*, 120) TMPHOURE,TMPMINUTEE,TMPSECONDE,TMPHUNDE

C *** TRANSFORM  CONSERVATION VARIABLES TO STANDARD VARIABLES

        DO IP = 1, NPOIN
                DO IA = 2, 4
                        UNKNO(IA,IP) = UNKNO(IA,IP)/UNKNO(1,IP)
                ENDDO !IA
      ENDDO !IP

C *** OUTPUT FOR RESTART FILE AND NASTRAN AND TECPLOT

      CALL OUTPUT(MXPOI, MXELE, MXBOU, NPOIN, NELEM, NBOUN, INTMA,
     &            COORD, UNKNO, ISIDO, CONIN, CINF , NTIME, ISTEP,
     &            TIMT , IWRITE,ILOTS, DTFIX, CSAFM, THETA, PRES,
     &            SOUND, NITER, CSMOO, NSMOO, CINF2, ABSV ,AMACH,
     &            NWALL, IWPOIN )

      STOP
      END
C*************************************************************************C
C       THIS SUBROUTINE READ MESH CONNECTION AND OTHER VALUES INPUT          C


C*************************************************************************C

      SUBROUTINE INPUT(MXPOI ,MXELE ,MXBOU ,NPOIN  ,NELEM ,NBOUN ,IOPT ,
     &                 INTMA ,COORD ,UNKNO ,ISIDO  ,CONIN ,CINF  ,NTIME,
     &                 ISTEP ,TIMT  ,ILOTS ,IWRITE ,DTFIX ,CSAFM ,THETA,
     &                 CC    ,TINF  ,SOUND ,TT1    ,PRES  ,PRES1 ,METHOD,
     &                 CSMOO ,NSMOO ,PINF  ,CC2    ,PINF2 ,TINF2 ,CINF2 )

      IMPLICIT  REAL*8(A-H,O-Z)

      INTEGER   INTMA(3,MXELE) ,ISIDO(4,MXBOU)

      REAL*8    CONIN(3)       ,CINF(5)       ,CINF2(5)       ,COORD(2,MXPOI)
      REAL*8    UNKNO(4,MXPOI) ,SOUND(MXPOI)  ,TT1(MXPOI)
      REAL*8    THETA(2)       ,PRES(MXPOI)   ,PRES1(MXPOI)

      REAL*8    TEXT(20)

C *** SPECIFIED THE RELAXATION PARAMETERS FOR VELOCITIES AND PRESSURE
C *** AND NUMBER OF SMOOTING

       IOPT    = 0
```

```
        THETA(1) = 0.5
        THETA(2) = 0.0
        CONIN(2) = 0.5
        CONIN(3) = 0.0
        ILOTS    = -1
        CSAFM    = 1.0

C ***   READING SPECIFIC HEAT RATIO (GAMMA)

      READ(5,1) TEXT
      READ(5,*) CONIN(1)
    1 FORMAT(20A4)

C *** READING NO.OF STEPS TO BE RUN,NO.OF STEPS AFTER WHICH TO WRITE RESIDUES
C *** FIXED TIME STEP VALUE

      READ(5,1) TEXT
      READ(5,*) NTIME ,IWRITE ,DTFIX

C *** READING FREE STREAM VALUES OF RHO, U1, U2, P AND MACH
C *** NOTE: PINF IS ANYWAY CALCULATED FROM OTHERS

      READ(5,1) TEXT
      READ(5,*) (CINF(I),I = 1,5)
      READ(5,1) TEXT
      READ(5,*) (CINF2(I),I = 1,5)

C *** READING THE COORDINATES OF THE NODAL POINTS

      READ(5,1) TEXT
      DO I = 1, NPOIN
               READ(5,*) IP, (COORD(J,IP),J = 1,2)
               IF(I.NE.IP) THEN
                       WRITE(*,5) I
    5                  FORMAT(/,'NODE NO.', I6,' IN DATA FILE IS MISSING')
                       STOP
               ENDIF
      ENDDO !I

C *** READING THE UNKNOWNS(RHO,U1,U2 AND E)

      READ(5,1) TEXT
      IF(IOPT.EQ.0) READ(5,*) RASSUM, UASSUM, VASSUM
      IF(IOPT.EQ.1) THEN
        DO I = 1, NPOIN
               READ(5,*) IP, (UNKNO(J,I),J = 1,4)
               IF(I.NE.IP)THEN
                       WRITE(*,10) I
   10                  FORMAT(/,'NODE NO.', I6,' IN DATA FILE IS MISSING')
                       STOP
               ENDIF
        ENDDO !I
      ENDIF

C *** READING ELEMENT CONNECTIVITY

      READ(5,1) TEXT
      DO I = 1, NELEM
               READ(5,*) IE, (INTMA(J,IE),J = 1,3)
               IF(I.NE.IE) THEN
                       WRITE(*,15) I
   15                  FORMAT(/,'ELEMENT NO.', I6,' IN DATA FILE IS MISSING')
                       STOP
               ENDIF
      ENDDO !I

C *** READING THE BOUNDARY SIDES INFORMATION

      READ(5,1) TEXT
      DO I = 1, NBOUN
               READ(5,*) (ISIDO(J,I),J = 1,4)
               IF(ISIDO(1,I).EQ.0)THEN
                       WRITE(*,20)ISIDO(3,I)
```

```
                STOP
            ENDIF
            IF(ISIDO(2,I).EQ.0)THEN
                WRITE(*,20)ISIDO(3,I)
                STOP
            ENDIF
            IF(ISIDO(3,I).EQ.0)THEN
                WRITE(*,20)ISIDO(3,I)
                STOP
            ENDIF
            IF(ISIDO(4,I).EQ.0)THEN
                WRITE(*,20)ISIDO(3,I)
                STOP
            ENDIF
   20       FORMAT(/,'ELEMENT NO.', I6,'ON BOUNDARY SIDE IN DATAFILE IS
     &MISSING')
      ENDDO !I

C *** CALCULATING FREE STREAM SPEED OF SOUND, PINF AND TINF

      CC   = DSQRT( CINF(2)**2 + CINF(3)**2 )/CINF(5)
      PINF = CC**2*CINF(1)/CONIN(1)
      TINF = CC**2/( CONIN(1)-1.0D+00 )

      CC2   = DSQRT( CINF2(2)**2 + CINF2(3)**2 )/CINF2(5)
      PINF2 = CC2**2*CINF2(1)/CONIN(1)
      TINF2 = CC2**2/( CONIN(1)-1.0D+00 )

      IF(IOPT.EQ.0) THEN
            DO IP = 1, NPOIN
                UNKNO(1,IP) = RASSUM
                UNKNO(2,IP) = UASSUM
                UNKNO(3,IP) = VASSUM
                UNKNO(4,IP) = TINF/CONIN(1) + 0.5*( UASSUM**2 + VASSUM**2)
                TT1(IP)     = TINF
                SOUND(IP)   = CC
                PRES(IP)    = (SOUND(IP)**2)*UNKNO(1,IP)/CONIN(1)
                PRES1(IP)   = PRES(IP)
            ENDDO !IP
            DO IB = 1, NBOUN
                DO  IBB = 1, 2
                    IP = ISIDO(IBB,IB)
                    IF(ISIDO(4,IB).EQ.3) THEN
                        UNKNO(1,IP) = CINF2(1)
                        UNKNO(2,IP) = CINF2(2)
                        UNKNO(3,IP) = CINF2(3)
                        UNKNO(4,IP) = TINF2/CONIN(1)+0.5*(CINF2(2)**2
     &                                +CINF2(3)**2)
                        TT1(IP) = TINF2
                        SOUND(IP) = CC2
                        PRES(IP)    = (SOUND(IP)**2)*UNKNO(1,IP)/CONIN(1)
                        PRES1(IP)   = PRES(IP)
                    ENDIF
                ENDDO !IBB
            ENDDO !IB
      ENDIF
C
      IF(IOPT.EQ.1) THEN
            DO IP = 1, NPOIN
                TT1(IP)   = CONIN(1)*( UNKNO(4,IP) - 0.5*(UNKNO(2,IP)**2
     &                      +UNKNO(3,IP)**2) )
                PRES(IP)  = ( CONIN(1)-1 )*UNKNO(1,IP)*TT1(IP)/CONIN(1)
                PRES1(IP) = PRES(IP)
            ENDDO !IP
      ENDIF
      CLOSE(5)
      END

C ********************************************************************C
C       SUBROUTINE PRELIMINARY , CALCULATE AREA , LUMP MASS              C
C ********************************************************************C

      SUBROUTINE PRELIM(MXPOI,MXELE,MXBOU,NPOIN,NELEM,NBOUN,INTMA,NCMAX,
```

```
     &                    MATCON,IFLAG,ILOTS,COORD ,GEOME,ALEN,DMMAT,
     &                     ISIDO ,RSIDO,IWPOIN ,WNOR  ,NWALL )

       IMPLICIT   REAL*8(A-H,O-Z)

       INTEGER    INTMA(3,MXELE), MATCON(MXPOI,20), NCMAX(MXPOI)
       INTEGER    IFLAG(MXPOI)  ,IHLEP2(MXPOI)
       INTEGER    ISIDO(4,MXBOU),IWPOIN(3,MXBOU)
       INTEGER    MXPOI, MXBOU, NPOIN, NBOUN,IB,IPOI0,IPOI1

       REAL*8     GEOME(7,MXELE),COORD(2,MXPOI),ALEN(MXPOI),DMMAT(MXPOI)
       REAL*8     X(3),Y(3),PNXI(3),PNET(3)
       REAL*8     RSIDO(4,MXBOU)
       REAL*8     WNOR(2,MXBOU)
       REAL*8     DX,DY,RL

C *** OBTAIN NODES ASSOCIATED WITH EACH NODE

       IF(ILOTS.NE.-1)THEN
         DO IP = 1, NPOIN
              CALL IFILLV(IFLAG, NPOIN, 0)
              NCMAX(IP) = 0
              DO IE = 1, NELEM
                      DO IK = 1, 3
                      IKP = INTMA(IK,IE)
                              IF(IKP.EQ.IP) THEN
                                      IK1 = IK + 1
                                      IF(IK1.GT.3) IK1 = IK1 - 3
                                      IKP1 = INTMA(IK1,IE)
                                      IK2 = IK1 + 1
                                      IF(IK2.GT.3) IK2 = IK2 - 3
                                      IKP2 = INTMA(IK2,IE)
                                      IF(IFLAG(IKP1).NE.1) THEN
                                              IFLAG(IKP1) = 1
                                              NCMAX(IP) = NCMAX(IP) + 1
                                              MATCON(IP,NCMAX(IP)) = IKP1
                                      ENDIF
                                      IF(IFLAG(IKP2).NE.1) THEN
                                              IFLAG(IKP2) = 1
                                              NCMAX(IP) = NCMAX(IP) + 1
                                              MATCON(IP,NCMAX(IP)) = IKP2
                                      ENDIF
                              ENDIF
                      ENDDO !IK
              ENDDO !IE
         ENDDO !IP
       ENDIF

C       OBTAIN [dNi/dX] = bi/2A  ,  [dNi/dY] = ci/2A

       DATA PNXI/-1.0D+00, 1.0D+00, 0.0D+00/
       DATA PNET/-1.0D+00, 0.0D+00, 1.0D+00/

       DO IELEM = 1,NELEM
              DO INODE = 1,3
                      IN = INTMA(INODE,IELEM)
                      X(INODE) = COORD(1,IN)
                      Y(INODE) = COORD(2,IN)
              ENDDO !INODE
              X21 = X(2)-X(1)
              X31 = X(3)-X(1)
              Y21 = Y(2)-Y(1)
              Y31 = Y(3)-Y(1)
              RJ  = X21*Y31-X31*Y21
              RJ1 = 1.0D+00/RJ
              XIX =  Y31*RJ1
              XIY = -X31*RJ1
              ETX = -Y21*RJ1
              ETY =  X21*RJ1
              DO IN = 1,2
                      RNXI = PNXI(IN)
                      RNET = PNET(IN)
                      GEOME(IN,IELEM) = XIX*RNXI + ETX*RNET
```

```
                GEOME(IN+3,IELEM) = XIY*RNXI + ETY*RNET
            ENDDO !IN
            GEOME(3,IELEM) = -( GEOME(1,IELEM) + GEOME(2,IELEM) )
            GEOME(6,IELEM) = -( GEOME(4,IELEM) + GEOME(5,IELEM) )
            GEOME(7,IELEM) = RJ
      ENDDO !IELEM

C     OBTINE MINIMUM THE CHARACTERISTIC HEIGHT OF EACH NODE

      DO IP = 1, NPOIN
            ALEN(IP) = 1.0D+00
      ENDDO !IP
      DO IE = 1, NELEM
            DO I = 1, 3
                  IP  = INTMA(I,IE)
                  ANX = GEOME(I,IE)
                  ANY = GEOME(I+3,IE)
                  HEIGHT = 1.0/DSQRT(ANX*ANX + ANY*ANY)
                  ALEN(IP) = MIN( ALEN(IP),HEIGHT )
            ENDDO !I
      ENDDO !IE

C     OBTAIN THE LUMPED MASS MATRIX [M]

      CALL RFILLV(DMMAT, NPOIN, 0.0D+00)
      DO IELEM = 1, NELEM
            RJ = GEOME(7,IELEM)
            RJ6 = RJ/6.0D+00
                  DO INODE = 1, 3
                        IN = INTMA(INODE,IELEM)
                        DMMAT(IN) = DMMAT(IN) + RJ6
                  ENDDO !INODE
      ENDDO !IELEM
      DO IP = 1, NPOIN
            DMMAT(IP) = 1.0D+00/DMMAT(IP)
      ENDDO !I
      END

C*************************************************************************C
C     OBTAIN DIRECTION COSINE OF BOUNDARY NORMAL WALL AND LENGHT OF WALL  C
C*************************************************************************C

      SUBROUTINE GETBOUN(MXPOI,MXBOU, NPOIN, NBOUN,ISIDO,IHELP, RSIDO,
     &                   IWPOIN,WNOR, NWALL, COORD )

      IMPLICIT   REAL*8(A-H,O-Z)

      INTEGER     ISIDO(4,MXBOU), IWPOIN(3,MXBOU), IHELP(MXPOI)

      REAL*8      RSIDO(3,MXBOU), WNOR(2,MXBOU) ,COORD(2,MXPOI)

      CALL IFILLV(IHELP, NPOIN, 0)
      CALL IFILLM(IWPOIN, 3, NBOUN, 0)
      CALL RFILLM(RSIDO, 3, NBOUN, 0.0D+00)

C *** FIND DIRECTION COSINE

      DO IB = 1, NBOUN
            IPOI0 = ISIDO(1,IB)
            IPOI1 = ISIDO(2,IB)
            DX = COORD(1,IPOI1) - COORD(1,IPOI0)
            DY = COORD(2,IPOI1) - COORD(2,IPOI0)
            RL = DSQRT(DX*DX+DY*DY)
            RSIDO(1,IB) = DY/RL
            RSIDO(2,IB) = -DX/RL
            RSIDO(3,IB) =  RL
      ENDDO !IB

      NWALL = 0
      DO IN = 1, 2
            DO I = 1, NBOUN
                  IF(ISIDO(4,I).EQ.2) THEN
                  NN = ISIDO(IN,I)
```

```
                        JJ = IHELP(NN)
                                IF(JJ.EQ.0) THEN
                                        NWALL = NWALL + 1
                                        IWPOIN(1,NWALL) = NN
                                        IWPOIN(2,NWALL) = I
                                        IHELP(NN) = NWALL
                                ELSE
                                        IWPOIN(3,JJ) = I
                                ENDIF
                        ENDIF
                ENDDO !I
        ENDDO !IN

C *** FIND NORMAL WALL VECTOR

        DO IW = 1, NWALL
                IB = IWPOIN(2,IW)
                IB2 = IWPOIN(3,IW)
                ANX1 = RSIDO(3,IB)*RSIDO(1,IB)
                ANY1 = RSIDO(3,IB)*RSIDO(2,IB)
                IF(IB2.NE.0) THEN
                        ANX1 = ANX1 + RSIDO(3,IB2)*RSIDO(1,IB2)
                        ANY1 = ANY1 + RSIDO(3,IB2)*RSIDO(2,IB2)
                        ACH = RSIDO(1,IB)*RSIDO(1,IB2) + RSIDO(2,IB)*RSIDO(2,IB2)
                        IF(ACH.LT.-0.2) THEN
                                ITRAIL = IW
                                WRITE(*,*) IWPOIN(1,IW),' IS TRAILING EDGE'
                                WNOR(1,IW) = 0.0D+00
                                WNOR(2,IW) = 0.0D+00
                                GO TO 300
                        ENDIF
                ENDIF
                ANOR = DSQRT(ANX1*ANX1 + ANY1*ANY1)
                ANX1 = ANX1/ANOR
                ANY1 = ANY1/ANOR
                WNOR(1,IW) = ANX1
                WNOR(2,IW) = ANY1
  300 CONTINUE
        ENDDO !IW
        END

C*************************************************************************C
C       SUBROUTINE FILL VALUE IV IN TO VECTOR {LV} FOR INTEGER VECTOR     C
C*************************************************************************C

        SUBROUTINE IFILLV(LV,NL,IV)
        IMPLICIT   REAL*8(A-H,O-Z)
        INTEGER   LV(NL)
        DO J = 1,NL
                LV(J) = IV
        ENDDO !J
        END

C*************************************************************************C
C       SUBROUTINE FILL VALUE K INTO MATRIX [MA] FOR INTEGER MATRIX       C
C*************************************************************************C

         SUBROUTINE IFILLM(MA,NA,NELEM,K)
         IMPLICIT   REAL*8(A-H,O-Z)
        INTEGER   MA(NA,NELEM)
        DO J = 1,NELEM
                DO I = 1,NA
                        MA(I,J) = K
                ENDDO !I
        ENDDO !J
        END

C*************************************************************************C
C       SUBROUTINE FILL VALUE C IN TO VECTOR {A} FOR REAL VECTOR
        C
C*************************************************************************C

        SUBROUTINE RFILLV(A, NA, C)
```

```
      IMPLICIT   REAL*8(A-H,O-Z)
      REAL*8    A(NA)
      DO I = 1, NA
              A(I) = C
      ENDDO !I
      END

C*****************************************************************************C
C       SUBROUTINE FILL VALUE CM INTO MATRIX [A] FOR REAL MATRIX              C
C*****************************************************************************C

      SUBROUTINE RFILLM(A, NA, MA, CM)
      IMPLICIT   REAL*8(A-H,O-Z)
      REAL*8     A(NA,MA)
      DO I = 1, NA
              DO J = 1, MA
                      A(I,J) = CM
              ENDDO !J
      ENDDO !I
      END

C*****************************************************************************C
C       SUBROUTINE STEP 1 CALCULATES INTERMEDATE MOMENTUM IN X,Y DIRECTION    C
C*****************************************************************************C

      SUBROUTINE STEP1(MXPOI, MXELE, MXBOU , NPOIN, NELEM, NBOUN ,
     &                 INTMA, GEOME, DMMAT , UNKNO, RSIDO, ISIDO ,
     &                 RHS0 , RHS1 , RHS2  , PSWE , IHELP, DELTP ,
     &                 DELTE, SOUND, CONIN , NCMAX, ITIME, INTIME,
     &                 CINF , THETA, FXSEC , FYSEC, PRES , PRES1 ,
     &                 CC   , CSAFM, DTFIX , ILOTS, TT1  , ALEN  ,
     &                 NITER, MATCON,CC2   ,CINF2)

      IMPLICIT   REAL*8(A-H,O-Z)

      INTEGER     INTMA(3,MXELE), ISIDO(4,MXBOU)  , NUMBER(MXPOI)
      INTEGER     MATCON(MXPOI,20), NCMAX(MXPOI)  , IHELP(MXPOI)

      REAL*8      ALEN(MXPOI)   , GEOME(7,MXELE)  , DMMAT(MXPOI)
      REAL*8      UNKNO(4,MXPOI), RSIDO(3,MXBOU)  , RHS0(4,MXPOI)
      REAL*8      RHS1(4,MXPOI) , RHS2(4,MXPOI)   , PSWE(MXPOI)
      REAL*8      DELTP(MXPOI)  , DELTE(MXELE)    , SOUND(MXPOI)
      REAL*8      CONIN(3)      , CINF(5)         , THETA(2)
      REAL*8      FXSEC(4,MXELE), FYSEC(4,MXELE)  , PRES(MXPOI)
      REAL*8      PRES1(MXPOI)  , TT1(MXPOI)      , CINF2(5)

      C00 = 0.0D+00
      CSAFE = CSAFM

C *** CALCULATES PRESSURE FROM VELOCITIES AND TOTAL ENERGY

C     CALL GETPRES(MXPOI, MXBOU, NPOIN, NBOUN, UNKNO, SOUND, CONIN,
C    &             CINF, ISIDO, PRES , CC ,CC2 ,CINF2 )

C *** CALCULATES THE CRITICAL TIME STEP FOR ALL THE NODES

       IF(ILOTS.NE.-1)THEN
              CALL ALOTIM(MXPOI, MXELE, NELEM, NPOIN, ILOTS, INTMA, CSAFM,
     &                    DTFIX, UNKNO, DELTP, DELTE, SOUND, GEOME, PRES ,
     &                    CONIN, ALEN , NCMAX, MATCON)
        ELSE
              DTFIX2 = DTFIX
              CALL RFILLV(DELTP, NPOIN, DTFIX)
              CALL RFILLV(DELTE, NELEM, DTFIX2)
       ENDIF

C *** CALCULATES THE  CONVECTIVE COMPONENT FOR (*) VELOCITIES
C *** AND STABILIZING TERMS

      CALL RFILLM(RHS2, 4, NPOIN, C00)

      CALL ADVECT( MXPOI, MXELE, NPOIN, NELEM, INTMA, GEOME, UNKNO,
     &             DELTE, RHS2,  FXSEC, FYSEC, THETA, PRES )
```

```
      CALL ADVSIDE(MXPOI, MXELE, MXBOU, NPOIN, NELEM, NBOUN, RSIDO,
     &             UNKNO, RHS2, FXSEC,  FYSEC, ISIDO )

C *** ADD ADVECTION COMPONENT AND MULTIPLY BY INVERSED MASS
C *** UPDATE THE SOLUTION.

      DO IP = 1, NPOIN
               UNKNO(2,IP) = UNKNO(2,IP) + DELTP(IP)*DMMAT(IP)*RHS2(2,IP)
               UNKNO(3,IP) = UNKNO(3,IP) + DELTP(IP)*DMMAT(IP)*RHS2(3,IP)
      ENDDO !IP
      END

C*************************************************************************C
C       SUBROUTINE CALCULATES PRESSURE FROM VELOCITIES AND ENERGY          C
C*************************************************************************C

      SUBROUTINE GETPRES(MXPOI, MXBOU, NPOIN, NBOUN, UNKNO, SOUND,
     &                   CONIN, CINF,  ISIDO, PRE  , CC,CC2,CINF2 )

      IMPLICIT   REAL*8(A-H,O-Z)

      INTEGER    ISIDO(4,MXBOU)

      REAL*8     PRE(MXPOI),  UNKNO(4,MXPOI), SOUND(MXPOI)
      REAL*8     CONIN(3), CINF(5) ,CINF2(5)

      GAM = CONIN(1)
      GAM1 = GAM - 1.0D+00
      PINF = (CC**2)*CINF(1)/GAM
      PINF2= (CC2**2)*CINF2(1)/GAM
      DO IP = 1, NPOIN
               PRE(IP) = GAM1*( UNKNO(4,IP) - 0.5*( UNKNO(2,IP)**2
     &                   +UNKNO(3,IP)**2 )/UNKNO(1,IP) )
      ENDDO !IP
      DO IB = 1, NBOUN
               DO IBB = 1, 2
                        IP = ISIDO(IBB,IB)
                        IF(ISIDO(4,IB).EQ.1) THEN
                        PRE(IP) = PINF
                        ENDIF
               ENDDO !IBB
      ENDDO !IB
C
      DO IB = 1, NBOUN
               DO IBB = 1, 2
                        IP = ISIDO(IBB,IB)
                        IF(ISIDO(4,IB).EQ.3) THEN
                        PRE(IP) = PINF2
                        ENDIF
               ENDDO !IBB
      ENDDO !IB
      END

C*************************************************************************C
C       SUBROUTINE CALCULATES THE CRITICAL TIME STEP FOR ALL NODES         C
C*************************************************************************C

      SUBROUTINE ALOTIM(MXPOI, MXELE, NELEM, NPOIN, ILOTS, INTMA, CSAFM,
     &                  DTFIX, UNKNO, DELTP, DELTE, SOUND, GEOME, PRES ,
     &                  CONIN, ALEN , NCMAX, MATCON)

      IMPLICIT   REAL*8(A-H,O-Z)

      INTEGER    MPOI
      PARAMETER (MPOI=200000)

      INTEGER    INTMA(3,MXELE), NCMAX(MXPOI)   , MATCON(MXPOI,20)

      REAL*8     VMOD(MPOI)     , UNKNO(4,MXPOI), DELTP(MXPOI)
      REAL*8     DELTE(MXELE)   , SOUND(MXPOI)  , GEOME(7,MXELE)
      REAL*8     PRES(MXPOI)    , CONIN(3)      , ALEN(MXPOI)
      REAL*8     VNORM(MPOI)    , PNEW(MPOI)    , RHONEW(MPOI)
```

```
      IF(ILOTS.LE.-1) THEN
               DTFIX2 = DTFIX*2.0
               CALL RFILLV(DELTP, NPOIN, DTFIX)
               CALL RFILLV(DELTE, NELEM, DTFIX2)
               RETURN
      ENDIF
      DO IP = 1, NPOIN
               UVEL = UNKNO(2,IP)/UNKNO(1,IP)
               VVEL = UNKNO(3,IP)/UNKNO(1,IP)
               VMOD(IP) = DSQRT( UVEL*UVEL + VVEL*VVEL )
      ENDDO !IP
      DO IP = 1, NPOIN
               DELTP(IP)  = 1.0E+06
               VNORM(IP)  = 0.0D+00
               PNEW(IP)   = 0.0D+00
               RHONEW(IP) = 0.0D+00
               DO IK = 1, NCMAX(IP)
                        IKK = MATCON(IP,IK)
                        VNORM(IP) = VNORM(IP) + VMOD(IKK)
                        PNEW(IP) = PNEW(IP) + PRES(IKK)
                        RHONEW(IP) = RHONEW(IP) + UNKNO(1,IKK)
               ENDDO !IK
               VNORM(IP) = VNORM(IP)/NCMAX(IP)
               PNEW(IP) = PNEW(IP)/NCMAX(IP)
               RHONEW(IP) = RHONEW(IP)/NCMAX(IP)
               SOUND(IP) = DSQRT(CONIN(1)*PNEW(IP)/RHONEW(IP))
      ENDDO !IP
      DO IE = 1, NELEM
               IP1 = INTMA(1,IE)
               IP2 = INTMA(2,IE)
               IP3 = INTMA(3,IE)
               CMAX = MAX( SOUND(IP1),SOUND(IP2),SOUND(IP3) )
               VMAX = MAX( VNORM(IP1),VNORM(IP2),VNORM(IP3) )
               VMAX = VMAX + CMAX
               ALOTI = (ALEN(IP1)/VMAX)
               DELTP(IP1) = MIN(ALOTI,DELTP(IP1))
               ALOTI = (ALEN(IP2)/VMAX)
               DELTP(IP2) = MIN(ALOTI,DELTP(IP2))
               ALOTI = (ALEN(IP3)/VMAX)
               DELTP(IP3) = MIN(ALOTI,DELTP(IP3))
      ENDDO !IE
      DO IE = 1, NELEM
               IP1 = INTMA(1,IE)
               IP2 = INTMA(2,IE)
               IP3 = INTMA(3,IE)
               DELTE(IE) = (DELTP(IP1)+DELTP(IP2)+DELTP(IP3))/3.0
      ENDDO !IE
      DO IP = 1, NPOIN
               DELTP(IP) = CSAFM*DELTP(IP)
      ENDDO !IP
      IF(ILOTS.EQ.0) GO TO 700
      RETURN
  700 DTSML = 1.0D06
      DO IP = 1, NPOIN
               DTSML = MIN(DTSML,DELTP(IP))
      ENDDO !IP
      DTBIG = 1.0D06
      DO IE = 1, NELEM
                DTBIG = MIN(DTBIG,DELTE(IE))
      ENDDO !IE
      CALL RFILLV( DELTP, NPOIN, DTSML)
      CALL RFILLV( DELTE, NELEM, DTBIG)
      END

C*************************************************************************C
C       SUBROUTINE CALCULATES ARTIFICAIL DIFFUSION BASED ON SECOND GRADIENT    C
C*************************************************************************C

      SUBROUTINE ARTVIS(MXPOI, MXELE, NPOIN, NELEM, INTMA, PRES, CONIN,
     &                  GEOME, DMMAT, UNKNO, RHS1 , SOUND, ALEN )

      IMPLICIT REAL*8(A-H,O-Z)
```

```
      INTEGER     MPOI,MELE
      PARAMETER   (MPOI=200000, MELE=400000)

      INTEGER   INTMA(3,MXELE)

      REAL*8    PRES(MXPOI)   , CONIN(3)       , GEOME(7,MXELE)
      REAL*8    DMMAT(MXPOI) , UNKNO(4,MXPOI), RHS1(4,MXPOI)
      REAL*8    SOUND(MXPOI) , ALEN(MXPOI)    , DSPDX(MELE)
      REAL*8    DSPDY(MELE)   , DSPDN(MELE )  , DPDX(MPOI)
      REAL*8    DPDY(MPOI)    , UNKEL(4,3)     , GEO(6)
      REAL*8    SN(3)         , UVEL(3)        , VVEL(3)
      REAL*8    PRX(3)    , AREA3(MPOI)

      CALL RFILLM(RHS1, 4, NPOIN, 0.0D+0)
      CALL RFILLV(DPDX , MXPOI, 0.0D+00)
      CALL RFILLV(DPDY , MXPOI, 0.0D+00)
      CALL RFILLV(DSPDX, MXELE, 0.0D+00)
      CALL RFILLV(DSPDY, MXELE, 0.0D+00)
      CALL RFILLV(DSPDN, MXELE, 0.0D+00)
      CALL RFILLV(AREA3, MXPOI, 0.0D+00)

      DO IE = 1, NELEM
               AR3 = GEOME(7,IE)/6.0D+00
               DO I = 1, 3
                        IP = INTMA(I,IE)
                        DO J = 1, 3
                                 JP = INTMA(J,IE)
                                 DPDX(IP) = DPDX(IP) + AR3*GEOME(J,IE)*PRES(JP)
                                 DPDY(IP) = DPDY(IP) + AR3*GEOME(J+3,IE)*PRES(JP)
                        ENDDO !J
               ENDDO !I
      ENDDO !IE
      DO IP = 1, NPOIN
               DPDX(IP) = DMMAT(IP)*DPDX(IP)
               DPDY(IP) = DMMAT(IP)*DPDY(IP)
      ENDDO !IP
      DO IE = 1, NELEM
               IP1 = INTMA(1,IE)
               IP2 = INTMA(2,IE)
               IP3 = INTMA(3,IE)
               DSPDX(IE) = GEOME(1,IE)*DPDX(IP1) + GEOME(2,IE)*DPDX(IP2)
     &              +GEOME(3,IE)*DPDX(IP3)
               DSPDY(IE) = GEOME(4,IE)*DPDY(IP1) + GEOME(5,IE)*DPDY(IP2)
     &             +GEOME(6,IE)*DPDY(IP3)
               DSPDN(IE) = DSQRT(DSPDX(IE)**2 + DSPDY(IE)**2)
      ENDDO !IE
      DO IE = 1, NELEM
               AR = GEOME(7,IE)/2.0D+00
               ALSUM = 0.0D+00
               DO I = 1, 3
                        IP = INTMA(I,IE)
                        UNKEL(1,I) = UNKNO(1,IP)
                        UNKEL(2,I) = UNKNO(2,IP)
                        UNKEL(3,I) = UNKNO(3,IP)
                        UNKEL(4,I) = UNKNO(4,IP) + PRES(IP)
                        UVEL(I) = UNKEL(2,I)/UNKEL(1,I)
                        VVEL(I) = UNKEL(3,I)/UNKEL(1,I)
                        SN(I) = SOUND(IP)
                        PRX(I) = PRES(IP)
                        GEO(I) = GEOME(I,IE)
                        GEO(I+3) = GEOME(I+3,IE)
                        ALSUM = ALSUM + ALEN(IP)
               ENDDO !I
          VN1 = DSQRT( UVEL(1)**2 + VVEL(1)**2 )
          VN2 = DSQRT( UVEL(2)**2 + VVEL(2)**2 )
          VN3 = DSQRT( UVEL(3)**2 + VVEL(3)**2 )
          VELN = MAX(VN1, VN2, VN3)
          CMAX = MAX(SN(1), SN(2), SN(3))
          VELMAX = VELN + CMAX
          PBAR = ( PRX(1) + PRX(2) + PRX(3) )/3.0D+0
          ALSUM = ALSUM/3.0D+00
          DRDX=UNKEL(1,1)*GEO(1)+UNKEL(1,2)*GEO(2)+UNKEL(1,3)*GEO(3)
          DRDY=UNKEL(1,1)*GEO(4)+UNKEL(1,2)*GEO(5)+UNKEL(1,3)*GEO(6)
```

```
      DUDX=UNKEL(2,1)*GEO(1)+UNKEL(2,2)*GEO(2)+UNKEL(2,3)*GEO(3)
      DUDY=UNKEL(2,1)*GEO(4)+UNKEL(2,2)*GEO(5)+UNKEL(2,3)*GEO(6)
      DVDX=UNKEL(3,1)*GEO(1)+UNKEL(3,2)*GEO(2)+UNKEL(3,3)*GEO(3)
      DVDY=UNKEL(3,1)*GEO(4)+UNKEL(3,2)*GEO(5)+UNKEL(3,3)*GEO(6)
      DHDX=UNKEL(4,1)*GEO(1)+UNKEL(4,2)*GEO(2)+UNKEL(4,3)*GEO(3)
      DHDY=UNKEL(4,1)*GEO(4)+UNKEL(4,2)*GEO(5)+UNKEL(4,3)*GEO(6)
      CONST = AR*CONIN(2)*(ALSUM**3)*VELMAX*DSPDN(IE)/PBAR
      DO I = 1, 3
          IP = INTMA(I,IE)
          RHS1(1,IP) = RHS1(1,IP) - CONST*
     &                 (DRDX*GEO(I) + DRDY*GEO(I+3) )
          RHS1(2,IP) = RHS1(2,IP) - CONST*
     &                 (DUDX*GEO(I) + DUDY*GEO(I+3) )
          RHS1(3,IP) = RHS1(3,IP) - CONST*
     &                 (DVDX*GEO(I) + DVDY*GEO(I+3) )
          RHS1(4,IP) = RHS1(4,IP) - CONST*
     &                 (DHDX*GEO(I) + DHDY*GEO(I+3) )
           ENDDO !I
       ENDDO !IE
       END

C*************************************************************************C
C       SUBROUTINE CALCULATES THE CONVECTIVE COMPONENT FOR (*) VELOCITIES      c
C*************************************************************************C

       SUBROUTINE ADVECT(MXPOI, MXELE, NPOIN, NELEM, INTMA, GEOME, UNKNO,
      &                  DELTE, RHS2,  FXSEC, FYSEC, THETA, PRES )

       IMPLICIT REAL*8(A-H,O-Z)

       INTEGER   INTMA(3,MXELE)

       REAL*8    GEOME(7,MXELE), UNKNO(4,MXPOI), DELTE(MXELE)
       REAL*8    RHS2(4,MXPOI) , FXSEC(4,MXELE), FYSEC(4,MXELE)
       REAL*8    THETA(2)      , UNKEL(3,3)    , FXEL(3,3)
       REAL*8    FYEL(3,3)     , VELOC(2,3)    , GEO(6)
       REAL*8    FXME(3)       , FYME(3)       , DVFEL(3)
       REAL*8    FXMSE(3)      , FYMSE(3)      , AM1(3,3)
       REAL*8    AM2(3,3)      , NODE(3)       , PRES(NPOIN)

       DO IE = 1, NELEM
                AREA = 0.5D+00*GEOME(7,IE)
                DPDX = 0.0D+00
                DPDY = 0.0D+00
                DO IN = 1, 3
                       IP = INTMA(IN,IE)
                       NODE(IN) = IP
                       GEO(IN) = GEOME(IN,IE)
                       GEO(IN+3) = GEOME(IN+3,IE)
                       DPDX = DPDX + GEO(IN)*PRES(IP)
                       DPDY = DPDY + GEO(IN+3)*PRES(IP)
                       DO IA = 1,3
                              UNKEL(IA,IN) = UNKNO(IA,IP)
                       ENDDO !IA
                ENDDO !IN
                VELOC(1,1) = UNKEL(2,1)/UNKEL(1,1)
                VELOC(1,2) = UNKEL(2,2)/UNKEL(1,2)
                VELOC(1,3) = UNKEL(2,3)/UNKEL(1,3)
                VELOC(2,1) = UNKEL(3,1)/UNKEL(1,1)
                VELOC(2,2) = UNKEL(3,2)/UNKEL(1,2)
                VELOC(2,3) = UNKEL(3,3)/UNKEL(1,3)
                DO IUNKN = 2, 3
                       FXME(IUNKN) = 0.0D+00
                       FYME(IUNKN) = 0.0D+00
                ENDDO !IUNKN
                DO INODE=1, 3
                       FXEL(2,INODE) = VELOC(1,INODE)*UNKEL(2,INODE)
                       FXEL(3,INODE) = VELOC(1,INODE)*UNKEL(3,INODE)
                       FYEL(2,INODE) = VELOC(2,INODE)*UNKEL(2,INODE)
                       FYEL(3,INODE) = VELOC(2,INODE)*UNKEL(3,INODE)
                       DO IUNKN = 2,3
                              FXME(IUNKN) = FXME(IUNKN)+FXEL(IUNKN,INODE)
                              FYME(IUNKN) = FYME(IUNKN)+FYEL(IUNKN,INODE)
```

```
                ENDDO !IUNKN
            ENDDO !INODE
            DO IUNKN = 2, 3
                DVFEL(IUNKN) = 0.0D+00
                DO INODE = 1, 3
                    ANX = GEO(INODE)
                    ANY = GEO(INODE+3)
                    DVFEL(IUNKN) = DVFEL(IUNKN) + ANX*FXEL(IUNKN,INODE)
     &                                +ANY*FYEL(IUNKN,INODE)
                ENDDO !INODE
            ENDDO !IUNKN

C ***  ADD STABILIZING TERMS

            U1 = ( UNKEL(1,1) + UNKEL(1,2) + UNKEL(1,3) )/3.0
            U2 = ( UNKEL(2,1) + UNKEL(2,2) + UNKEL(2,3) )/3.0
            U3 = ( UNKEL(3,1) + UNKEL(3,2) + UNKEL(3,3) )/3.0
            U21 = U2/U1
            U31 = U3/U1
            AM1(2,2) = U21
            AM1(3,3) = U21
            AM2(2,2) = U31
            AM2(3,3) = U31
            DO IUNKN = 2, 3
                FXMSE(IUNKN) = 0.0D+00
                FYMSE(IUNKN) = 0.0D+00
                FXMSE(IUNKN) = FXMSE(IUNKN)+AM1(IUNKN,IUNKN)*DVFEL(IUNKN)
                FYMSE(IUNKN) = FYMSE(IUNKN)+AM2(IUNKN,IUNKN)*DVFEL(IUNKN)
            ENDDO !IUNKN
            FXMSE(2) = FXMSE(2) + AM1(2,2)*( 1.0-THETA(2) )*DPDX
            FYMSE(2) = FYMSE(2) + AM2(2,2)*( 1.0-THETA(2) )*DPDX
            FXMSE(3) = FXMSE(3) + AM1(3,3)*( 1.0-THETA(2) )*DPDY
            FYMSE(3) = FYMSE(3) + AM2(3,3)*( 1.0-THETA(2) )*DPDY
            DO IUNKN=2, 3
                FXSEC(IUNKN,IE) = FXMSE(IUNKN)*DELTE(IE)*1.5
                FYSEC(IUNKN,IE) = FYMSE(IUNKN)*DELTE(IE)*1.5
            ENDDO !IUNKN
            AREA = AREA/3.0D+00
            DO IN = 1,3
                IP = NODE(IN)
                ANX = GEO(IN)*AREA
                ANY = GEO(IN+3)*AREA
                DO IA = 2, 3
                    RHS2(IA,IP) = RHS2(IA,IP)+ANX*FXME(IA)+ANY*FYME(IA)
     &                            -ANX*FXSEC(IA,IE) -ANY*FYSEC(IA,IE)
                ENDDO !IA
            ENDDO !IN
        ENDDO !IE
        END

C*************************************************************************C
C       CALCULATES SIDE CONTRIBUTION TO CONVECTIVE TERM FOR (*)VELOCITIES     C
C*************************************************************************C

        SUBROUTINE ADVSIDE( MXPOI, MXELE, MXBOU, NPOIN, NELEM, NBOUN,
     &                      RSIDO, UNKNO, RHS2,  FXSEC, FYSEC, ISIDO )

        IMPLICIT REAL*8(A-H,O-Z)

        INTEGER      ISIDO(4,MXBOU),  NODE(2)

        REAL*8       RSIDO(3,MXBOU), UNKNO(4,MXPOI), RHS2(4,MXPOI)
        REAL*8       FXSEC(4,MXELE), FYSEC(4,MXELE), UNKEL(3,2)
        REAL*8       VELOC(2,2)    , FN(3,2)       , FXSI(3,2)

        C16 = 1.0D+00/6.0D+00
        DO IS = 1, NBOUN
            INDEX = ISIDO(4,IS)
            ALENG = RSIDO(3,IS)*C16
            ANX = RSIDO(1,IS)*ALENG
            ANY = RSIDO(2,IS)*ALENG
            IE = ISIDO(3,IS)
            DO IN = 1, 2
```

```
                IP = ISIDO(IN,IS)
                NODE(IN) = IP
                DO IA = 1, 3
                        UNKEL(IA,IN) = UNKNO(IA,IP)
                ENDDO !IA
            ENDDO !IN
            VELOC(1,1) = UNKEL(2,1)/UNKEL(1,1)
            VELOC(2,1) = UNKEL(3,1)/UNKEL(1,1)
            VELOC(1,2) = UNKEL(2,2)/UNKEL(1,2)
            VELOC(2,2) = UNKEL(3,2)/UNKEL(1,2)
            FXSI(2,1) = UNKEL(2,1)*( ANX*VELOC(1,1) + ANY*VELOC(2,1) )
            FXSI(2,2) = UNKEL(2,2)*( ANX*VELOC(1,2) + ANY*VELOC(2,2) )
            FXSI(3,1) = UNKEL(3,1)*( ANX*VELOC(1,1) + ANY*VELOC(2,1) )
            FXSI(3,2) = UNKEL(3,2)*( ANX*VELOC(1,2) + ANY*VELOC(2,2) )
            FN(2,1) = 2.0*FXSI(2,1) + FXSI(2,2)
            FN(2,2) = 2.0*FXSI(2,2) + FXSI(2,1)
            FN(3,1) = 2.0*FXSI(3,1) + FXSI(3,2)
            FN(3,2) = 2.0*FXSI(3,2) + FXSI(3,1)
            DO IN = 1, 2
                    IP = NODE(IN)
                    DO IA = 2, 3
                            RHS2(IA,IP) = RHS2(IA,IP)-FN(IA,IN)
     &                                    +ANX*FXSEC(IA,IE)
     &                                    +ANY*FYSEC(IA,IE)
                    ENDDO !IA
            ENDDO !IN
      ENDDO !IS
      END

C*****************************************************************************C
C     SUBROUTINE STEP 2 CALCULATES DENSITY(EXPLICIT SOLVER)                     C
C*****************************************************************************C

      SUBROUTINE STEP2 (MXPOI, MXELE, MXBOU, NPOIN, NBOUN, NELEM,
     &                  INTMA, ISIDO, RSIDO, GEOME, DMMAT, UNKNO,
     &                  UNKN1, RHS1 , RHS2 , IELSI, PRES , DELTP,
     &                  DELTE, THETA, NITER )

      IMPLICIT REAL*8(A-H,O-Z)

      INTEGER   INTMA(3,MXELE), ISIDO(4,MXBOU), IELSI(2,MXELE)

      REAL*8    RSIDO(3,MXBOU), GEOME(7,MXELE), DMMAT(MXPOI)
      REAL*8    UNKNO(4,MXPOI), UNKN1(4,MXPOI), RHS2(4,MXPOI)
      REAL*8    PRES(MXPOI)   , DELTP(MXPOI)  , DELTE(MXELE)
      REAL*8    RHS1(4,MXPOI) , THETA(2)      , RHSP(3)
      REAL*8    ELRHS(3)

C *** CLEAR MATRIX [RHS2] BY FILL ZERO IN THIS MATRIX

      CALL RFILLM(RHS2, 4, NPOIN, 0.0D+00)

      DO IELEM = 1, NELEM

C *** EVALUATES THE VELOCITY TERMS ON RHS FOR DENSITY EQUATION

            CALL GETRHS(MXPOI, MXELE, MXBOU, NPOIN, NELEM, NBOUN,
     &                  INTMA, ISIDO, IELSI, RSIDO, GEOME, UNKNO,
     &                  UNKN1, THETA, ELRHS, IELEM )

C *** EVALUATES THE PRESSURE LAPLACIAN ON RHS OF DENSITY EQUATION

            CALL PRHP(MXPOI,MXELE,MXBOU,NPOIN, NELEM, NBOUN, ISIDO,
     &                INTMA, IELSI, GEOME, RSIDO, PRES , DELTE, RHSP,
     &                IELEM )
C
            DO I = 1, 3
                    IP = INTMA(I,IELEM)
                    RHS2(1,IP) = RHS2(1,IP) + RHSP(I)*THETA(1)
     &                           +ELRHS(I)
            ENDDO !I
      ENDDO !IELEM
```

```
C *** MULTIPLY BY INVERSED MASS AND TIME STEP FOR ALL NODES
C *** UPDATE THE SOLUTION

      DO IP = 1, NPOIN
               UNKNO(1,IP) = UNKNO(1,IP) + DELTP(IP)*DMMAT(IP)*RHS2(1,IP)
      ENDDO !IP
      END

C*************************************************************************C
C     SUBROUTINE EVALUATES THE VELOCITY TERMS ON RHS FOR DENSITY EQUATION     C
C*************************************************************************C

      SUBROUTINE GETRHS(MXPOI, MXELE, MXBOU, NPOIN, NELEM, NBOUN,
     &                  INTMA, ISIDO, IELSI, RSIDO, GEOME, UNKNO,
     &                  UNKN1, THETA, ELRHS, IELEM )

      IMPLICIT REAL*8(A-H,O-Z)

      INTEGER    INTMA(3,MXELE), ISIDO(4,MXBOU), IELSI(2,MXELE)

      REAL*8     RSIDO(3,MXBOU), GEOME(7,MXELE), UNKNO(4,MXPOI)
      REAL*8     UNKN1(4,MXPOI), THETA(2)      , ELRHS(3)
      REAL*8     ELUKN(3,3)    , ELUKN1(3,3)

C *** CALCULATES VELOCITES FIRST TWO TERMS IN STEP 2 EQUATION
C *** EQ 10 IN SHOCK CAPTURING PAPER

      DO INODE = 1, 3
               IP = INTMA(INODE,IELEM)
               DO IU = 2, 3
                      ELUKN(IU,INODE) = (1.0 - THETA(1) )*UNKN1(IU,IP)
     &                                  +THETA(1)*UNKNO(IU,IP)
                      ELUKN1(IU,INODE) = UNKN1(IU,IP)
               ENDDO !IU
      ENDDO !INODE
      VELO1 = 0.0D+00
      VELO2 = 0.0D+00
      DO JNODE = 1, 3
               VELO1 = VELO1 + ELUKN(2,JNODE)
               VELO2 = VELO2 + ELUKN(3,JNODE)
      ENDDO !JNODE
C
      DO INODE = 1, 3
               ELRHS(INODE) = 0.0D+00
               JNODE = INODE + 3
               ELRHS(INODE) = ELRHS(INODE) + GEOME(INODE,IELEM)*VELO1
     &                        +GEOME(JNODE,IELEM)*VELO2
               ELRHS(INODE) = ELRHS(INODE)*GEOME(7,IELEM)/6.0
      ENDDO !INODE
C
      DO ISI=1,2
               IS=IELSI(ISI,IELEM)
               IF(IS.NE.0) THEN
                      INDEX=ISIDO(4,IS)
                      ALENG=RSIDO(3,IS)/6.0D+00
                      ANX=RSIDO(1,IS)*ALENG
                      ANY=RSIDO(2,IS)*ALENG
                      IP=ISIDO(1,IS)
                      DO INODE=1, 3
                             IF(IP.EQ.INTMA(INODE,IELEM)) THEN
                                    ID1 = INODE
                                    GO TO 60
                             ENDIF
                      ENDDO !INODE
                      PRINT *, 'ERROR IN SIDE CONTRIBUTION'
                      PRINT *, IELEM
                      STOP
   60                 CONTINUE
                      ID2 = ID1+1
                      IF(ID2.GT.3) ID2 = ID2 - 3
                      IP1 = INTMA(ID1,IELEM)
                      UP = ELUKN1(2,ID1)
                      VP = ELUKN1(3,ID1)
```

```
                    ANORV1 = UP*ANX + VP*ANY
                    IP2 = INTMA(ID2,IELEM)
                    UP = ELUKN1(2,ID2)
                    VP = ELUKN1(3,ID2)
                    ANORV2 = UP*ANX + VP*ANY
                    ELRHS(ID1) = ELRHS(ID1) - 2.0*ANORV1 - ANORV2
                    ELRHS(ID2) = ELRHS(ID2) - ANORV1 - 2.0*ANORV2
               ENDIF
      ENDDO !ISI
      END

C*************************************************************************C
C     SUBROUTINE EVALUATES THE PRESSURE LAPLACIAN ON RHS OF DENSITY EQUATION  C
C*************************************************************************C

      SUBROUTINE PRHP(MXPOI,MXELE,MXBOU,NPOIN, NELEM, NBOUN, ISIDO,
     &                INTMA, IELSI, GEOME, RSIDO, PRES , DELTE, RHSP,
     &                IE )

      IMPLICIT   REAL*8(A-H,O-Z)

      INTEGER    ISIDO(4,MXBOU), INTMA(3,MXELE), IELSI(2,MXELE)

      REAL*8     GEOME(7,MXELE), RSIDO(3,MXBOU), PRES(MXPOI)
      REAL*8     DELTE(MXELE)  , RHSP(3)       , P(3), B(2)

      AR = GEOME(7,IE)/2.0D+00
      DO LOK = 1, 3
               IM = INTMA(LOK,IE)
               P(LOK) = PRES(IM)
               RHSP(LOK) = 0.0D+00
      ENDDO !LOK
      B(1) = 0.0D+00
      B(2) = 0.0D+00
      DO LOK = 1, 3
               IN = INTMA(LOK,IE)
               B(1) = B(1) + GEOME(LOK,IE)*P(LOK)
               LOK1 = LOK+3
               B(2) = B(2) + GEOME(LOK1,IE)*P(LOK)
      ENDDO !LOK
      B(1) = B(1)*DELTE(IE)
      B(2) = B(2)*DELTE(IE)
      DO LOK = 1, 3
               LOK1 = LOK + 3
               IN = INTMA(LOK,IE)
               RHSP(LOK) = RHSP(LOK) - AR*( GEOME(LOK,IE)*B(1)
     &                    +GEOME(LOK1,IE)*B(2) )
      ENDDO !LOK
      END

C*************************************************************************C
C     THIS SUBROUTINE VELOCITY CORRECTION STEP                            C
C*************************************************************************C

      SUBROUTINE STEP3(MXPOI, MXELE, MXBOU, NPOIN, NELEM, NBOUN, INTMA,
     &                 ISIDO, RSIDO, GEOME, DMMAT, UNKNO, UNKN1, IELSI,
     &                 RHS2 , DELTP, DELTE, THETA, PRES , PRES1 )

      IMPLICIT    REAL*8(A-H,O-Z)

      INTEGER     INTMA(3,MXELE), ISIDO(4,MXBOU), IELSI(2,MXELE)

      REAL*8      GEOME(7,MXELE), RSIDO(3,MXBOU), UNKNO(4,MXPOI)
      REAL*8      UNKN1(4,MXPOI), DMMAT(MXPOI)  , RHS2(4,MXPOI)
      REAL*8      DELTP(MXPOI)  , THETA(2)      , PRES(MXPOI)
      REAL*8      PRES1(MXPOI)  , DELTE(MXELE)

      DO IPOIN = 1, NPOIN
               DO IUNKN = 2, 3
                        RHS2(IUNKN,IPOIN) = 0.0D+00
               ENDDO !IUNKN
      ENDDO !IPOIN
```

```
C *** CALCULATE PRESSURE TERM2 IN THIRD STEP

      CALL CORRECT(MXPOI, MXELE, MXBOU, NPOIN, NELEM, NBOUN, ISIDO,
     &             INTMA, IELSI, UNKNO, RSIDO, GEOME, THETA, PRES ,
     &             PRES1, RHS2 )

C *** MULTIPLY BY INVERSE MASS MATRIX AND TIME STEP AND UPDATE THE SOLUTION

      DO IP = 1, NPOIN
               DO IU = 2, 3
                      DT = DMMAT(IP)
                      RHS2(IU,IP)  = RHS2(IU,IP)*DT*DELTP(IP)
                      UNKNO(IU,IP) = UNKNO(IU,IP) + RHS2(IU,IP)
               ENDDO !IU
      ENDDO !IP
      END

C*************************************************************************C
C       SUBROUTINE MOMENTUM CORRECTON STEP                                C
C*************************************************************************C

      SUBROUTINE CORRECT(MXPOI,MXELE,MXBOU,NPOIN, NELEM, NBOUN, ISIDO,
     &                   INTMA, IELSI,UNKNO,RSIDO,GEOME, THETA, PRES ,
     &                   PRES1, RHS2)

      IMPLICIT   REAL*8(A-H,O-Z)

      INTEGER    INTMA(3,MXELE), ISIDO(4,MXBOU), IELSI(2,MXELE)

      REAL*8     RSIDO(3,MXBOU), GEOME(7,MXELE), THETA(2)
      REAL*8     PRES(MXPOI)   , PRES1(MXPOI)  , RHS2(4,MXPOI)
      REAL*8     UNKNO(4,MXPOI), ELRHS(3,3)    , ELPRS(3)

      DO IELEM = 1, NELEM
               AREA = GEOME(7,IELEM)/2.0D+00
               DO INODE = 1, 3
                      IP = INTMA(INODE,IELEM)
                      ELPRS(INODE) = (1.0-THETA(2))*PRES(IP)
     &                              +THETA(2)*PRES1(IP)
                      ELRHS(1,INODE) = 0.0D+00
                      ELRHS(2,INODE) = 0.0D+00
                      ELRHS(3,INODE) = 0.0D+00
               ENDDO !INODE
               DO IUNKN = 2, 3
                      IDIME = IUNKN - 1
                      DO INODE = 1, 3
                             KNODE = INODE + (IDIME-1)*3
                             DO JNODE = 1, 3
                                    ELRHS(IUNKN,INODE)=ELRHS(IUNKN,INODE)
     &                                   +GEOME(KNODE,IELEM)*ELPRS(JNODE)
                             ENDDO !JNODE
                      ENDDO !INODE
               ENDDO !IUNKN
               DO IUNKN = 2, 3
                      DO INODE = 1, 3
                             ELRHS(IUNKN,INODE) = ELRHS(IUNKN,INODE)*AREA/3.0
                      ENDDO !INODE
               ENDDO !IUNKN
               DO ISI = 1, 2
                      IS = IELSI(ISI,IELEM)
                      IF(IS.NE.0) THEN
                             INDEX = ISIDO(4,IS)
                             ALENG = RSIDO(3,IS)/6.0
                             ANX = RSIDO(1,IS)*ALENG
                             ANY = RSIDO(2,IS)*ALENG
                             IP = ISIDO(1,IS)
                             DO INODE = 1, 3
                                    IF(IP.EQ.INTMA(INODE,IELEM)) THEN
                                           ID1 = INODE
                                           GO TO 8
                                    ENDIF
                             ENDDO !INODE
                             PRINT *, 'ERROR IN SIDE CONTRIBUTION'
```

```
                              STOP
    8                         CONTINUE
                              ID2 = ID1 + 1
                              IF(ID2.GT.3) ID2 = ID2 - 3
                              ELRHS(2,ID1) = ELRHS(2,ID1)
     &                          -(ELPRS(ID1)*2.0 + ELPRS(ID2) )*ANX
                              ELRHS(2,ID2) = ELRHS(2,ID2)
     &                                    -(ELPRS(ID2)*2.0 + ELPRS(ID1) )*ANX
                              ELRHS(3,ID1) = ELRHS(3,ID1)
     &                                    -(ELPRS(ID1)*2.0 + ELPRS(ID2) )*ANY
                              ELRHS(3,ID2) = ELRHS(3,ID2)
     &                                    -(ELPRS(ID2)*2.0 + ELPRS(ID1) )*ANY
                       ENDIF
                ENDDO !ISI
                DO INODE = 1, 3
                       DO IU = 2, 3
                              IP = INTMA(INODE,IELEM)
                              RHS2(IU,IP) = RHS2(IU,IP) + ELRHS(IU,INODE)
                       ENDDO !IU
           ENDDO !INODE
       ENDDO !IELEM
       END

C*************************************************************************C
C       SUBROUTINE STEP4 CALCULATES ENERGY EQUATION                       C
C*************************************************************************C

       SUBROUTINE STEP4(MXPOI, MXELE, MXBOU,NPOIN,NELEM,NBOUN,INTMA,
      &                 GEOME,UNKN1,RHS2, DELTE,FXSEC,FYSEC,PRES,ISIDO,
      &                 RSIDO,RHS0,TT1,DELTP,UNKNO,INTIME,NITER,DMMAT,
      &                 RHS1)

       IMPLICIT REAL*8(A-H,O-Z)

       INTEGER INTMA(3,MXELE), ISIDO(4,MXBOU) ,NODE3(3)

       REAL*8  GEOME(7,MXELE), RHS2(4,MXPOI), DELTE(MXELE), DELTP(MXPOI)
       REAL*8  UNKN1(4,MXPOI), UNKNO(4,MXPOI),FXSEC(4,MXELE)
       REAL*8  FYSEC(4,MXELE), RHS1(4,MXPOI)
       REAL*8  PRES(MXPOI), RSIDO(3,MXBOU), RHS0(4,MXPOI)
       REAL*8  DMMAT(MXPOI), TT1(MXPOI)

C *** CALCULATES CONVECTIVE AND STABILIZING TERM  IN ENERGY EQUATION

       CALL ENERCON(MXPOI, MXELE,NPOIN, NELEM, INTMA, GEOME, UNKNO,
      &             RHS2, DELTE, FXSEC, FYSEC, PRES )
       CALL ENERSID(MXPOI, MXELE, MXBOU, NPOIN, NELEM, NBOUN, ISIDO,
      &             RSIDO, UNKNO, FXSEC, FYSEC, RHS2 , PRES )

C *** MULTIPLY BY INVERSE MATRIX
C *** UPDATE SOLUTION

       DO IP = 1,NPOIN
                UNKNO(4,IP) = UNKNO(4,IP) + DELTP(IP)*DMMAT(IP)*RHS2(4,IP)
       ENDDO !IP
       END

C*************************************************************************C
C     SUBROUTINE COMPUTES THE CONVECTIVE COMPONENT OF ENERGY EQUATION     C
C*************************************************************************C

       SUBROUTINE ENERCON( MXPOI, MXELE, NPOIN, NELEM, INTMA,GEOME,UNKNO,
      &                    RHS2,  DELTE, FXSEC, FYSEC, PRES1 )

       IMPLICIT REAL*8(A-H,O-Z)

       INTEGER   INTMA(3,MXELE),NODE(3)

       REAL*8    GEOME(7,MXELE), UNKNO(4,MXPOI), RHS2(4,MXPOI)
       REAL*8    DELTE(MXELE)  , FXSEC(4,MXELE), FYSEC(4,MXELE)
       REAL*8    UNKEL(4,3)    , FXEL(3)       , FYEL(3)
       REAL*8    VELOC(2,3)    , ENTA0(3)      , GEO(6)
       REAL*8    PRX(3)        , PRES1(MXPOI)
```

```
      DO IE = 1, NELEM
               AREA = 0.5*GEOME(7,IE)
               DO IN = 1, 3
                       IP       = INTMA(IN,IE)
                       NODE(IN) = IP
                       GEO(IN)  = GEOME(IN,IE)
                       GEO(IN+3)= GEOME(IN+3,IE)
                       VELOC(1,IN) = UNKNO(2,IP)/UNKNO(1,IP)
                       VELOC(2,IN) = UNKNO(3,IP)/UNKNO(1,IP)
                       DO IA = 1, 4
                              UNKEL(IA,IN) = UNKNO(IA,IP)
                       ENDDO !IA
                       PRX(IN) = PRES1(IP)
                       ENTA0(IN) = ( UNKEL(4,IN) + PRX(IN) )
               ENDDO !IN
               FXME = 0.0D+00
               FYME = 0.0D+00
               DO INODE=1, 3
                       FXEL(INODE) = VELOC(1,INODE)*ENTA0(INODE)
                       FYEL(INODE) = VELOC(2,INODE)*ENTA0(INODE)
                       FXME = FXME + FXEL(INODE)
                       FYME = FYME + FYEL(INODE)
               ENDDO !INODE
               DVFEL = 0.0D+00
               DO INODE = 1, 3
                       ANX = GEO(INODE)
                       ANY = GEO(INODE+3)
                       DVFEL = DVFEL + ANX*FXEL(INODE) + ANY*FYEL(INODE)
               ENDDO !INODE
               RHO  = (UNKEL(1,1)+UNKEL(1,2)+UNKEL(1,3))/3.0
               UVEL = (UNKEL(2,1)+UNKEL(2,2)+UNKEL(2,3))/3.0
               VVEL = (UNKEL(3,1)+UNKEL(3,2)+UNKEL(3,3))/3.0
               U1 = UVEL/RHO
               U2 = VVEL/RHO

C *** SECOND ORDER CONTRIBUTION

               FXMSE = 0.0D+00
               FYMSE = 0.0D+00
               FXMSE = FXMSE + U1*DVFEL
               FYMSE = FYMSE + U2*DVFEL

C *** STORE FOR SIDE CONTRIBUTION

               FXSEC(4,IE) = FXMSE*DELTE(IE)*1.5
               FYSEC(4,IE) = FYMSE*DELTE(IE)*1.5

C *** NOW DISTRIBUTE

               AREA = AREA/3.0D+00
               DO IN = 1, 3
                       IP = NODE(IN)
                       ANX = GEO(IN)*AREA
                       ANY = GEO(IN+3)*AREA
                       RHS2(4,IP) = RHS2(4,IP) + ANX*FXME + ANY*FYME
     &                     -ANX*FXSEC(4,IE)-ANY*FYSEC(4,IE)
               ENDDO !IN
      ENDDO !IE
      END

C*************************************************************************C
C     SUBUBROUTINE COMPUTES THE BOUNDARY SIDES CONTRIBUTION FOR ENERGY     C
C*************************************************************************C

      SUBROUTINE ENERSID( MXPOI, MXELE, MXBOU,NPOIN, NELEM,NBOUN,ISIDO,
     &                    RSIDO, UNKNO, FXSEC, FYSEC, RHS2 , PRES1 )

      IMPLICIT REAL*8(A-H,O-Z)

      INTEGER   ISIDO(4,MXBOU), NODE(2)

      REAL*8    RSIDO(3,MXBOU), UNKNO(4,MXPOI), FXSEC(4,MXELE)
```

```
      REAL*8    FYSEC(4,MXELE), RHS2(4,MXPOI) , UNKEL(4,2)
      REAL*8    FXSI(4,2)     , FN(4,2)       , ENTA0(2)
      REAL*8    VELOC(2,2)    , PRES1(MXPOI)

      C16 = 1.0D+00/6.0D+00
      DO IS = 1, NBOUN
               INDEX = ISIDO(4,IS)
               ALENG = RSIDO(3,IS)*C16
               ANX = RSIDO(1,IS)*ALENG
               ANY = RSIDO(2,IS)*ALENG
               IE = ISIDO(3,IS)
               DO IN = 1, 2
                      IP = ISIDO(IN,IS)
                      NODE(IN) = IP
                      DO IA = 1,4
                             UNKEL(IA,IN) = UNKNO(IA,IP)
                      ENDDO !IA
                      ENTA0(IN) = ( UNKEL(4,IN) + PRES1(IP) )
               ENDDO !IN
               VELOC(1,1) = UNKEL(2,1)/UNKEL(1,1)
               VELOC(1,2) = UNKEL(2,2)/UNKEL(1,2)
               VELOC(2,1) = UNKEL(3,1)/UNKEL(1,1)
               VELOC(2,2) = UNKEL(3,2)/UNKEL(1,2)
               DO IN = 1, 2
                      QQ = ANX*VELOC(1,IN) + ANY*VELOC(2,IN)
                      FXSI(4,IN) = ENTA0(IN)*QQ
               ENDDO !IN
               FN(4,1) = 2.0*FXSI(4,1) + FXSI(4,2)
               FN(4,2) = 2.0*FXSI(4,2) + FXSI(4,1)
               DO IN = 1, 2
                      IP = NODE(IN)
                      RHS2(4,IP) = RHS2(4,IP) - FN(4,IN)
     &                           +ANX*FXSEC(4,IE)+ ANY*FYSEC(4,IE)
               ENDDO !IN
      ENDDO !IS
      END

C ************************************************************************ C
C       SUBROUTINE APPLIES SUPERSONIC BOUNDARY CONDITION                   C
C ************************************************************************ C

      SUBROUTINE BOUND (MXPOI ,MXBOU ,NPOIN ,NBOUN  ,ISIDO ,CINF ,UNKNO ,
     &                  CINF2 ,NWALL ,WNOR  ,IWPOIN ,UNKN1 ,SOUND ,CONIN ,
     &                   PRES1 ,CC    ,CC2   ,TINF   ,TINF2 ,RSIDO )

      IMPLICIT REAL*8(A-H,O-Z)

      INTEGER     ISIDO(4,MXBOU),IWPOIN(3,MXBOU)

      REAL*8      UNKNO(4,MXPOI),UNKN1(4,MXPOI),CINF(5),CINF2(5)
      REAL*8      WNOR(2,MXBOU) ,RSIDO(3,MXBOU),CONIN(3),SOUND(MXPOI)
      REAL*8      PRES1(MXPOI)

C *** SUPERSONIC INFLOW BOUNDARY CONDITION

      PINF = (CC**2)*CINF(1)/CONIN(1)
      PINF2= (CC2**2)*CINF2(1)/CONIN(1)

       DO IB = 1, NBOUN
              DO IN = 1, 2
                     IP = ISIDO(IN,IB)
                     IF(ISIDO(4,IB).EQ.1) THEN
                            UNKNO(1,IP) = CINF(1)
                            UNKNO(2,IP) = UNKNO(1,IP)*CINF(2)
                            UNKNO(3,IP) = UNKNO(1,IP)*CINF(3)
                     ENDIF
              ENDDO !IN
      ENDDO !IB
       DO IB = 1, NBOUN
              DO IN = 1, 2
                     IP = ISIDO(IN,IB)
                     IF(ISIDO(4,IB).EQ.3) THEN
                            UNKNO(1,IP) = CINF2(1)
```

```
                              UNKNO(2,IP) = UNKNO(1,IP)*CINF2(2)
                              UNKNO(3,IP) = UNKNO(1,IP)*CINF2(3)
                        ENDIF
                ENDDO !IN
        ENDDO !IS

C *** INVISCID WALL BOUNDARY CONDITION

        DO IW = 1, NWALL
                IP = IWPOIN(1,IW)
                ANX = WNOR(1,IW)
                ANY = WNOR(2,IW)
                FN = UNKN1(2,IP)*ANX + UNKN1(3,IP)*ANY
                RN = UNKNO(2,IP)*ANX + UNKNO(3,IP)*ANY
                RN = RN - FN
                FN = 0.1D+00*FN
                UNKNO(2,IP) = UNKNO(2,IP) - (RN+FN)*ANX
                UNKNO(3,IP) = UNKNO(3,IP) - (RN+FN)*ANY
        ENDDO !IW

C *** SYMMETRY BOUNDARY CONDITION

        DO IB = 1, NBOUN
                IF(ISIDO(4,IB).EQ.4) THEN
                        RANX = RSIDO(1,IB)
                        RANY = RSIDO(2,IB)
                        DO IN = 1, 2
                                IP = ISIDO(IN,IB)
                                US = -UNKNO(2,IP)*RANY + UNKNO(3,IP)*RANX
                                UNKNO(2,IP) = - US*RANY
                                UNKNO(3,IP) =  US*RANX
                        ENDDO !IN
                ENDIF
        ENDDO !IS

C *** SUPERSONIC ENERGY INFLOW BOUNDARY CONDITION

        DO IB = 1, NBOUN
                DO  IBB = 1, 2
                        IP = ISIDO(IBB,IB)
                        IF(ISIDO(4,IB).EQ.1) THEN
                                ENER1 = TINF/CONIN(1)
                                ENER2 = 0.5*( CINF(2)**2 + CINF(3)**2 )
                                ENER = ENER1 + ENER2
                                UNKNO(4,IP) = CINF(1)*ENER
                        ENDIF
                        IF(ISIDO(4,IB).EQ.3) THEN
                                ENER1 = TINF2/CONIN(1)
                                ENER2 = 0.5*( CINF2(2)**2 + CINF2(3)**2 )
                                ENER = ENER1 + ENER2
                                UNKNO(4,IP) = CINF2(1)*ENER
                        ENDIF
                ENDDO !IBB
        ENDDO !IB
          END

C*************************************************************************C
C     WRITE THE OUTPUT AFTER PRESCRIBED NUMBER OF ITERATIONS              C
C*************************************************************************C

      SUBROUTINE OUTPUT(MXPOI, MXELE, MXBOU, NPOIN, NELEM, NBOUN, INTMA,
     &                  COORD, UNKNO, ISIDO, CONIN, CINF , NTIME, ISTEP,
     &                  TIMT , IWRITE,ILOTS, DTFIX, CSAFM, THETA, PRES,
     &                  SOUND, NITER, CSMOO, NSMOO, CINF2, ABSV ,AMACH,
     &                  NWALL, IWPOIN)

      IMPLICIT REAL*8(A-H,O-Z)

      INTEGER       ISIDO(4,MXBOU), INTMA(3,MXELE),IWPOIN(3,MXBOU)

      REAL*8        COORD(2,MXPOI), UNKNO(4,MXPOI), CONIN(3)
      REAL*8        CINF(5)       , THETA(2)      , SOUND(MXPOI)
      REAL*8        PRES(MXPOI)   , HH(8)         , CINF2(5)
```

```
      REAL*8         AMACH(MXPOI)   , ABSV(MXPOI)

C *** WRITE THE SOLUTIONS OF THE PROBLEM

      WRITE(9,1)NPOIN
    1 FORMAT('  NODAL VALUES SOLUTIONS [',I6,']:')
      WRITE(9,2)
    2 FORMAT(/,'   NODE                RHO                  U
     &      V                   E        ')
        DO IP = 1,NPOIN
              WRITE(9,3)IP ,UNKNO(1,IP),UNKNO(2,IP),UNKNO(3,IP),UNKNO(4,IP)
    3         FORMAT(X,I6,4X,4(3X,E16.8))
      ENDDO !IP


C=============================================================================C
C       !!!!!!!!!!!!!!!!!!!!!!!!!!GOOD LUCK!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  C
C=============================================================================C
```

ประวัติผู้เขียนวิทยานิพนธ์

นายปริญญา บุญมาเลิศ เกิดเมื่อวันที่ 15 เดือนสิงหาคม พุทธศักราช 2522 จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิตจากภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ เมื่อปีการศึกษา 2543 เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2544