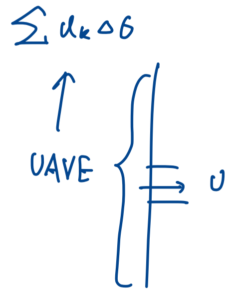
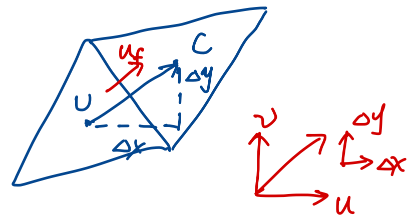
```fortran
C############################################################################
c                                                                          #
c                    SUBROUTINE PROGRAM                                    #
C                  VERSION 1.0 (25/05/2009)                                #
C                  AUTHORIZED BY ZHANG JINGXIN                             #
C                                SHANGHAI JIAO TONG UNIVERSITY             #
C                                SHANGHAI, CHINA                           #
c--------------------------------------------------------------------------#
c                      computes the velocity                               #
c                                                                          #
c############################################################################
      Subroutine PROFV
      Include './Include/OCERM_INF'
    Parameter(SCHEME = 2, I_METHOD = 2)
    Dimension AAAA(KBM,KBM), BBBB(KBM)
    Dimension AA(3*KBM-2), BB(KBM)
    Integer L
    Dimension GRAD2X(IJM),GRAD2Y(IJM)
C==========================================================================C
C                        matrix of the equations                           c
C==========================================================================C
      Do K1 = 1, KBM
         Do K2 = 1, KBM
          AAAA(K1,K2) = 0.0
       Enddo
    Enddo
    Do K1 = 1, 3 * KBM - 2
       AA(K1) = 0.0
    Enddo
    Do K1 = 1, KBM
       BB(K1) = 0.0
       BBBB(K1) = 0.0
    Enddo
!$OMP PARALLEL DEFAULT(SHARED) PRIVATE(I,J,K,K1,K2,AAAA,BBBB,AA,BB,BU,
!$OMP&          BV,IL,IR,UD)
C----   Redefine the upwind cells     -------------------------------------c
!$OMP DO
    Do I = 1, IJM
       If(CCM(I) .EQ. 1.0) Then
         UAVE(I) = 0.0
         VAVE(I) = 0.0
```




```
           Do K = 1, KBM
              UAVE(I) = UAVE(I) + U(I,K) * DZ(K)
              VAVE(I) = VAVE(I) + V(I,K) * DZ(K)
           Enddo
         Endif
      Enddo
!$OMP END DO
!$OMP DO
         Do I = 1, IJE
            If(CFM(I) .EQ. 1.0) Then
              IL = INDEX_EDGE(I,1,1)
              IR = INDEX_EDGE(I,1,2)
```



$u_f \times \Delta r$

```
              UD = .5 * (UAVE(IL) + UAVE(IR)) * (CXY(IR,1) - CXY(IL,1)) +
     &             .5 * (VAVE(IL) + VAVE(IR)) * (CXY(IR,2) - CXY(IL,2))
              If(UD .LT. 0.0) Then
                INDEX_EDGE(I,1,1) = IR
                INDEX_EDGE(I,1,2) = IL
              Endif
            Endif
         Enddo
!$OMP END DO
c-----------------------------------------------------------------------------c
!$OMP DO
         Do I = 1, IJM
           GRAD2X(I) = 0.0
           GRAD2Y(I) = 0.0
           If(CCM(I) .EQ. 1.0) Then
              If(I_METHOD .EQ. 1) Then
                Do J = 1, CELL_POLYGEN(I)
                  If(CFM(CELL_SIDE(I,J,1)) .EQ. 1.0) Then
                    GRAD2X(I) = GRAD2X(I) + CELL_CUV(I,J,6) *
     &                   (ELF(CELL_SIDE(I,J,2)) + ELF(I)) / 2. *
     &                    CELL_CUV(I,J,7)
                    GRAD2Y(I) = GRAD2Y(I) + CELL_CUV(I,J,6) *
     &                   (ELF(CELL_SIDE(I,J,2)) + ELF(I)) / 2. *
     &                    CELL_CUV(I,J,8)
                  Else
                   GRAD2X(I) = GRAD2X(I) + CELL_CUV(I,J,6) *
     &                   ELF(I) * CELL_CUV(I,J,7)
                   GRAD2Y(I) = GRAD2Y(I) + CELL_CUV(I,J,6) *
     &                   ELF(I) * CELL_CUV(I,J,8)
```

判断上下风向

$$\int_{cv} \nabla \xi \, dv = \int_{cs} \vec{n} \xi \, ds$$

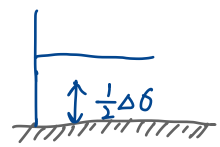
$$\nabla \xi A \Delta\sigma = \xi_f \Delta l \, n_i \Delta\sigma$$

$$\nabla \xi = \frac{\xi_f \Delta l}{A} n_i$$

```
                Endif
             Enddo
             GRAD2X(I) = GRAD2X(I) / AREA(I)
             GRAD2Y(I) = GRAD2Y(I) / AREA(I)
           Endif
           If(I_METHOD .EQ. 2) Then
              Do J = 1, CELL_POLYGEN(I)
               If(CFM(CELL_SIDE(I,J,1)) .EQ. 1.0) Then
                  GRAD2X(I) = GRAD2X(I) +
     &                       WIX(I,J) * (ELF(CELL_SIDE(I,J,2)) - ELF(I))
                  GRAD2Y(I) = GRAD2Y(I) +
     &                       WIY(I,J) * (ELF(CELL_SIDE(I,J,2)) - ELF(I))
               Endif
              Enddo
            Endif
          Endif
      Enddo
!$OMP END DO
!$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
         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
```

```fortran
              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
          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) / 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)
        Enddo
C         BU = 0.0
C       BV = 0.0
         BU = GRAD2X(I)
```

$\frac{\Delta t\, \nu_t}{D^2 \cdot (\frac{1}{2}\Delta\sigma)}$



$B_i^n = g D_i^n \Delta t\, \theta\, \Delta\sigma$

$\left(\frac{\partial \zeta^*}{\partial x}\right)_i$

```
         BV = GRAD2Y(I)
```

$\left(\frac{\partial \xi^*}{\partial y}\right)_i$

```
C        Do J = 1, CELL_POLYGEN(I)
C           If(CFM(CELL_SIDE(I,J,1)) .EQ. 1.0) Then
C               BU = BU + WIX(I,J) * (ELF(CELL_SIDE(I,J,2)) - ELF(I))
C               BV = BV + WIY(I,J) * (ELF(CELL_SIDE(I,J,2)) - ELF(I))
C             BU = BU + 0.5 * (ELF(I) + ELF(CELL_SIDE(I,J,2))) *   ! GAUSSIAN METHOD
C    &                CELL_CUV(I,J,7) * CELL_CUV(I,J,6)
C             BV = BV + 0.5 * (ELF(I) + ELF(CELL_SIDE(I,J,2))) *
C     &                CELL_CUV(I,J,8) * CELL_CUV(I,J,6)
C           Else
C             BU = BU + ELF(I) * CELL_CUV(I,J,7) * CELL_CUV(I,J,6)
C             BV = BV + ELF(I) * CELL_CUV(I,J,8) * CELL_CUV(I,J,6)
C           Endif
C           Enddo
         Goto (1,2) SCHEME
C============================================================================C
C                  solve by operation of matrixs                             c
C============================================================================C
1        Continue
           Call BRINV(AAAA,KBM,L)
           Do K = 1, KBM
            U(I,K) = 0.0
            V(I,K) = 0.0
            Do J = 1, KBM
               U(I,K) = U(I,K) - AAAA(K,J) * BBBB(J) * BU
               V(I,K) = V(I,K) - AAAA(K,J) * BBBB(J) * BV
            Enddo
C           U(I,K) = U(I,K) / AREA(I)   ! GAUSSIAN METHOD
C           V(I,K) = V(I,K) / AREA(I)
         Enddo
         Goto 100
C============================================================================C
C            solve by forward elimination and back-substitution              c
C============================================================================C
2          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)  = -BBBB(K)  * BU
```

$$A_{ix}^{n} Q_{xi}^{*} = -B_{i}^{n}\left(\frac{\partial \xi^{*}}{\partial x}\right)_{i}$$

```
          Enddo
            Call ATRDE(AA,KBM,3*KBM-2,BB,L)
          Do K = 1,  KBM
              U(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)  = -BBBB(K)  * BV
          Enddo
            Call ATRDE(AA,KBM,3*KBM-2,BB,L)
          Do K = 1,  KBM
              V(I,K)  = BB(K)
          Enddo
          Goto 100
100     Continue
        Endif
      Enddo
!$OMP END DO
c    print*,  (elf(i),u(i,kbm),v(i,kbm),i=1,2)
c    pause
C============================================================================C
C                    elf,  qx and qy at n+1 time step                        c
C============================================================================C
!$OMP DO
      Do I = 1,  IJM
        If(CCM(I) .EQ.  1.0)  Then
```

$A^n_{ix} Q^{(1)}_{xi} = -B^n_i \left(\frac{\partial \zeta^*}{\partial x}\right)_i$

$A^n_{ix} Q^{(2)}_{xi} = G^n_{xi}$

```
          Do K = 1,  KBM
             U(I,K)  = U(I,K)  + USTAR(I,K)
             V(I,K)  = V(I,K)  + VSTAR(I,K)
          Enddo
C         ELF(I)  = ELF(I)  + EL(I)
       Endif
    Enddo
!$OMP END DO
!$OMP END PARALLEL
C----------------------------------------------------------------------------C
      Return
      End
```