```fortran
C#############################################################################
c                                                                           #
c                    SUBROUTINE PROGRAM                                     #
C                  VERSION 1.0 (28/04/2012)                                 #
C                  AUTHORIZED BY ZHANG JINGXIN                              #
C                                SHANGHAI JIAO TONG UNIVERSITY              #
C                                SHANGHAI, CHINA                            #
c---------------------------------------------------------------------------#
c            computes the flow after the hydrodynamics solved               #
c                                                                           #
c#############################################################################
      Subroutine UPDATEFLOW
      Include './Include/OCERM_INF'
    Dimension FLUX(IJM,KB), WUD(IJM,KB), WDU(IJM,KB)
c---------------------------------------------------------------------------c
c                  Calculate new vertical velocity QW                       C
c---------------------------------------------------------------------------c
!$OMP PARALLEL DEFAULT(SHARED) PRIVATE(I,J,K,IL,IR,ELFT,FXE,FYE,FXH,FYH)
!$OMP DO
    Do I = 1, IJM
       If(CCM(I) .EQ. 1.0) Then
          W(I,1) = 0.0
          W(I,KB) = 0.0
          WUD(I,1) = 0.0
          WUD(I,KB) = 0.0
          WDU(I,1) = 0.0
          WDU(I,KB) = 0.0
       Endif
       Do K = 1, KB
          FLUX(I,K) = 0.0
       Enddo
    Enddo
!$OMP END DO
!$BARRIER
c----    flux of every cell  -------------------------------------------------c
    Do K = 1, KBM
!$OMP DO
       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
```

```
                     IL = I
                   IR = CELL_SIDE(I,J,2)
CC                 FLUX(I,K) = FLUX(I,K) + CELL_CUV(I,J,6) *
CC     &                           ((U(IL,K) + U(IR,K)) / 2. *
CC     &                            CELL_CUV(I,J,7) +
CC     &                           (V(IL,K) + V(IR,K)) / 2. *
CC     &                            CELL_CUV(I,J,8))

c                    FLUX(I,K) = FLUX(I,K) + CELL_CUV(I,J,6) *
c     &                        ((U(IL,K) * Sqrt(AREA(IR)) +
c     &                            U(IR,K) * Sqrt(AREA(IL))) /
c     &                           (Sqrt(AREA(IL)) + Sqrt(AREA(IR))) *
c     &                           CELL_CUV(I,J,7) +
c     &                           (V(IL,K) * Sqrt(AREA(IR)) +
c     &                            V(IR,K) * Sqrt(AREA(IL)))/
c     &                           (Sqrt(AREA(IL)) + Sqrt(AREA(IR))) *
c     &                           CELL_CUV(I,J,8))

                FLUX(I,K) = FLUX(I,K) + THITA * CELL_CUV(I,J,6) *
     &                     (0.5 * (U(I,K)+U(CELL_SIDE(I,J,2),K)) *
     &                        CELL_CUV(I,J,7) +
     &                        0.5 * (V(I,K)+V(CELL_SIDE(I,J,2),K)) *
     &                        CELL_CUV(I,J,8)) +
     &                        (1. - THITA) * CELL_CUV(I,J,6) *
     &                        DS(CELL_SIDE(I,J,1)) *
     &                       (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            open boundary                                            c
c=====================================================================c
!$OMP MASTER
C-----  elevation boundary condition
     If(NUMEBC .NE. 0) Then
        Do N = 1, NUMEBC
```

$\theta\cdot\Delta l\cdot(q_x^{n+1}\cos\alpha + q_y^{n+1}\sin\alpha)$

$+(1-\theta)\cdot\Delta l\cdot(q_x^{n}\cos\alpha + q_y^{n}\sin\alpha)$

半隐式时间步

```
          ID = IEBC(N)
          IS = IEBCINX(N)
          Do K = 1, KBM
c            FLUX(ID,K) = 0.0
             FLUX(ID,K) = FLUX(ID,K) + CELL_CUV(ID,IS,6) *
     &                        (U(ID,K) * CELL_CUV(ID,IS,7) +
     &                         V(ID,K) * CELL_CUV(ID,IS,8))
          Enddo
        Enddo
      Endif
c-----   discharge boundary condition
      If(NUMQBC .NE. 0) Then
        Do N = 1, NUMQBC
          ID = IQBC(N)
            IS = IQBCINX(N)
          ISS = CELL_SIDE(ID,IS,1)
          Do K = 1, KBM
             FLUX(ID,K) = FLUX(ID,K) + CELL_CUV(ID,IS,6)* DS(ISS) *
     &                         (UN(ISS,K) * CELL_CUV(ID,IS,7) +
     &                          VN(ISS,K) * CELL_CUV(ID,IS,8))
          Enddo
        Enddo
      Endif
c-----   velocity boundary condition
      If(NUMVBC .NE. 0) Then
        Do N = 1, NUMVBC
          ID = IVBC(N)
            IS = IVBCINX(N)
          ISS = CELL_SIDE(ID,IS,1)
          Do K = 1, KBM
             FLUX(ID,K) = FLUX(ID,K) + CELL_CUV(ID,IS,6)* DS(ISS) *
     &                         (UN(ISS,K) * CELL_CUV(ID,IS,7) +
     &                          VN(ISS,K) * CELL_CUV(ID,IS,8))
          Enddo
        Enddo
      Endif
c-----   offshore discharge boundary condition
      If(NUMDBC .NE. 0) Then
        Do N = 1, NUMDBC
          ID = IDBC(N)
          Do K = 1, KBM
```

```
            FLUX(ID,K) = FLUX(ID,K) -
     &                      QDIFF(N) * VDDIST(N,K) / 100. * RAMP
         Enddo
       Enddo
    Endif
c-----   astrotidle boundary condition
    If(NUMAST .NE. 0) Then
       Do N = 1, NUMAST
            ID = IABC(N)
          IS = IABCINX(N)
          Do K = 1, KBM
             FLUX(ID,K) = 0.0
c            FLUX(ID,K) + CELL_CUV(ID,IS,6) *
c     &                      (U(ID,K) * CELL_CUV(ID,IS,7) +
c     &                       V(ID,K) * CELL_CUV(ID,IS,8))
         Enddo
       Enddo
    Endif
C============================================================================C
C              numerical wave flume                                         c
C============================================================================C
    If(IFLUME .EQ. 1) Then
       Call WAVEGEN(1)
       Do I = 1, N_SOURCE
          II = IGEN(I)
        Do K = 1, KBM
             FLUX(II,K) = FLUX(II,K) -
     &                      WGEN(I,K) * AREA(II) * HC(II)
           ENDDO
       Enddo
    Endif
!$OMP END MASTER
!$OMP BARRIER
C!$OMP DO
C      Do I = 1, IJM
C      If(CCM(I) .EQ. 1.0) Then
C         ELFT = 0.0
C         Do K = 1, KBM
C              ELFT = ELFT + DZ(K) * FLUX(I,K)
C           Enddo
C           ELF(I) = EL(I) - DTI * ELFT / AREA(I)
```

```fortran
C       Endif
C    Enddo
C!$OMP END DO
!$OMP MASTER
c-----     elevation
     If(NUMEBC .NE. 0) Call BCOND(1)
c-----     astrotide boundary
     If(NUMAST .NE. 0) Call BCOND(5)
!$OMP END MASTER
C-----------------------------------------------------------------------------C
C              calculating the vertical velocity                              c
c-----------------------------------------------------------------------------c
!$OMP DO
     Do I = 1, IJM
        If(CCM(I) .EQ. 1.0) Then
           Do K = KBM, 2, -1
              WDU(I,K) = WDU(I,K+1) - DZ(K) * (ELF(I) - EL(I)) / DTI -
     &                   FLUX(I,K) * DZ(K) / AREA(I)
           Enddo

           Do K = 2, KBM
              WUD(I,K) = WUD(I,K-1) + DZ(K-1) * (ELF(I) - EL(I)) / DTI +
     &                   FLUX(I,K-1) * DZ(K-1) / AREA(I)
           Enddo

           Do K = 2, KBM
c             W(I,K) = (WDU(I,K) + WUD(I,K)) / 2.
               W(I,K) = WDU(I,K)      ω̃ 计算域垂向流速
           Enddo

        Endif
     Enddo
!$OMP END DO
!$OMP END PARALLEL

     Return
     End
```