```fortran
C##############################################################################
c                                                                             #
c                   SUBROUTINE PROGRAM                                        #
C                 VERSION 1.0 (25/05/2009)                                    #
C                 AUTHORIZED BY ZHANG JINGXIN                                 #
C                               SHANGHAI JIAO TONG UNIVERSITY                 #
C                               SHANGHAI, CHINA                               #
c-----------------------------------------------------------------------------#
c                     computes the water elevation                            #
c                                                                             #
c##############################################################################
    Subroutine ELTION
    Include './Include/OCERM_INF'
    Include './Include/VORGEN_INF'

    Parameter(SCHEME = 2)
    Common/ELFBLK/CS(IJM,IPOLYGEN),CB(IJM),CP(IJM),X(IJM)
    Dimension AAAA(KBM,KBM), BBBB(KBM), ZZZ1(KBM), ZZZ2(KBM),TEMP(KBM)
    Dimension COES(IJE), WGENDEL(N_SOURCE,KB),
     &           USTARV(IJE,KBM),VSTARV(IJE,KBM)
    Dimension AA(3*KBM-2), BB(KBM)
    Dimension FLU_SOURCE(NUM_CELL,NUM_VER)

    DIMENSION TEMP2(80)
C============================================================================c
c                        initializing the matrixs                            c
c============================================================================c
      Do K1 = 1, KBM
         Do K2 = 1, KBM
          AAAA(K1,K2) = 0.0
       Enddo
      Enddo
!$OMP PARALLEL DEFAULT(SHARED) PRIVATE(I,J,K)
!$OMP DO
    Do I = 1, IJM
       Do J = 1, CELL_POLYGEN(I)
            CS(I,J) = 0.0
       Enddo
       CP(I) = 0.0
       CB(I) = 0.0
       X(I) = 0.0
```

```
    Enddo
!$OMP END DO NOWAIT
!$OMP DO
    Do I = 1, IJM
       Do K = 1, KB
          USTAR(I,K) = 0.0
          VSTAR(I,K) = 0.0
       Enddo
       ESTAR(I) = 0.0
    Enddo
!$OMP END DO NOWAIT
!$OMP DO
       Do I = 1, IJE
          COES(I) = 0.0
       Enddo
!$OMP END DO NOWAIT
!$OMP END PARALLEL
c=============================================================================c
c                    offshore discharge boundary condition                     c
c=============================================================================c
    If(NUMDBC .NE. 0) Call BCOND(4)
C=============================================================================c
c                     coefficients on the side edges                           c
c=============================================================================c
      Do K = 1, KBM
       ZZZ1(K) = DTI * THITA * DZ(K)            Z1i = Δt θ Δσk
       ZZZ2(K) = DTI * (1.0 - THITA) * DZ(K)    Z2i = Δt(1-θ)σk
    Enddo
!$OMP PARALLEL DEFAULT(SHARED) PRIVATE(I,J,K,IL,IR,AAAA,BBBB,TTTT1,
!$OMP&          TTTT2,AA,BB,TEMP,TTTTX,TTTTY,L,PORE_AVE)
!$OMP DO
    Do I = 1, IJE
C      Do K1 = 1, KBM
C         Do K2 = 1, KBM
C            AAAA(K1,K2) = 0.0
C         Enddo
C        Enddo
       If(CFM(I) .EQ. 1.0) Then
          Do K = 2, KBM
             IL = INDEX_EDGE(I,K,1)
             IR = INDEX_EDGE(I,K,2)
```

$$\frac{q_{x_i}^* - q_{x_i}^n}{\Delta t} = F q_{x_i}^n - gD\theta\left(\frac{\partial \zeta^*}{\partial x}\right)_i - gD(1-\theta)\left(\frac{\partial \zeta^n}{\partial x}\right)_i + \left[\frac{\partial}{D\partial\sigma}\left(\frac{\nu_{tv}}{D}\frac{\partial q_x^*}{\partial \sigma}\right)\right]_i$$

$$G_{x_i}^n \qquad B_i^n\left(\frac{\partial \zeta^*}{\partial x}\right)_i$$

$$\frac{q_{x_i}^*}{\Delta t} = \frac{\nu_{tv}}{D^2}\,\frac{q^*_{x,i+1} - 2q^*_{x,i} + q^*_{x,i-1}}{(\Delta\sigma)^2}$$

$$-q^*_{x,i+1}\left(\frac{\nu_{tv}\cdot\Delta t}{D^2\Delta\sigma}\right) - q^*_{x,i-1}\left(\frac{\nu_{tv}\cdot\Delta t}{D^2\Delta\sigma}\right) + \left[\Delta\sigma - 2\cdot\frac{\nu_{tv}\Delta t}{D^2\Delta\sigma}\right] q^*_{x,i}$$

```
            AAAA(K,K-1) = -DTI * 0.5 * ( PORE_HF(I,K-1) + PORE_HF(I,K) ) *
     &                     (UMOL + (.5 * (KM(IL,K-1) + KM(IR,K-1)) +
     &                        .5 * (KM(IL,K) + KM(IR,K))) / 2. ) /
     &                       DS(I) ** 2. / DZZ(K-1)
            AAAA(K-1,K) = AAAA(K,K-1)
         Enddo
         Do K = 2, KBM - 1
              AAAA(K,K) = PORE_HF(I,K) * DZ(K) -
     &                     AAAA(K,K-1) - AAAA(K,K+1)
         Enddo
         IL = INDEX_EDGE(I,1,1)
         IR = INDEX_EDGE(I,1,2)
         If(KBM .GT. 1) Then
            AAAA(1,1) = PORE_HF(I,1) * DZ(1) - AAAA(1,2)
         Else
            AAAA(1,1) = PORE_HF(I,1) * DZ(1)
         Endif
         IL = INDEX_EDGE(I,KBM,1)
         IR = INDEX_EDGE(I,KBM,2)
         If(KBM .GT. 1) Then
           If(WFBC .EQ. '      FUN1') Then             ! wall function for RANS
             If(VERTMIX .EQ. 'SSTMODEL  ') Then
                 AAAA(KBM,KBM) = PORE_HF(I,KBM) * DZ(KBM) -
     &                            AAAA(KBM,KBM-1) + PORE_HF(I,KBM) *
     &                          DTI * (CBC(IL) +CBC(IR)) / 2. /DS(I)
               Else
                 AAAA(KBM,KBM) = PORE_HF(I,KBM) * DZ(KBM) -
     &                            AAAA(KBM,KBM-1) + PORE_HF(I,KBM) *
     &                        DTI * (CBC(IL) +CBC(IR)) / 2. *
     &                Sqrt(UN(I,KBM) ** 2. + VN(I,KBM) ** 2.) / DS(I)
               Endif
           Endif
           If(WFBC .EQ. '      FUN2') Then             ! wall function for LES
             AAAA(KBM,KBM) = PORE_HF(I,KBM) * DZ(KBM) -
     &                          AAAA(KBM,KBM-1) + PORE_HF(I,KBM) *
     &                        DTI * (CBC(IL) + CBC(IR)) / 2. / DS(I)
           Endif
           If(WFBC .EQ. '      FUN3') Then             ! no-slip B.C.
             AAAA(KBM,KBM) = PORE_HF(I,KBM) * DZ(KBM) -
     &                 AAAA(KBM,KBM-1) + PORE_HF(I,KBM) * DTI *
     &               (.5 * (KM(IL,KBM) + KM(IR,KBM)) + UMOL) /
```

```
     &             DS(I) / (DS(I)*DZ(KBM)*.5)
C     -(Z01(IL)+Z01(IR))*.5)
            Endif
          Else
            If(WFBC .EQ. '       FUN1') Then              ! wall function for RANS
               AAAA(KBM,KBM) = PORE_HF(I,KBM) * DZ(KBM) +
     &               PORE_HF(I,KBM) * DTI * (CBC(IL) + CBC(IR))/2.*
     &             Sqrt(UN(I,KBM) ** 2. + VN(I,KBM) ** 2.) / DS(I)
            Endif
            If(WFBC .EQ. '       FUN2') Then              ! wall function for LES
               AAAA(KBM,KBM) = PORE_HF(I,KBM) * DZ(KBM) +
     &               PORE_HF(I,KBM) * DTI*(CBC(IL)+CBC(IR))/2./DS(I)
            Endif
            If(WFBC .EQ. '       FUN3') Then              ! no-slip B.C.
               AAAA(KBM,KBM) = PORE_HF(I,KBM) * DZ(KBM) +
     &               PORE_HF(I,KBM) * DTI *
     &             (.5 * (KM(IL,KBM) + KM(IR,KBM)) + UMOL ) /
     &               DS(I)/ (DS(I)*DZ(KBM)*.5)
C     -(Z01(IL)+Z01(IR))*.5)
            Endif
          Endif
          Do K = 1, KBM
             BBBB(K) = GRAV * DS(I) * DTI * THITA * DZ(K)*          B_i^n = gD_i^n Δt θ Δσ_k
     &                   PORE_HF(I,K)
C            ZZZ1(K) = DTI * THITA * DZ(K)
C            ZZZ2(K) = DTI * (1.0 - THITA) * DZ(K)
            Enddo
            Goto (1, 2) SCHEME
1         Continue
            Call BRINV(AAAA,KBM,L)
C-----   coefficients on the sides
          TTTT1 = 0.0
          Do K = 1, KBM
             TEMP(K) = 0.0
             Do J = 1, KBM
                TEMP(K) = TEMP(K) + AAAA(K,J) * BBBB(J)     A^-1 B
             Enddo
          Enddo
          TTTT2 = 0.0
          Do K = 1, KBM
             TTTT2 = TTTT2 + ZZZ1(K) * TEMP(K) * PORE_HF(I,K)
```

$ZA^{-1}B$

```
          Enddo
          COES(I) = TTTT2
            Goto 50
2         Continue
          AA(1) = AAAA(1,1)
          If(KBM .GT. 1) AA(2) = AAAA(1,2)
            Do K = 2, KBM - 1
              AA(2*(K-1)+K-1) = AAAA(K,K-1)
           AA(2*(K-1)+K)    = AAAA(K,K)
              AA(2*(K-1)+K+1) = AAAA(K,K+1)
            Enddo
          If(KBM .GT. 1) AA(3*KBM-3) = AAAA(KBM,KBM-1)
          AA(3*KBM-2) = AAAA(KBM,KBM)
            Do K = 1, KBM
              BB(K) = BBBB(K)
          Enddo
            Call ATRDE(AA,KBM,3*KBM-2,BB,L)
          Do K = 1, KBM
              TEMP(K) = BB(K)
          Enddo
          TTTT2 = 0.0
          Do K = 1, KBM
              TTTT2 = TTTT2 + ZZZ1(K) * TEMP(K) * PORE_HF(I,K)
          Enddo
          COES(I) = TTTT2
          Goto 50
50        Continue
       Endif
    Enddo
!$OMP END DO
!$OMP BARRIER
C============================================================================C
C                    coefficients of the equations for water elevation      c
c============================================================================c
!$OMP DO
    Do I = 1, IJM
C      Do K1 = 1, KBM
C         Do K2 = 1, KBM
C            AAAA(K1,K2) = 0.0
C         Enddo
C        Enddo
```

```fortran
        IF(CCM(I) .EQ. 1.0) Then
          Do K = 2, KBM
             AAAA(K,K-1) = -DTI * ((KM(I,K-1) + KM(I,K)) / 2. + UMOL) /
     &                    DC(I) ** 2. / DZZ(K-1) * PORE_VF(I,K)
             AAAA(K-1,K) = AAAA(K,K-1)
          Enddo
          Do K = 2, KBM - 1
             AAAA(K,K) = DZ(K) * PORE(I,K) - AAAA(K,K-1) - AAAA(K,K+1)
          Enddo
          If(KBM .GT. 1) Then
               AAAA(1,1) = DZ(1) * PORE(I,1) - AAAA(1,2)
          Else
               AAAA(1,1) = DZ(1) * PORE(I,1)
          Endif
          If(KBM .GT. 1) Then
            If(WFBC .EQ. '      FUN1') Then      ! wall function for RANS
               If(VERTMIX .EQ. 'SSTMODEL  ') Then
                  AAAA(KBM,KBM) = DZ(KBM) * PORE(I,KBM) - AAAA(KBM,KBM-1)
     &                           + DTI*CBC(I)/ DC(I) * PORE(I,KBM)
                 Else
                  AAAA(KBM,KBM) = DZ(KBM) * PORE(I,KBM) - AAAA(KBM,KBM-1)
     &            + DTI*CBC(I) * Sqrt(U(I,KBM) ** 2. + V(I,KBM) ** 2.) /
     &              DC(I) ** 2. * PORE(I,KBM)
                 Endif
            Endif
            If(WFBC .EQ. '      FUN2') Then      ! wall function for LES
               AAAA(KBM,KBM) = DZ(KBM) * PORE(I,KBM) - AAAA(KBM,KBM-1) +
     &                        DTI * CBC(I) / DC(I) * PORE(I,KBM)
            Endif
            If(WFBC .EQ. '      FUN3') Then      ! no-slip B.C.
                 AAAA(KBM,KBM) = DZ(KBM) * PORE(I,KBM) - AAAA(KBM,KBM-1)
     &                          + DTI * (UMOL+KM(I,KBM)) / DC(I) /
     &                          (DC(I)*DZ(KBM)*.5)* PORE(I,KBM)
C     -Z01(I))
            Endif
          Else
            If(WFBC .EQ. '      FUN1') Then      ! wall function for RANS
                 AAAA(KBM,KBM) = DZ(KBM) * PORE(I,KBM) + DTI * CBC(I) *
     &              Sqrt(U(I,KBM) ** 2. + V(I,KBM) ** 2.) / DC(I) ** 2.*
     &                PORE(I,KBM)
            Endif
```

```fortran
          If(WFBC .EQ. '       FUN2') Then       ! wall function for LES
               AAAA(KBM,KBM) = DZ(KBM) * PORE(I,KBM) + DTI * CBC(I) /
     &                         DC(I) * PORE(I,KBM)

          Endif
          If(WFBC .EQ. '       FUN3') Then       ! no-slip B.C.
               AAAA(KBM,KBM) = DZ(KBM) * PORE(I,KBM) + DTI *
     &                       (UMOL+KM(I,KBM)) / DC(I) /
     &                         (DC(I)*DZ(KBM)*.5) * PORE(I,KBM)
C     -Z01(I))
          Endif
        Endif
        Do K = 1, KBM
           BBBB(K) = GRAV * DC(I) * DTI * THITA * DZ(K) * PORE(I,K)
C          ZZZ1(K) = DTI * THITA * DZ(K)
C          ZZZ2(K) = DTI * (1.0 - THITA) * DZ(K)
        Enddo
C==============================================================================c
C                         predictor step 分步计算                               c
C==============================================================================c
        TTTTX = 0.0
        TTTTY = 0.0
C       Do J = 1, CELL_POLYGEN(I)
C          If(CFM(CELL_SIDE(I,J,1)) .EQ. 1.0) Then
C               TTTTX = TTTTX +
C    &                WIX(I,J) * (ELF(CELL_SIDE(I,J,2)) - ELF(I))
C                TTTTY = TTTTY +
C    &                WIY(I,J) * (ELF(CELL_SIDE(I,J,2)) - ELF(I))
C          Endif
C       Enddo
          Goto (10, 20) SCHEME
10      Continue
        Call BRINV(AAAA,KBM,L)
          Do K = 1, KBM
           Do J = 1, KBM
              USTAR(I,K) = USTAR(I,K) + AAAA(K,J) *
     &                     (UF(I,J) - BBBB(J) * TTTTX * AREA(I))
              VSTAR(I,K) = VSTAR(I,K) + AAAA(K,J) *
     &                     (VF(I,J) - BBBB(J) * TTTTY * AREA(I))
           Enddo
           USTAR(I,K) = USTAR(I,K) / AREA(I)
```

$$A_{ix}^{n} Q_{xi}^{*} = G_{xi}^{*}$$

$$\rightarrow Q_{xi}^{*(1)} = A_{ix}^{*-1} \cdot G_{xi}^{*}$$

```fortran
          VSTAR(I,K) = VSTAR(I,K) / AREA(I)
       Enddo
       Goto 100
20     Continue
C-----   QX
       AA(1) = AAAA(1,1)
       If(KBM .GT. 1) AA(2) = AAAA(1,2)
         Do K = 2, KBM - 1
            AA(2*(K-1)+K-1) = AAAA(K,K-1)
          AA(2*(K-1)+K)   = AAAA(K,K)
            AA(2*(K-1)+K+1) = AAAA(K,K+1)
         Enddo
       If(KBM .GT. 1) AA(3*KBM-3) = AAAA(KBM,KBM-1)
       AA(3*KBM-2) = AAAA(KBM,KBM)
         Do K = 1, KBM
          BB(K) = (UF(I,K) - BBBB(K) * TTTTX * AREA(I)) / AREA(I)
       Enddo
         Call ATRDE(AA,KBM,3*KBM-2,BB,L)
       Do K = 1, KBM
          USTAR(I,K) = BB(K)
       Enddo
C-----   QY
       AA(1) = AAAA(1,1)
       If(KBM .GT. 1) AA(2) = AAAA(1,2)
         Do K = 2, KBM - 1
            AA(2*(K-1)+K-1) = AAAA(K,K-1)
          AA(2*(K-1)+K)   = AAAA(K,K)
            AA(2*(K-1)+K+1) = AAAA(K,K+1)
         Enddo
       If(KBM .GT. 1) AA(3*KBM-3) = AAAA(KBM,KBM-1)
       AA(3*KBM-2) = AAAA(KBM,KBM)
       Do K = 1, KBM
          BB(K) = (VF(I,K) - BBBB(K) * TTTTY * AREA(I)) / AREA(I)
       Enddo
         Call ATRDE(AA,KBM,3*KBM-2,BB,L)
       Do K = 1, KBM
          VSTAR(I,K) = BB(K)
       Enddo
       Goto 100
100    Continue
     Endif
```

```
    Enddo
!$OMP END DO
C=============================================================================C
c            discharge boundary conditions                                     C
C=============================================================================C
C   If(NUMQBC .NE. 0) Then
C      Do N = 1, NUMQBC
C         ID = IQBC(N)
C         IS = IQBCINX(N)
C         Do K = 1, KBM
C            TEMP(K) = 0.0
C            ZZZ1(K) = DTI * THITA * DZ(K)
C            ZZZ2(K) = DTI * (1.0 - THITA) * DZ(K)
C            TEMP(K) = DS(CELL_SIDE(ID,IS,1)) *
C     &          (UN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,7) +
C     &               VN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,8)) *
C     &               CELL_CUV(ID,IS,6)
C            TEMP(K) = (ZZZ1(K) + ZZZ2(K)) * TEMP(K)
C            ESTAR(ID) = ESTAR(ID) - TEMP(K) / AREA(ID)
C         Enddo
C      Enddo
C   Endif
C=============================================================================C
c            velocity boundary conditions                                     C
C=============================================================================C
C   If(NUMVBC .NE. 0) Then
C      Do N = 1, NUMVBC
C         ID = IVBC(N)
C         IS = IVBCINX(N)
C         Do K = 1, KBM
C            TEMP(K) = 0.0
C            ZZZ1(K) = DTI * THITA * DZ(K)
C            ZZZ2(K) = DTI * (1.0 - THITA) * DZ(K)
C            TEMP(K) = DS(CELL_SIDE(ID,IS,1)) *
C     &          (UN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,7) +
C     &               VN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,8)) *
C     &               CELL_CUV(ID,IS,6)
C            TEMP(K) = (ZZZ1(K) + ZZZ2(K)) * TEMP(K)
C            ESTAR(ID) = ESTAR(ID) - TEMP(K) / AREA(ID)
C         Enddo
C      Enddo
```

```
C   Endif
c---------------------------------------------------------------------------c
c-----   source term for wave generation in numerical wave flume  -----------c
C   If(IFLUME .EQ. 1) Then
C      Call WAVEGEN(0)
C      Do I = 1, N_SOURCE
C         II = IGEN(I)
C         WSG = 0.0
C         Do K = 1, KBM
C            WSG = WSG + HC(II) * DZ(K) * WGEN(I,K)
C           ENDDO
C         ESTAR(II) = ESTAR(II) + WSG * DTI
C      Enddo
C   Endif
c---------------------------------------------------------------------------c
C===========================================================================c
c           coefficient matrix for water elevation calculation              c
C===========================================================================c
    Do K = 1, KBM
!$OMP DO
       Do I = 1, IJP
         USTARV(I,K) = 0.0
         VSTARV(I,K) = 0.0
         Do J = 1, POINT_CELL(I)
            If(CCM(INDEX_POINT(I,J)) .EQ. 1.0) Then
                USTARV(I,K) = USTARV(I,K) + USTAR(INDEX_POINT(I,J),K)*RCOE(I,J)
                VSTARV(I,K) = VSTARV(I,K) + VSTAR(INDEX_POINT(I,J),K)*RCOE(I,J)
              Endif
           Enddo
         Enddo
!$OMP END DO
      Enddo
!$OMP DO
    Do I = 1, IJM
       If(CCM(I) .EQ. 1.0) Then
         PORE_AVE = 0.0
         Do K = 1,KBM
            PORE_AVE = PORE_AVE + PORE(I,K) * DZ(K)
         Enddo
C-----     block1
         Do K = 1, KBM
```

$$\varsigma_i^* - Z_{1i}\left\{\frac{\partial}{\partial x}\left[A^{n-1}B^n\left(\frac{\partial \varsigma^*}{\partial x}\right)\right]\right\}_i - Z_{1i}\left\{\frac{\partial}{\partial y}\left[A^{n-1}B^n\left(\frac{\partial \varsigma^*}{\partial y}\right)\right]\right\}_i = \langle BB_i\rangle$$

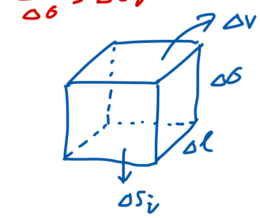
$$\Rightarrow \varsigma_i^*\Delta S_i - Z_{1i}\sum_{s=1}^{NS} A_i^{n-1}B_i^n\left\langle\frac{\partial \varsigma^*}{\partial x}\right\rangle_{is}^f \cos\alpha_{is}\,\Delta l_{is} - Z_{1i}\sum_{s=1}^{NS} A_i^{n-1}B_i^n\left\langle\frac{\partial \varsigma^*}{\partial y}\right\rangle_{is}^f \sin\alpha_{is}\,\Delta l_{is} = \langle BB_i\rangle$$

$$★ \int_{cv} \nabla\cdot\phi\, dv = \int_{cs} n\cdot\phi\, ds$$

$$\Rightarrow \nabla\cdot\phi\,\Delta V = n_x\cdot\phi\cdot\Delta S$$

$$\Rightarrow \nabla\cdot\phi\,\Delta V = \sum \cos\alpha\cdot\phi\cdot\Delta l\cdot\Delta\sigma$$

$$\frac{\Delta V}{\Delta\sigma} = \Delta S_i$$



$$Q^{n'} = A^{n-1}G^n$$

```
      TEMP(K) = 0.0
        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)

              TEMP(K) = TEMP(K) + CELL_CUV(I,J,6) *
     &                  PORE_HF(CELL_SIDE(I,J,1),K) *
     &                  ( (USTAR(IL,K) + USTAR(IR,K)) / 2. *
     &                  CELL_CUV(I,J,7) +
     &                  (VSTAR(IL,K) + VSTAR(IR,K)) / 2. *
     &                  CELL_CUV(I,J,8) )
           Endif
        Enddo
      Enddo
      Do K = 1, KBM
        CB(I) = CB(I) + ZZZ1(K) * TEMP(K)
      Enddo
```

中间步积分

$$\int_{cv} Z_{1i}\frac{\partial Q_x^{n'}}{\partial x} + Z_{1i}\frac{\partial Q_y^{n'}}{\partial y}\, dv$$

$$= \int_{cs} Z_{1i}(n_x\cdot Q_x^{n'} + n_y\cdot Q_y^{n'})\, ds$$

$$= Z_{1i}(\cos\alpha\, Q_x^{n'} + \sin\alpha\, Q_y^{n'})\,\Delta l\cdot\cancel{\Delta\sigma}$$

```
C-----  block2
      Do K = 1, KBM
        TEMP(K) = 0.0
          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)

                TEMP(K) = TEMP(K) + PORE_HF(CELL_SIDE(I,J,1),K) *
     &                  CELL_CUV(I,J,6) *
     &                  ( (U(IL,K) + U(IR,K)) / 2. *
     &                  CELL_CUV(I,J,7) +
     &                  (V(IL,K) + V(IR,K)) / 2. *
     &                  CELL_CUV(I,J,8) )

           Endif
        Enddo
      Enddo
      Do K = 1, KBM
        CB(I) = CB(I) + ZZZ2(K) * TEMP(K)
      Enddo
C-----  CB(I) terms
```

显式项积分

$$\int_{cv} Z_{1i}\frac{\partial Q_x^{n}}{\partial x} + Z_{1i}\frac{\partial Q_y^{n}}{\partial y}\, dv$$

$$= \int_{cs} Z_{1i}(n_x\cdot Q_x^{n} + n_y\cdot Q_y^{n})\, ds$$

$$= Z_{1i}(\cos\alpha\, Q_x^{n} + \sin\alpha\, Q_y^{n})\,\Delta l\cdot\cancel{\Delta\sigma}$$

右端显式项

$\xi_i^n \cdot \Delta S_i \to \frac{\Delta V}{\Delta \sigma}$

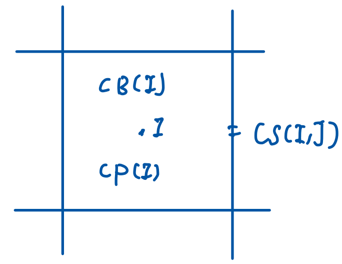
```
      CB(I) = -CB(I) + AREA(I) * EL(I) * PORE_AVE
c-----------------------------------------------------------------------------c
         Do J = 1, CELL_POLYGON(I)
            If(CFM(CELL_SIDE(I,J,1)) .EQ. 1.0) Then
               CS(I,J) = COES(CELL_SIDE(I,J,1)) *
     &                   (DISCOE(I,J,1) - DISCOE(I,J,8))
            Endif
            CP(I) = CP(I) + CS(I,J)
         Enddo
         CP(I) = CP(I) + AREA(I) * PORE_AVE
      Endif
   Enddo
!$OMP END DO
!$OMP END PARALLEL
```

局部坐标计算

$$\left\langle \frac{\partial \xi^*}{\partial x} \right\rangle_{is}^{f} = \frac{1}{J_{is}} \left\langle \frac{\partial \xi^*}{\partial \xi} y_\eta - \frac{\partial \xi^*}{\partial \eta} y_\xi \right\rangle_{is}^{f}$$

$\Delta S_i + \sum AP_{is}$

$$\underbrace{AP_i \xi_i^*}_{CP(I)} - \sum_{s=1}^{NS} \underbrace{AP_{is}}_{CS(I,J)} \xi_{is}^* = \underbrace{\langle \tilde{B}_i \rangle}_{CB(I)}$$




```
C=============================================================================C
C                        boundary conditions                                  c
C=============================================================================C
c-----   discharge boundary conditions
    If(NUMQBC .NE. 0) Then
       Do N = 1, NUMQBC
          ID = IQBC(N)
          IS = IQBCINX(N)
          Do K = 1, KBM
             ZZZ1(K) = DTI * THITA * DZ(K)
             ZZZ2(K) = DTI * (1.0 - THITA) * DZ(K)
C            TEMP(K) = CELL_CUV(ID,IS,6) * (
C     &                USTAR(ID,K) * CELL_CUV(ID,IS,7) +
C     &                VSTAR(ID,K) * CELL_CUV(ID,IS,8))
C            TEMP(K) = ZZZ1(K) * TEMP(K)
C            CB(ID) = CB(ID) - TEMP(K)

             TEMP(K) = -QDIS(N) * VQDIST(N,K) / 100. * RAMP / DZ(K)
c            TEMP(K) = CELL_CUV(ID,IS,6) * (
c     &                DS(CELL_SIDE(ID,IS,1)) *
c     &                UN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,7) +
c     &                DS(CELL_SIDE(ID,IS,1)) *
c     &                VN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,8))
             TEMP(K) = (ZZZ1(K) + ZZZ2(K)) * TEMP(K)
             CB(ID) = CB(ID) - TEMP(K)
          Enddo
       Enddo
```

```fortran
      Endif
c-----    velocity boundary conditions
      If(NUMVBC .NE. 0) Then
        Do N = 1, NUMVBC
          ID = IVBC(N)
          IS = IVBCINX(N)
          Do K = 1, KBM
            ZZZ1(K) = DTI * THITA * DZ(K)
            ZZZ2(K) = DTI * (1.0 - THITA) * DZ(K)
C           TEMP(K) = CELL_CUV(ID,IS,6) * (
C     &                    USTAR(ID,K) * CELL_CUV(ID,IS,7) +
C     &                    VSTAR(ID,K) * CELL_CUV(ID,IS,8))
C           TEMP(K) = ZZZ1(K) * TEMP(K)
C           CB(ID) = CB(ID) - TEMP(K)

            TEMP(K) = CELL_CUV(ID,IS,6) * (
     &                   DS(CELL_SIDE(ID,IS,1)) *
     &                   UN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,7) +
     &                   DS(CELL_SIDE(ID,IS,1)) *
     &                   VN(CELL_SIDE(ID,IS,1),K) * CELL_CUV(ID,IS,8))
            TEMP(K) = (ZZZ1(K) + ZZZ2(K)) * TEMP(K)
            CB(ID) = CB(ID) - TEMP(K)
          Enddo
        Enddo
      Endif
c-----    elevation

      If(NUMEBC .NE. 0) Then
       Call BCOND(1)

       Do N = 1, NUMEBC
          ID = IEBC(N)
          CP(ID) = 1.0
          Do J = 1, CELL_POLYGEN(ID)
             CS(ID,J) = 0.0
          Enddo
          CB(ID) = ELF(ID)
       Enddo
      Endif
c-----    discharge / offshore
      If(NUMDBC .NE. 0) Then
```

```
      Do N = 1, NUMDBC
         ID = IDBC(N)
         CB(ID) = CB(ID) + QDIFF(N) * DTI*RAMP
      Enddo
    Endif
c-----    astrotide boundary
    If(NUMAST .NE. 0) Then
      Call  BCOND(5)
      Do N = 1, NUMAST
         ID = IABC(N)
         CP(ID) = 1.0
         Do J = 1, CELL_POLYGEN(ID)
            CS(ID,J) = 0.0
         Enddo
         CB(ID) = ELF(ID)
      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)
          WSG = 0.0
        Do K = 1, KBM
              WSG = WSG + DZ(K) * AREA(II) * WGEN(I,K)
           Enddo
           CB(II) = CB(II) + WSG * DTI
       Enddo
    Endif
C---------------------------------------------------------------------------C
C                Fluctuation generating                                     c
C---------------------------------------------------------------------------C
    If(DES .EQ. 'SAZDES1') Then
         Do I = 1, NUM_CELL
            ID = ID_CELL(I)
            IS = ID_CELL_EDGE(I)
            Do K = 1, NUM_VER
               FLU_SOURCE(I,K) = DS(CELL_SIDE(ID,IS,1)) *
     &              CELL_CUV(ID,IS,6) *
```

```fortran
     &              (UDIS(I,K) * CELL_CUV(ID,IS,7) +
     &               VDIS(I,K) * CELL_CUV(ID,IS,8))
            Enddo
         Enddo
         Do I = 1, NUM_CELL
            ID = ID_CELL(I)
            WSG = 0.0
            Do K = 1, NUM_VER
               WSG = WSG +  DZ(K) * FLU_SOURCE(I,K) * 2.
            Enddo
c           TEMP2(I) = WSG
            CB(ID) = CB(ID) + DTI * (WSG + WDIS(I,1) * AREA(ID))
         Enddo
      Endif
C---------------------------------------------------------------------------C
c         solve the water elevation based on iteration method               c
C---------------------------------------------------------------------------C
C      Call SOLVEELFSORCG
C   Call SOLVEELF
    CALL SOLVEELFPOLCG
!$OMP PARALLEL DO DEFAULT(SHARED) PRIVATE(I)
    Do I = 1, IJM
       If(CCM(I) .EQ. 1.0) Then
          ELF(I) = X(I)
       Endif
    Enddo
!$OMP END PARALLEL DO
C---------------------------------------------------------------------------C
      Return
      End
```