# Review bdyvol bug fix



For reference, Pierre has added:

| |
|---|
| Ticket: https://forge.ipsl.jussieu.fr/nemo/ticket/2200 |
| Branch: NEMO/branches/UKMO/dev_r10448_bdyvol (https://forge.ipsl.jussieu.fr/nemo/browser/NEMO/branches/UKMO/dev_r10448_bdyvol) |

Diffs are (revision 10448-10476)

```
Files dev_r10448_bdyvol/doc/latex/NEMO/subfiles/chap_LBC.tex and dev_r10448_bdyvol_CLEAN/doc/latex/NEMO/subfiles/chap_LBC.tex differ


Files dev_r10448_bdyvol/src/OCE/BDY/bdyini.F90 and dev_r10448_bdyvol_CLEAN/src/OCE/BDY/bdyini.F90 differ
Files dev_r10448_bdyvol/src/OCE/BDY/bdyvol.F90 and dev_r10448_bdyvol_CLEAN/src/OCE/BDY/bdyvol.F90 differ
Files dev_r10448_bdyvol/src/OCE/DYN/dynspg_ts.F90 and dev_r10448_bdyvol_CLEAN/src/OCE/DYN/dynspg_ts.F90 differ
```

## BDY/bdyini.F90

### SUBROUTINE bdy_init

removes **flagu** and **flagv**

Adds sanity check if tides are used:

```
48c48
<    !! $Id: ShelfSeasCodeReview8dyVolBugFix.txt,v 1.2 2019/01/09 10:54:22 Enda0Dea Exp $
---
>    !! $Id: ShelfSeasCodeReview8dyVolBugFix.txt,v 1.2 2019/01/09 10:54:22 Enda0Dea Exp $
134a135
>         REAL(wp), POINTER  ::  flagu, flagv                   !    -   -
386,394d388
<          !
<          ! sanity check if used with tides
<          IF( ln_tide ) THEN
<             IF(lwp) WRITE(numout,*) ' The total volume correction is not working with tides. '
<             IF(lwp) WRITE(numout,*) ' Set ln_vol to .FALSE. '
<             IF(lwp) WRITE(numout,*) ' or '
<             IF(lwp) WRITE(numout,*) ' equilibriate your bdy input files '
<             CALL ctl_stop( 'The total volume correction is not working with tides.' )
<          END IF
1253a1246,1279
>
>       ! Compute total lateral surface for volume correction:
>       ! ----------------------------------------------------
>       ! JC: this must be done at each time step with non-linear free surface
>       bdysurftot = 0._wp
>       IF( ln_vol ) THEN
>          igrd = 2      ! Lateral surface at U-points
>          DO ib_bdy = 1, nb_bdy
>             DO ib = 1, idx_bdy(ib_bdy)%nblenrim(igrd)
>                nbi => idx_bdy(ib_bdy)%nbi(ib,igrd)
>                nbj => idx_bdy(ib_bdy)%nbj(ib,igrd)
>                flagu => idx_bdy(ib_bdy)%flagu(ib,igrd)
>                bdysurftot = bdysurftot + hu_n   (nbi  , nbj)                          &
>                   &                    * e2u    (nbi  , nbj) * ABS( flagu ) &
>                   &                    * tmask_i(nbi  , nbj)                          &
>                   &                    * tmask_i(nbi+1, nbj)
>             END DO
>          END DO
>
>          igrd=3 ! Add lateral surface at V-points
>          DO ib_bdy = 1, nb_bdy
>             DO ib = 1, idx_bdy(ib_bdy)%nblenrim(igrd)
>                nbi => idx_bdy(ib_bdy)%nbi(ib,igrd)
>                nbj => idx_bdy(ib_bdy)%nbj(ib,igrd)
>                flagv => idx_bdy(ib_bdy)%flagv(ib,igrd)
>                bdysurftot = bdysurftot + hv_n   (nbi, nbj  )                          &
>                   &                    * e1v    (nbi, nbj  ) * ABS( flagv ) &
>                   &                    * tmask_i(nbi, nbj  )                          &
>                   &                    * tmask_i(nbi, nbj+1)
>             END DO
>          END DO
>          !
>          CALL mpp_sum( 'bdyini', bdysurftot )      ! sum over the global domain
>       END IF
1260a1287
>
```

removed (moved) section that calculates lateral surface for volume correction (this needs to be done not just as once off)

## BDY/bdyvol.F90

## dev_r10448_bdyvol_CLEAN/src/OCE/BDY/bdyvol.F90

```fortran
+--  4 lines: MODULE bdyvol
   !!                 and filtered free surface are used
   !!===================================================================
   !! History :  1.0  !  2005-01  (J. Chanut, A. Sellar)  Original code
   !!             -   !  2006-01  (J. Chanut) Bug correction
   !!            3.0  !  2008-04  (NEMO team)  add in the reference version
   !!            3.4  !  2011     (D. Storkey) rewrite in preparation for OBC-BDY merge
------
   !!----------------------------------------------------------------------
   USE oce            ! ocean dynamics and tracers
   USE bdy_oce        ! ocean open boundary conditions
   USE sbc_oce        ! ocean surface boundary conditions
   USE dom_oce        ! ocean space and time domain
   USE phycst         ! physical constants
+--  3 lines: USE sbcisf         ice shelf
   USE lib_mpp        ! for mppsum
   USE lib_fortran    ! Fortran routines library

   IMPLICIT NONE
   PRIVATE

   PUBLIC   bdy_vol    ! called by ???

   !!----------------------------------------------------------------------
   !! NEMO/OCE 4.0 , NEMO Consortium (2018)
   !! $Id: ShelfSeasCodeReviewBdyVolBugFix.txt,v 1.2 2019/01/09 10:54:22 EndaODea Exp $
   !! Software governed by the CeCILL? license (see ./LICENSE)
   !!----------------------------------------------------------------------
CONTAINS

   SUBROUTINE bdy_vol( kt )
      !!----------------------------------------------------------------------
      !!                      ***  ROUTINE bdyvol  ***
      !!
      !! ** Purpose :   This routine controls the volume of the system.
      !!      A correction velocity is calculated to correct the total transport
      !!      through the unstructured OBC.
      !!      The total depth used is constant (H0) to be consistent with the
      !!      linear free surface coded in OPA 8.2    <<<=== !!gm  ???? true ????
      !!
      !! ** Method  :   The correction velocity (zubtpecor here) is defined calculating
      !!      the total transport through all open boundaries (trans_bdy) minus
      !!      the cumulate E-P flux (z_cflxemp) divided by the total lateral
      !!      surface (bdysurftot) of the unstructured boundary.
      !!         zubtpecor = [trans_bdy - z_cflxemp ]*(1./bdysurftot)
+-- 11 lines: !      with z_cflxemp => sum of (Evaporation minus Precipitation)
      !!         2/ The volume is constant even with E-P flux. In this case
      !!            the correction velocity must balance both the flux
      !!            through open boundaries and the ones through the free
      !!            surface.
      !!            (set nn_volctl to 1 in tne namelist for this option)
      !!----------------------------------------------------------------------
      INTEGER, INTENT(in) ::   kt   ! ocean time-step index
      !
      INTEGER  ::   ji, jj, jk, jb, jgrd
      INTEGER  ::   ib_bdy, ii, ij
      REAL(wp) ::   zubtpecor, z_cflxemp, ztranst
------
------
------
      TYPE(OBC_INDEX), POINTER :: idx
      !!-----------------------------------------------------------------------------
      !
      IF( ln_vol ) THEN
      !
      IF( kt == nit000 ) THEN
         IF(lwp) WRITE(numout,*)
         IF(lwp) WRITE(numout,*)'bdy_vol : Correction of velocities along unstructured OBC'
         IF(lwp) WRITE(numout,*)'~~~~~~~'
      END IF

      ! Calculate the cumulate surface Flux z_cflxemp (m3/s) over all the domain
      ! -----------------------------------------------------------------------
!!gm replace these lines :
      z_cflxemp = SUM ( ( emp(:,:) - rnf(:,:) + fwfisf(:,:) ) * bdytmask(:,:) * e1e2t(:,:) ) / rau0
      CALL mpp_sum( 'bdyvol', z_cflxemp )     ! sum over the global domain
!!gm   by :
!!gm      z_cflxemp = glob_sum(  'bdyvol', ( emp(:,:)-rnf(:,:)+fwfisf(:,:) ) * bdytmask(:,:) * e1e2t(:,:)  ) / rau0
!!gm
------
------
------
------
------

      ! Transport through the unstructured open boundary
      ! ------------------------------------------------
      zubtpecor = 0._wp
      DO ib_bdy = 1, nb_bdy
         idx => idx_bdy(ib_bdy)
         !
         jgrd = 2                               ! cumulate u component contribution first
         DO jb = 1, idx%nblenrim(jgrd)
            DO jk = 1, jpkm1
               ii = idx%nbi(jb,jgrd)
               ij = idx%nbj(jb,jgrd)
               zubtpecor = zubtpecor + idx%flagu(jb,jgrd) * ua(ii,ij, jk) * e2u(ii,ij) * e3u_n(ii,ij,jk)
            END DO
         END DO
         jgrd = 3                               ! then add v component contribution
         DO jb = 1, idx%nblenrim(jgrd)
            DO jk = 1, jpkm1
               ii = idx%nbi(jb,jgrd)
               ij = idx%nbj(jb,jgrd)
               zubtpecor = zubtpecor + idx%flagv(jb,jgrd) * va(ii,ij, jk) * e1v(ii,ij) * e3v_n(ii,ij,jk)
            END DO
         END DO
         !
      END DO
      CALL mpp_sum( 'bdyvol', zubtpecor )   ! sum over the global domain

      ! The normal velocity correction
      ! ------------------------------
      IF( nn_volctl==1 ) THEN   ;   zubtpecor = ( zubtpecor - z_cflxemp ) / bdysurftot
      ELSE                      ;   zubtpecor =   zubtpecor               / bdysurftot
      END IF

      ! Correction of the total velocity on the unstructured boundary to respect the mass flux conservation
      ! -------------------------------------------------------
      ztranst = 0._wp
      DO ib_bdy = 1, nb_bdy
         idx => idx_bdy(ib_bdy)
         !
         jgrd = 2                               ! correct u component
         DO jb = 1, idx%nblenrim(jgrd)
            DO jk = 1, jpkm1
               ii = idx%nbi(jb,jgrd)
               ij = idx%nbj(jb,jgrd)
               ua(ii,ij,jk) = ua(ii,ij,jk) - idx%flagu(jb,jgrd) * zubtpecor * umask(ii,ij,jk)
               ztranst = ztranst + idx%flagu(jb,jgrd) * ua(ii,ij,jk) * e2u(ii,ij) * e3u_n(ii,ij,jk)
            END DO
         END DO
         jgrd = 3                              ! correct v component
         DO jb = 1, idx%nblenrim(jgrd)
            DO jk = 1, jpkm1
               ii = idx%nbi(jb,jgrd)
               ij = idx%nbj(jb,jgrd)
               va(ii,ij,jk) = va(ii,ij,jk) -idx%flagv(jb,jgrd) * zubtpecor * vmask(ii,ij,jk)
               ztranst = ztranst + idx%flagv(jb,jgrd) * va(ii,ij,jk) * e1v(ii,ij) * e3v_n(ii,ij,jk)
            END DO
         END DO
         !
      END DO
      CALL mpp_sum( 'bdyvol', ztranst )   ! sum over the global domain

      ! Check the cumulated transport through unstructured OBC once barotropic velocities corrected
      ! ------------------------------------------------------
      IF( lwp .AND. MOD( kt, nwrite ) == 0 ) THEN
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
         IF(lwp) WRITE(numout,*)
         IF(lwp) WRITE(numout,*)'bdy_vol : time step :', kt
         IF(lwp) WRITE(numout,*)'~~~~~~~ '
         IF(lwp) WRITE(numout,*)'          cumulate flux EMP             =', z_cflxemp  , ' (m3/s)'
         IF(lwp) WRITE(numout,*)'          total lateral surface of OBC  =', bdysurftot, '(m2)'
         IF(lwp) WRITE(numout,*)'          correction velocity zubtpecor =', zubtpecor , '(m/s)'
         IF(lwp) WRITE(numout,*)'          cumulated transport ztranst   =', ztranst   , '(m3/s)'
      END IF
      !
      END IF ! ln_vol
      !
   END SUBROUTINE bdy_vol
------
------
------
------
------
------

------
------
```

## dev_r10448_bdyvol/src/OCE/BDY/bdyvol.F90

```fortran
+--  4 lines: MODULE bdyvol
   !!                 and filtered free surface are used
   !!===================================================================
   !! History :  1.0  !  2005-01  (J. Chanut, A. Sellar)  Original code
   !!             -   !  2006-01  (J. Chanut) Bug correction
   !!            3.0  !  2008-04  (NEMO team)  add in the reference version
   !!            3.4  !  2011     (D. Storkey) rewrite in preparation for OBC-BDY merge
   !!            4.0  !  2019-01  (P. Mathiot) adapted to time splitting
   !!----------------------------------------------------------------------
   USE oce            ! ocean dynamics and tracers
   USE bdy_oce        ! ocean open boundary conditions
   USE sbc_oce        ! ocean surface boundary conditions
   USE dom_oce        ! ocean space and time domain
   USE phycst         ! physical constants
+--  3 lines: USE sbcisf         ice shelf
   USE lib_mpp        ! for mppsum
   USE lib_fortran    ! Fortran routines library

   IMPLICIT NONE
   PRIVATE

   PUBLIC   bdy_vol2d    ! called by dynspg_ts

   !!----------------------------------------------------------------------
   !! NEMO/OCE 4.0 , NEMO Consortium (2018)
   !! $Id: ShelfSeasCodeReviewBdyVolBugFix.txt,v 1.2 2019/01/09 10:54:22 EndaODea Exp $
   !! Software governed by the CeCILL? license (see ./LICENSE)
   !!----------------------------------------------------------------------
CONTAINS

   SUBROUTINE bdy_vol2d( kt, kc, pua2d, pva2d, phu, phv )
      !!----------------------------------------------------------------------
      !!                      ***  ROUTINE bdyvol  ***
      !!
      !! ** Purpose :   This routine controls the volume of the system.
      !!      A correction velocity is calculated to correct the total transport
      !!      through the unstructured OBC.
------
------
      !!
      !! ** Method  :   The correction velocity (zubtpecor here) is defined calculating
      !!      the total transport through all open boundaries (trans_bdy) minus
      !!      the cumulate E-P flux (z_cflxemp) divided by the total lateral
      !!      surface (bdysurftot) of the unstructured boundary.
      !!         zubtpecor = [trans_bdy - z_cflxemp ]*(1./bdysurftot)
+-- 11 lines: !      with z_cflxemp => sum of (Evaporation minus Precipitation)
      !!         2/ The volume is constant even with E-P flux. In this case
      !!            the correction velocity must balance both the flux
      !!            through open boundaries and the ones through the free
      !!            surface.
      !!            (set nn_volctl to 1 in tne namelist for this option)
      !!----------------------------------------------------------------------
      INTEGER, INTENT(in) ::   kt, kc   ! ocean time-step index, cycle time-step
      !
      INTEGER  ::   ji, jj, jk, jb, jgrd
      INTEGER  ::   ib_bdy, ii, ij
      REAL(wp) ::   zubtpecor, ztranst
      REAL(wp), SAVE :: z_cflxemp
      REAL(wp), DIMENSION(jpi,jpj), INTENT(inout) :: pua2d, pva2d
      REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv
      TYPE(OBC_INDEX), POINTER :: idx
      !!-----------------------------------------------------------------------------
      !
------
------
------
------
------
------
------
------
      ! Calculate the cumulate surface Flux z_cflxemp (m3/s) over all the domain
      ! -----------------------------------------------------------------------
      IF ( kc == 1 ) z_cflxemp = glob_sum( 'bdyvol', ( emp(:,:) - rnf(:,:) + fwfisf(:,:) ) * bdytmask(:,:) * e1e2t(:,:)  ) / rau0

      ! Compute bdy surface each cycle if non linear free surface
      ! ---------------------------------------------------------
      IF ( .NOT. ln_linssh ) THEN
         ! compute area each time step
         bdysurftot = bdy_segs_surf( phu, phv )
      ELSE
         ! compute area only the first time
         IF ( ( kt == nit000 ) .AND. ( kc == 1 ) ) bdysurftot = bdy_segs_surf( phu, phv )
      END IF

      ! Transport through the unstructured open boundary
      ! ------------------------------------------------
      zubtpecor = 0._wp
      DO ib_bdy = 1, nb_bdy
         idx => idx_bdy(ib_bdy)
         !
         jgrd = 2                               ! cumulate u component contribution first
         DO jb = 1, idx%nblenrim(jgrd)
            ii = idx%nbi(jb,jgrd)
            ij = idx%nbj(jb,jgrd)
            zubtpecor = zubtpecor + idx%flagu(jb,jgrd) * pua2d(ii,ij) * e2u(ii,ij) * phu(ii,ij) * tmask_i(ii,ij) * tmask_i(ii+1,ij)
------
------
         END DO
         jgrd = 3                               ! then add v component contribution
         DO jb = 1, idx%nblenrim(jgrd)
            ii = idx%nbi(jb,jgrd)
            ij = idx%nbj(jb,jgrd)
            zubtpecor = zubtpecor + idx%flagv(jb,jgrd) * pva2d(ii,ij) * e1v(ii,ij) * phv(ii,ij) * tmask_i(ii,ij) * tmask_i(ii,ij+1)
------
------
         END DO
         !
      END DO
      IF( lk_mpp )   CALL mpp_sum( 'bdyvol', zubtpecor )   ! sum over the global domain

      ! The normal velocity correction
      ! ------------------------------
      IF( nn_volctl==1 ) THEN   ;   zubtpecor = ( zubtpecor - z_cflxemp ) / bdysurftot  ! maybe should be apply only once at the end
      ELSE                      ;   zubtpecor =   zubtpecor               / bdysurftot
      END IF

      ! Correction of the total velocity on the unstructured boundary to respect the mass flux conservation
      ! -------------------------------------------------------
------
      DO ib_bdy = 1, nb_bdy
         idx => idx_bdy(ib_bdy)
         !
         jgrd = 2                               ! correct u component
         DO jb = 1, idx%nblenrim(jgrd)
------
               ii = idx%nbi(jb,jgrd)
               ij = idx%nbj(jb,jgrd)
               pua2d(ii,ij) = pua2d(ii,ij) - idx%flagu(jb,jgrd) * zubtpecor * tmask_i(ii,ij) * tmask_i(ii+1,ij)
------
------
         END DO
         jgrd = 3                              ! correct v component
         DO jb = 1, idx%nblenrim(jgrd)
------
               ii = idx%nbi(jb,jgrd)
               ij = idx%nbj(jb,jgrd)
               pva2d(ii,ij) = pva2d(ii,ij) - idx%flagv(jb,jgrd) * zubtpecor * tmask_i(ii,ij) * tmask_i(ii,ij+1)
------
------
         END DO
         !
      END DO
      !
------
      ! Check the cumulated transport through unstructured OBC once barotropic velocities corrected
      ! ------------------------------------------------------
      IF( MOD( kt, nwrite ) == 0 .AND. ( kc == 1 ) ) THEN
         !
         ! compute residual transport across boundary
         ztranst = 0._wp
         DO ib_bdy = 1, nb_bdy
            idx => idx_bdy(ib_bdy)
            !
            jgrd = 2                               ! correct u component
            DO jb = 1, idx%nblenrim(jgrd)
                  ii = idx%nbi(jb,jgrd)
                  ij = idx%nbj(jb,jgrd)
                  ztranst = ztranst + idx%flagu(jb,jgrd) * pua2d(ii,ij) * e2u(ii,ij) * phu(ii,ij) * tmask_i(ii,ij) * tmask_i(ii+1,ij)
            END DO
            jgrd = 3                              ! correct v component
            DO jb = 1, idx%nblenrim(jgrd)
                  ii = idx%nbi(jb,jgrd)
                  ij = idx%nbj(jb,jgrd)
                  ztranst = ztranst + idx%flagv(jb,jgrd) * pva2d(ii,ij) * e1v(ii,ij) * phv(ii,ij) * tmask_i(ii,ij) * tmask_i(ii,ij+1)
            END DO
            !
         END DO
         IF( lk_mpp )   CALL mpp_sum('bdyvol', ztranst )   ! sum over the global domain


         IF(lwp) WRITE(numout,*)
         IF(lwp) WRITE(numout,*)'bdy_vol : time step :', kt
         IF(lwp) WRITE(numout,*)'~~~~~~~ '
         IF(lwp) WRITE(numout,*)'          cumulate flux EMP             =', z_cflxemp  , ' (m3/s)'
         IF(lwp) WRITE(numout,*)'          total lateral surface of OBC  =', bdysurftot, '(m2)'
         IF(lwp) WRITE(numout,*)'          correction velocity zubtpecor =', zubtpecor , '(m/s)'
         IF(lwp) WRITE(numout,*)'          cumulated transport ztranst   =', ztranst   , '(m3/s)'
      END IF
      !
   END SUBROUTINE bdy_vol2d
   !
   REAL(wp) FUNCTION bdy_segs_surf(phu, phv)
      !!----------------------------------------------------------------------
      !!                 ***  ROUTINE bdy_ctl_seg  ***
      !!
      !! ** Purpose :   Compute total lateral surface for volume correction
      !!
      !!----------------------------------------------------------------------

      REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points
      INTEGER            ::  igrd, ib_bdy, ib                ! loop indexes
```

```
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
------
   !!======================================================================
END MODULE bdyvol
```

```
      INTEGER , POINTER  ::  nbi, nbj                         ! short cuts
      REAL(wp), POINTER  ::  zflagu, zflagv                   !    -   -

      ! Compute total lateral surface for volume correction:
      ! ---------------------------------------------------
      bdy_segs_surf = 0._wp
      igrd = 2      ! Lateral surface at U-points
      DO ib_bdy = 1, nb_bdy
         DO ib = 1, idx_bdy(ib_bdy)%nblenrim(igrd)
            nbi => idx_bdy(ib_bdy)%nbi(ib,igrd)
            nbj => idx_bdy(ib_bdy)%nbj(ib,igrd)
            zflagu => idx_bdy(ib_bdy)%flagu(ib,igrd)
            bdy_segs_surf = bdy_segs_surf + phu(nbi, nbj)                             &
               &                             * e2u(nbi, nbj) * ABS( zflagu )          &
               &                             * tmask_i(nbi, nbj) * tmask_i(nbi+1, nbj)
         END DO
      END DO

      igrd=3 ! Add lateral surface at V-points
      DO ib_bdy = 1, nb_bdy
         DO ib = 1, idx_bdy(ib_bdy)%nblenrim(igrd)
            nbi => idx_bdy(ib_bdy)%nbi(ib,igrd)
            nbj => idx_bdy(ib_bdy)%nbj(ib,igrd)
            zflagv => idx_bdy(ib_bdy)%flagv(ib,igrd)
            bdy_segs_surf = bdy_segs_surf + phv(nbi, nbj)                             &
               &                             * e1v(nbi, nbj) * ABS( zflagv )          &
               &                             * tmask_i(nbi, nbj) * tmask_i(nbi, nbj+1)
         END DO
      END DO
      !
      ! redirect the time to bdyvol as this variable is only used by bdyvol
      IF( lk_mpp )   CALL mpp_sum( 'bdyvol', bdy_segs_surf ) ! sum over the global domain
      !
   END FUNCTION bdy_segs_surf
   !!======================================================================
END MODULE bdyvol
```

- Changes **bdy_vol** to **bdy_vol2d**
- It now takes new arguments:
    - barotorpic subcycle jn->kc
    - after barot vel ua,va -> pua2d, pva2d
    - ocean depth at u,v : zhup_e,zhvp_e - > phu,phv
- Remove comment that current depth is constant as it is not now
- Change z_cflxemp so it is storable (add SAVE)
- at kc=1, set emp flux = globsl sum of (emp,rnf,fwfisf) * the bdymask * e1e2 /rau0
- calculate bdysurf each time step if non linear, calls bdy_segs_surf(depth at u,v)
    - otherwise only do at first time step
- Calculate transport through open bdy , convert to 2D (remove k loop etc) * zubtpecor = sum over bdy points, firectionl fla g(in/out) u baro * e2u * depthu * tmask(ii)*tmask(ii+1) ! Thsi to remove possibel of land at point one inside even though here isd open bdy?
    - global sum
- actual correction , split:
    - nn_volctl==1, then re,pve emp / bdysurftot
    - otherwise dont just / bdysurftot
- correct u2d, v2d
    - u2d = u2d + directional flag *zubtpecor * mask
    - v similar
- Check cumulated transport , only do for kc=1
    - sums up transt over bdy points
    - then global sum
    - then write out for user

## FUNCTION bdy_segs_surf(phu,phv)

moved this calculation to this new function to it camn be called as required.

calculates (u/v) depth * cell width and sums both for u and v points.

## dynspg_ts.F90

adds call to IF( ln_bdy .AND. ln_vol ) CALL bdy_vol2d( kt, jn, ua_e, va_e, zhup2_e, zhvp2_e ) once ua,va and u v a depth available.

-- EndaODea - 08 Jan 2019