$$\frac{\partial k}{\partial t} + \nabla(ku) = \nabla[(\nu+\frac{\nu_t}{\sigma_k})\nabla k] + P_k - \beta^* k\omega$$

I convection　II diffusion　III production　IV source

```
C###########################################################################
c                                                                         #
c                     SUBROUTINE PROGRAM                                  #
C                   VERSION 1.0 (16/02/2011)                              #
C                   AUTHORIZED BY ZHANG JINGXIN                           #
C                                 SHANGHAI JIAO TONG UNIVERSITY           #
C                                 SHANGHAI, CHINA                         #
c-------------------------------------------------------------------------#
c       computes the advective,Coriolis,horizontal dispersive terms in the #
c       momentum equation of u                                            #
c                                                                         #
c###########################################################################
      Subroutine ADVTK(S1,F1,GRADX,GRADY,GRADZ,CDES)

      Include './Include/OCERM_INF'

      Common/VISKW/VISSOURCE(IJM,KB),TKEMARK(IJM,KB)

    Parameter (TINT = 0.05, DESTYPE = 1, CW = 0.15)

    Parameter(BETASTAR=0.09,FKAPA=0.41)
    Parameter(SIGMAK1=0.85, SIGMAO1=0.5, BETA1=0.075, A1=0.31,
     &           GAMA1=BETA1/BETASTAR-SIGMAO1*FKAPA**2./Sqrt(BETASTAR))
    Parameter(SIGMAK2=1.0, SIGMAO2=0.856, BETA2=0.0828,
     &           GAMA2=BETA2/BETASTAR-SIGMAO2*FKAPA**2./Sqrt(BETASTAR))

      Parameter(I_TURBMOD = 1, CFK=1.0)

    Dimension HQ(IJE,KB),VQ(IJM,KB), S1(IJM,KB)
    Dimension GRADX(IJM,KB),GRADY(IJM,KB),GRADZ(IJM,KB)
    Dimension TEMP(KBM)
        Dimension F1(IJM,KB),VISCOE(IJM,KB),VAR_T(IJM,-1:KB+1)
c=========================================================================c
c                    initialiing arrays                                   c
c=========================================================================c
!$OMP PARALLEL DEFAULT(SHARED) PRIVATE(I,K)
    Do K = 1, KB
!$OMP DO
        Do I = 1, IJM
           VISF(I,K) = 0.0
         VISCOE(I,K) = F1(I,K) * SIGMAK1 + (1. - F1(I,K)) * SIGMAK2
```

$$\phi = F_1\phi_1 + (1-F_1)\phi_2$$

```
            VISSOURCE(I,K) = 0.0
            TKEMARK(I,K) = 1.0
        Enddo
!$OMP END DO NOWAIT
!$OMP DO
        Do I = 1, IJE
           HQ(I,K) = 0.0
        Enddo
!$OMP END DO NOWAIT
!$OMP DO
        Do I = 1, IJM
           VQ(I,K) = 0.0
        Enddo
!$OMP END DO NOWAIT
     Enddo
!$OMP DO
       Do I = 1, IJM
        Do K = 1, KBM
           VAR_T(I,K) = TKE(I,K)
        Enddo
        VAR_T(I,0) = 2. * TKE(I,1) - TKE(I,2)
        VAR_T(I,-1) = VAR_T(I,0)
        VAR_T(I,KB) = 0.0
        VAR_T(I,KB+1) = 0.0
     Enddo
!$OMP END DO NOWAIT
!$OMP END PARALLEL
       If(IWENOSCHEME .NE. 0) Then
          Do K = 1, KBM
             Do I = -1, NUM_GHOST, -1
                QGHOST(I,K) = TKE(INDEX_GHOST(I),K)
             Enddo
          Enddo
     Endif
C============================================================================c
c             TVD schemes for the calculation of convective fluxes          c
c============================================================================c
       Call TVDSCHEMEH(HQ,TKE,GRADX,GRADY,IH_TVD)        HQ . k
       Call TVDSCHEMEV(VQ,TKE,VAR_T,GRADZ,IV_TVD)         Σφf
c============================================================================c
C                  Advection Descrization by 2nd Order TVD                   c
```

```
c==============================================================================c
!$OMP PARALLEL DEFAULT(SHARED)
!$OMP&          PRIVATE(I,J,K,XX,FV1,FV2,FV3,FT2,D1,D2,HMAX,HWN,M,N,
!$OMP&          SLOWERPART1,SLOWERPART2,SLOWER,DTIDLE,STIDLE,R,DIVXVIS,
!$OMP&          DIVYVIS,DIVZVIS,AAMF,FLUX1,FLUX2,FD,RDT,FT,FL,FE2,
!$OMP&          ALF1,FE1,FE,FB,FDT,FDITDLE,G,FW,ID,IS,ZDES,D2MAX,TTTT,
!$OMP&          S11,S12,S13,S21,S22,S23,S31,S32,S33,UP,YP,YSTAR,TKE_WALL)
      If (ADVECT.EQ.'NON-LINEAR') Then
c------------------------------------------------------------------------------c
c                     horizontal advective terms                               c
c------------------------------------------------------------------------------c
       Do K = 1, KBM
!$OMP DO
```

$I: \sum F_i \phi_i$

$F_i = \rho u \Delta A_i$

```
          Do I = 1, IJM
             If(CCM(I) .EQ. 1.0) Then
                Do J = 1, CELL_POLYGEN(I)
                   If(CFM(CELL_SIDE(I,J,1)) .EQ. 1.0) Then
                      VISF(I,K) = VISF(I,K) + DZ(K) *
     &                   HQ(CELL_SIDE(I,J,1),K) * CELL_CUV(I,J,6) *
     &                   (UN(CELL_SIDE(I,J,1),K) * CELL_CUV(I,J,7) +
     &                    VN(CELL_SIDE(I,J,1),K) * CELL_CUV(I,J,8))
                   Endif
                Enddo
             Endif
          Enddo
!$OMP END DO NOWAIT
       Enddo
!$OMP BARRIER
c------------------------------------------------------------------------------c
c                     vertical advective terms                                 c
c------------------------------------------------------------------------------c
       Do K = 1, KBM
!$OMP DO
```

$-\Delta A [\omega_k \phi_i - \omega_{k+1} \phi_i]$

```
          Do I = 1, IJM
             If(CCM(I) .EQ. 1.0) Then
                VISF(I,K) = -VISF(I,K) - AREA(I) *
     &                  (VQ(I,K) * W(I,K) - VQ(I,K+1) * W(I,K+1))
             Endif
          Enddo
!$OMP END DO NOWAIT
       Enddo
```

```
!$OMP BARRIER
    Endif
c=============================================================================c
C                    Horizontal Diffusion Descrization by CS                  c
c=============================================================================c
    Do K = 1, KBM
!$OMP DO
       Do I = 1, IJM
          If(CCM(I) .EQ. 1.0) Then
            Do J = 1, CELL_POLYGEN(I)
               FLUX1 = 0.0
               FLUX2 = 0.0
               If(CFM(CELL_SIDE(I,J,1)) .EQ. 1.0) Then
          Γ       AAMF = UMOL + (AAM(I,K) + AAM(CELL_SIDE(I,J,2),K)) / 2. *
     &                        (VISCOE(I,K) + VISCOE(CELL_SIDE(I,J,2),K))/2.
          Di        FLUX1 = (DISCOE(I,J,1) - DISCOE(I,J,8))* AAMF *
     &                        (TKE(CELL_SIDE(I,J,2),K) - TKE(I,K))
                    FLUX2 = (DISCOE(I,J,7) - DISCOE(I,J,2)) * AAMF*
     &        SD-cross         (TKEV(CELL_SIDE(I,J,4),K) -
     &                          TKEV(CELL_SIDE(I,J,3),K))
                    VISF(I,K) = VISF(I,K) +
     &                              (FLUX1 + FLUX2) * DZ(K)
                Endif
                If(CFM(CELL_SIDE(I,J,1)) .EQ. 0.0 .OR.
     &              CFM(CELL_SIDE(I,J,1)) .EQ. -1.0) Then
                   If(ISLIP .EQ. 0) Then
                      AAMF = UMOL + AAM(I,K) * VISCOE(I,K)
                     UP = Sqrt(UR(I,K) ** 2. +
     *                            VR(I,K) ** 2. + WR(I,K) ** 2.)
                      YP = D2D(I)
                      YSTAR =
     *                  Dmax1(Sqrt(0.3)*Sqrt(TKE(I,K))*YP/1.E-6,15.0)

                      TKE_WALL = 0.41*UP*Sqrt(0.3)*Sqrt(TKE(I,K))/
     *                             Dlog(9.81*YSTAR) / 0.3

                     FLUX1 = (DISCOE(I,J,1) - DISCOE(I,J,8)) * AAMF*
     &                           ( TKE_WALL - TKE(I,K))

C                    FLUX1 = -(DISCOE(I,J,1) - DISCOE(I,J,8)) * AAMF*
C     &                          TKE(I,K)
```

II. $\sum D_i(\phi_c - \phi_0) + S_{D\text{-cross}}$

$D_i = \frac{\Gamma}{\Delta\xi}\frac{n\cdot n}{n\cdot e_\xi}\Delta A_i$

$S_{D\text{-cross}} = -\Gamma\frac{e_\xi - e_\eta}{n\cdot e_\xi}\Delta A_i\frac{\phi_b-\phi_a}{\Delta\eta}$

```fortran
                      VISF(I,K) = VISF(I,K) + FLUX1 * DZ(K)
                    Endif
                 Endif
                   If(CFM(CELL_SIDE(I,J,1)) .EQ. -2.0) Then
                     AAMF = UMOL + AAM(I,K) * VISCOE(I,K)
                    FLUX1 = -(DISCOE(I,J,1) - DISCOE(I,J,8)) * AAMF*
     &                          TKE(I,K)
                     VISF(I,K) = VISF(I,K) + FLUX1 * DZ(K)
                 Endif
             Enddo
           Endif
        Enddo
!$OMP END DO NOWAIT
      Enddo
!$OMP BARRIER
C==============================================================================C
c                        source and sink terms                                 c
C==============================================================================C
!$OMP DO
      Do I = 1, IJM
         If(CCM(I) .EQ. 1.0) Then
          D2MAX = 0.0
          Do J = 1, CELL_POLYGEN(I)
             If(CELL_CUV(I,J,6) .GT. D2MAX) D2MAX = CELL_CUV(I,J,6)
          Enddo
          Do K = 1, KBM
             D1 = Sqrt(TKE(I,K)) / BETASTAR / (TDISS(I,K)+1.E-10 )

ccc          D1 = Dmax1(D1, 1.E-6)
CCC                D1 = Dmin1(D1,D2D(I))
             D2 = CDES * Max(D2MAX,DC(I)*DZ(K))
             ZDES = 0.0
c            D2 = CDES *
c     &           Dmin1(Dmax1(CW*D1,CW*HMAX,HWN),HMAX)   !LES length-scaleIDDES
             If(DES .NE. 'NEGLECT   ') Then
                If(DES .EQ. 'SSTDES    ') Goto 1
                If(DES .EQ. 'SSTDDES   ') Goto 2
                If(DES .EQ. 'SSTIDDES  ') Goto 3
1               Continue
c-----            SSTDES
ccc             DTIDLE = Dmin1(D1,D2)
```

```fortran
                DTIDLE = Max(D1/D2,1.0)
C-----          ZDEX
CC                If(DTIDLE .EQ. D2) Then
CC                   FV1 = 1.0
CC                   FV2 = 0.0
CC                   DTIDLE = CDES*(AREA(I) * DC(I) * DZ(K))**(1./3.)
C    &                      * SLOWER
CC                   ZDES = 1.0
CC                Endif
CC              STIDLE = VORTICITY(I,K) +
CC     &                      VIS(I,K) / VK ** 2. / DTIDLE ** 2. * FV2
CC            STIDLE = Dmax1(STIDLE, 1.E-6)
CC            R = (VIS(I,K) + UMOL) / VK ** 2. / DTIDLE ** 2. / STIDLE
c             R = (KM(I,K) - UMOL) / VK ** 2. / DTIDLE ** 2. /
c     &            Dmax1(TKE(I,K), 1.E-10)

C             R = VIS(I,K) / VK ** 2. / D1 ** 2. / TKE(I,K)
CC            R = Dmin1(R,10.)
              Goto 100
2             Continue
c-----          DDES by Spalart et al.
                R = VIS(I,K) / VK ** 2. / D1 ** 2. /
     &              Max(1.E-10,TKE(I,K))
              R = Min(R,10.)
              FD = 1. - Tanh((8. * R) ** 3.)
              DTIDLE = D1 - FD * Max(0.0, D1 - D2)
                STIDLE = TKE(I,K) +
     &                  VIS(I,K) / VK ** 2. / DTIDLE ** 2. * FV2
              STIDLE = Max(STIDLE, 1.E-10)
              Goto 100
3               Continue
c-----          IDDES by Spalart et al.
                RDT = VIS(I,K) / VK ** 2. / D1 ** 2. /
     &            Max(1.E-10,TKE(I,K))
                RDL = UMOL / VK ** 2. / D1 ** 2. /
     &            Max(1.E-10,TKE(I,K))
              FT = Tanh((1.63**2.*RDT)**3.)
              FL = Tanh((3.55**2.*RDL)**10.)
              FE2 = 1.0 - Max(FT,FL)
              ALF1 = 0.25 - D1 / HMAX
              If(ALF1 .GE. 0.0) Then
```

```fortran
                FE1 = 2.  * Exp(-11.09*ALF1**2.)
             Else
                FE1 = 2.  * Exp(-9.0*ALF1**2.)
             Endif
             FE = Max((FE1-1.0),  0.0) * FE2
             FB = Min(2.*Exp(-9.*ALF1**2.),1.0)
             FDT = 1.  - Tanh((8.*RDT)**3.)
               FDTIDLE = Max(1.-FDT,FB)
             DTIDLE = FDTIDLE * (1.  + FE) * D1 + (1.  - FDTIDLE) * D2
               STIDLE = TKE(I,K) +
     &                       VIS(I,K) / VK ** 2.  / DTIDLE ** 2.  * FV2
             STIDLE = Max(STIDLE,  1.E-10)
             R = RDT
             Goto 100
100          Continue
          Else
C-----          RANS MODEL
```

III: $P_k = 2\nu_t S_{ij} \cdot S_{ij} - \frac{2}{3}\rho k \frac{\partial u_i}{\partial x_j}\delta_{ij}$

```fortran
               DTIDLE = 1.0
          Endif
```

ν_t S_{ij}

```fortran
            S1(I,K) = VIS(I,K) * TENSOR(I,K) ** 2.
            S1(I,K) = Max(S1(I,K),0.0)
         If(10.*BETASTAR*TKE(I,K)*TDISS(I,K) .GT.  0.0)
     &    S1(I,K) = Min(S1(I,K),10.*BETASTAR*TKE(I,K)*TDISS(I,K)*DTIDLE)
            VISF(I,K) = VISF(I,K) + AREA(I) * DZ(K) * S1(I,K)
        VISSOURCE(I,K) = BETASTAR * TDISS(I,K) * DTIDLE
          Enddo
```

IV: $\beta^* \cdot \omega \cdot 1$

```fortran
      Endif
   Enddo
!$OMP END DO
      If(POREMODULE .EQ.  'INCLUDE' .AND.  DEM .EQ.  'NEGLECT') Then
!$OMP DO
        Do  I =1,IJM
          If(CCM(I).EQ.1.0)Then
            Do K=1,KBM
              If  (PORE(I,K) .NE.  1.0)  Then
                     VISF(I,K)  =  VISF(I,K) +AREA(I) * DZ(K)*
     &                CFK * 1.  / 2.  * APU(I,K) * CDC*
     &              Sqrt(UR(I,K) ** 2. + VR(I,K) ** 2. + WR(I,K) ** 2.) *
     &              TKE(I,K)
              Endif
            Enddo
```

```fortran
           Endif
          Enddo
!$OMP END DO
       Endif

       If(DEM .NE. 'NEGLECT' .AND. I_PTF .NE. 0) Then
!$OMP DO
          Do I =1,IJM
            If(CCM(I) .EQ. 1.0)Then
              Do K=1,KBM
                !If (PORE(I,K) .NE. 1.0) Then
c                  VISF(I,K) = VISF(I,K) + AREA(I) * DZ(K) * STKEDEM(I,K) ! STKEDEM
                !Endif
              Enddo
            Endif
          Enddo
!$OMP END DO
       Endif
!$OMP BARRIER
C       PAUSE
c============================================================================c
c                    open boundary treatments                                c
c============================================================================c
!$OMP MASTER
C-----   elevation boundary condition
       If(NUMEBC .NE. 0) Then
         Do N = 1, NUMEBC
            ID = IEBC(N)
            IS = IEBCINX(N)
            Do K = 1, KBM
               VISF(ID,K) = 0.0
               UNEBC = UR(ID,K) * CELL_CUV(ID,IS,7) +
     &                   VR(ID,K) * CELL_CUV(ID,IS,8)
               Do J = 1, CELL_POLYGEN(ID)
                    If(CFM(CELL_SIDE(ID,J,1)) .EQ. 1.0) Then
                        VISF(ID,K) = VISF(ID,K) - DZ(K) *
     &                   HQ(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,6) *
     &                   (UN(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,7) +
     &                      VN(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,8))
                  Endif
               Enddo
```

```fortran
              If(UNEBC .GT. 0.0) Then
               VISF(ID,K) = VISF(ID,K) - DZ(K) * CELL_CUV(ID,IS,6)*
     &                        TKE(ID,K) * UNEBC
            Else
                VISF(ID,K) = VISF(ID,K) - DZ(K) * CELL_CUV(ID,IS,6)*
     &                        UNEBC * TKEE(N)
              Endif
              VISSOURCE(ID,K) = 0.0
              TKEMARK(ID,K) = 0.0
          Enddo
        Enddo
      Endif
C-----   astrotidal boundary condition
      If(NUMAST .NE. 0) Then
        Do N = 1, NUMAST
           ID = IABC(N)
           IS = IABCINX(N)
           Do K = 1, KBM
              VISF(ID,K) = 0.0
              UNAST = UR(ID,K) * CELL_CUV(ID,IS,7) +
     &                VR(ID,K) * CELL_CUV(ID,IS,8)
              Do J = 1, CELL_POLYGEN(ID)
                 If(CFM(CELL_SIDE(ID,J,1)) .EQ. 1.0) Then
                    VISF(ID,K) = VISF(ID,K) - DZ(K) *
     &                  HQ(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,6) *
     &                  (UN(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,7) +
     &                     VN(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,8))
                 Endif
              Enddo
              If(UNAST .GT. 0.0) Then
               VISF(ID,K) = VISF(ID,K) - DZ(K) * CELL_CUV(ID,IS,6)*
     &                        TKE(ID,K) * UNAST
              Else
                VISF(ID,K) = VISF(ID,K) - DZ(K) * CELL_CUV(ID,IS,6)*
     &                        UNAST * Sqrt(1.5) * DC(ID) * DZ(K) *
     &                          UNAST * TINT
              Endif
              VISSOURCE(ID,K) = 0.0
              TKEMARK(ID,K) = 0.0
          Enddo
        Enddo
```

```
      Endif
c-----   discharge boundary condition
      If(NUMQBC .NE. 0) Then
         Call BCOND(3)
         Do N = 1, NUMQBC
            ID = IQBC(N)
            IS = IQBCINX(N)
            ISS = CELL_SIDE(ID,IS,1)
            Do K = 1, KBM
               VISF(ID,K) = 0.0
               UNQBC = UN(ISS,K) * CELL_CUV(ID,IS,7) +
     &                 VN(ISS,K) * CELL_CUV(ID,IS,8)
                 Do J = 1, CELL_POLYGEN(ID)
                    If(CFM(CELL_SIDE(ID,J,1)) .EQ. 1.0) Then
                       VISF(ID,K) = VISF(ID,K) - DZ(K) *
     &                   HQ(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,6) *
     &                  (UN(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,7) +
     &                     VN(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,8))
                  Endif
               Enddo
               If(UNQBC .GT. 0.0) Then
                 VISF(ID,K) = VISF(ID,K) - DZ(K) * CELL_CUV(ID,IS,6)*
     &                            TKE(ID,K) * UNQBC
               Else
                 VISF(ID,K) = VISF(ID,K) - DZ(K) * CELL_CUV(ID,IS,6)*
     &                     UNQBC * TKEQ(N,K)
C     &                  3./2. * UNQBC ** 2. * TINT ** 2.
                 Endif
                VISSOURCE(ID,K) = 0.0
                TKEMARK(ID,K) = 0.0
            Enddo
         Enddo
      Endif
C-----   velocity boundary condition
      If(NUMVBC .NE. 0) Then
         Do N = 1, NUMVBC
            ID = IVBC(N)
            IS = IVBCINX(N)
            Do K = 1, KBM
               VISF(ID,K) = 0.0
               UNVBC = UN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,7) +
```

```
     &                   VN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,8)
                 Do J = 1, CELL_POLYGEN(ID)
                    If(CFM(CELL_SIDE(ID,J,1)) .EQ. 1.0) Then
                      VISF(ID,K) = VISF(ID,K) - DZ(K) *
     &                  HQ(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,6) *
     &                 (UN(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,7) +
     &                    VN(CELL_SIDE(ID,J,1),K) * CELL_CUV(ID,J,8))
                  Endif
               Enddo
               If(UNVBC .GT. 0.0) Then
                 VISF(ID,K) = VISF(ID,K) - DZ(K) * CELL_CUV(ID,IS,6)*
     &                       TKE(ID,K) * UNVBC
               Else
                 VISF(ID,K) = VISF(ID,K) - DZ(K) * CELL_CUV(ID,IS,6)*
     &                         UNVBC * 3./2.* UNVBC ** 2. * TINT ** 2.
                 Endif
                 VISSOURCE(ID,K) = 0.0
                 TKEMARK(ID,K) = 0.0
              Enddo
           Enddo
        Endif
!$OMP END MASTER
!$OMP BARRIER
C==============================================================================C
C                      Step forward in time                                    C
C==============================================================================C
    Do K = 1, KBM
!$OMP DO
       Do I = 1, IJM
          If(CCM(I) .EQ. 1.0) Then
             VISF(I,K) = TKE(I,K) * AREA(I) * DZ(K) + DTI * VISF(I,K)
          Endif
        Enddo
!$OMP END DO
    Enddo
!$OMP END PARALLEL
c======================  end subroutine program  ==========================C
      Return
      End
```

VISF : explicit term