

Review bdyvol bug fix

```
Review bdyvol bug fix
BDYVol.f90
SUBROUTINE bdy_init
BDYVol.F90
FUNCTION bdy_segs_surfphu(phu)
dynsgo_ls.F90
```

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 dev_r10448_bdyvol/doc/latex/NEMO/subfiles/chap_IBC.tex and dev_r10448_bdyvol_CLEAN/doc/latex/NEMO/subfiles/chap_IBC.tex differ

```
Files dev_r10448_bdyvol/src/OC/BDYVol/bdyvol.F90 and dev_r10448_bdyvol_CLEAN/src/OC/BDYVol/bdyvol.F90 differ
Files dev_r10448_bdyvol/src/OC/BDYVol/bdyvol.F90 and dev_r10448_bdyvol_CLEAN/src/OC/BDYVol/bdyvol.F90 differ
Files dev_r10448_bdyvol/src/OC/BN/dynsgo_ls.F90 and dev_r10448_bdyvol_CLEAN/src/OC/BN/dynsgo_ls.F90 differ
```

BDY/bdyini.F90

SUBROUTINE bdy_init

removes flagu and flagv

Adds sanity check if tides are used:

```
49c40
< !! $Id: Shell Seas Code Review/bdyVolBugFix.txt,v 1.2 2019/01/09 10:54:22 EnolaDrea Exp $
> !! $Id: Shell Seas Code Review/bdyVolBugFix.txt,v 1.2 2019/01/09 10:54:22 EnolaDrea Exp $
134a135
> REAL(wp), POINTER :: flagu, flagv ! - -
> 386,394d350
< !
< ! 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_ to FALSE. '
<   IF( lwp ) WRITE( numout,*) ' If you want to use tides, '
<   IF( lwp ) WRITE( numout,*) ' equilibrate your bdy input files '
<   CALL ctrl_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
> ! before the call to bdyinit
> ! If( ln_volt ) THEN
>   lgn = .true. ! Lateral surface at U-points
>   DO ib_bdy = 1, nb_bdy
>     DO ib = 1, idb_bdy(ib_bdy)/nbklen(ib,lgrid)
>       nbj = idb_bdy(ib_bdy)/nbklen(ib,lgrid)
>       nbj1 = idb_bdy(ib_bdy)/nbklen(ib,lgrid)
>       flagu => idb_bdy(ib_bdy)/lagn(ib,lgrid)
>       bdsurftot = bdsurftot + bdsurf(ib,nbj) ! 6
>       & + tmask_ib(nbj, nbj) * ABS( flagu ) &
>       & + tmask_ib(nbj, nbj1) * ABS( flagu ) &
>       & + tmask_ib(nbj1, nbj)
>     END DO
>   END DO
>   lgn = .false. ! Add lateral surface at V-points
>   DO ib_bdy = 1, nb_bdy
>     DO ib = 1, idb_bdy(ib_bdy)/nbklen(ib,lgrid)
>       nbj = idb_bdy(ib_bdy)/nbklen(ib,lgrid)
>       nbj1 = idb_bdy(ib_bdy)/nbklen(ib,lgrid)
>       flagv => idb_bdy(ib_bdy)/lagn(ib,lgrid)
>       bdsurftot = bdsurftot + bdsurf(ib,nbj) ! 6
>       & + elv( nbj, nbj ) * ABS( flagv ) &
>       & + tmask_ib(nbj, nbj) * ABS( flagv ) &
>       & + tmask_ib(nbj, nbj1)
>     END DO
>   END DO
>   !
>   ! CALL npp_sum( 'bdyini', bdsurftot ) ! sum over the global domain
> END IF
1249a1277
>
```

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

BDY/bdyvol.F90

<pre> dev_r10448_bdyvol_CLEAN/src/OCE/BDY/bdyvol.F90 1^{**} ! 4 lines: MODULE bdyvol 2^{!!} ! and filtered free surface are used 3^{!!} ! History : 1.0 ! 2005-01 (J. Chanut, A. Sellar) Original code 4^{!!} ! 3.0 ! 2006-01 (J. Chanut) Bug correction 5^{!!} ! 3.4 ! 2011 (D. Storkey) rewrite in preparation for OBC-BDY merge 6^{!!} ! 3.4 ! 2011 (D. Storkey) rewrite in preparation for OBC-BDY merge 7^{!!} ! 3.4 ! 2011 (D. Storkey) rewrite in preparation for OBC-BDY merge 8^{!!} ! 3.4 ! 2011 (P. Matioc) adapted to time splitting 9^{!!} ! 3.4 ! 2011 (D. Storkey) rewrite in preparation for OBC-BDY merge 10^{!!} ! 3.4 ! 2011 (D. Storkey) rewrite in preparation for OBC-BDY merge 11^{!!} ! 3.4 ! 2011 (D. Storkey) rewrite in preparation for OBC-BDY merge 12^{!!} ! 3.4 ! 2011 (D. Storkey) rewrite in preparation for OBC-BDY merge 13 USE occ ! ocean dynamics and tracers 14 USE bdy_oce ! ocean open boundary conditions 15 USE doc_oce ! ocean closed boundary conditions 16 USE don_oce ! ocean space and time domain 17 USE physc ! physical constants 18 USE lib_mpp ! for nppsum 19 USE lib_fortran ! Fortran routines library 20 21 IMPLICIT NONE 22 PRIVATE 23 24 PUBLIC bdy_vol ! called by ??? 25 26 CONTAINS 27 28 SUBROUTINE bdy_vol(kt) 29 30 !NEMO/OCE 4.0 , NEMO Consortium (2018) 31 !! \$Id: ShallowCodeReview/bdyVolFix.txt,v 1.2 2019/01/09 10:54:22 Endo0 Exp \$ 32 !! Software governed by the CeCILL license (see ./LICENCE) 33 !! ----- 34 35 36 !ROUTINE bdyvol 37 38 !< 39 !> ** Purpose : This routine controls the volume of the system. 40 !> A correction velocity is calculated to correct the total transport 41 !> through the unstructured OBC. 42 !> The total depth used is constant (H0) to be consistent with the 43 !> linear free surface coded in OPA 8.3. (i.e. lntf = 1773 true ???) 44 !> 45 !> ** Method : The correction velocity (zubtpecor here) is defined calculating 46 !> the total transport through all open boundaries (trans_bdy) minus 47 !> the cumulate E-P flux (z_cflemp) divided by the total lateral 48 !> surface (bdysurftot) of the unstructured boundary. 49 !> zubtpecor = (trans_bdy - z_cflemp) / bdysurftot 50 !> z_cflemp = z_cflemp - sum of (Evaporation minus Precipitation) 51 !> The volume is constant even with E-P flux. In this case 52 !> the correction velocity is calculated by subtracting the 53 !> through open boundaries and the ones through the free 54 !> surface. 55 !> (note: volct1 to 1 in the namelist for this option) 56 !> 57 INTEGER, INTENT(in) :: kt ! ocean time-step index 58 59 INTEGER :: ji, jj, jk, jb, jgrd 60 INTEGER :: ib_bdy, ii, ij 61 REAL(wp) :: zubtpecor, z_cflemp, ztransit 62 63 TYPE(OBC_INDEX), POINTER :: idx 64 65 IF(ln_vol1) THEN 66 67 IF(kt == nn1000) THEN 68 !IF(lwp) WRITE(nnout,*)'bdy_vol : Correction of velocities along unstructured OBC' 69 !IF(lwp) WRITE(nnout,*)'-----' 70 END IF 71 72 ! Calculate the cumulate surface Flux z_cflemp (m/s) over all the domain 73 !Ign replace these lines 74 !CALL mpp_sum('emp(:, :) - rnf(:, :) + bwfisf(:, :) * bdymask(:, :) * ele2t(:, :)) / rau0 75 !CAL mpp_sum('bdyvol', 'z_cflemp') ! sum over the global domain 76 !by : 77 ! z_cflemp = glob_sum('bdyvol', (emp(:, :) - rnf(:, :) + bwfisf(:, :)) * bdymask(:, :) * ele2t(:, :)) / rau0 78 !Igno 79 !----- 80 81 ! Transport through the unstructured open boundary 82 !----- 83 !zubtpecor = 0._wp 84 DO ib_bdy = 1, nb_bdy 85 idb_idx_bdy(ib_bdy) 86 87 jgrd = 2 88 DO jb = 1, idxbnlenim(jgrd) 89 DO jk = 1, jkobj 90 ii = idxbnbi(jb,jgrd) 91 jj = idxbnbi(jk,jgrd) 92 ij = idxbnbi(jb,jk,jgrd) 93 zubtpecor = zubtpecor + idxvflagu(jb,jgrd) * ua(ii,ij,jk) * e2u(ii,ij,jk) * e3u_n(ii,ij,jk) 94 95 END DO 96 jgrd = 3 97 DO jb = 1, idxbnlenim(jgrd) 98 DO jk = 1, jkobj 99 ii = idxbnbi(jb,jgrd) 100 jj = idxbnbi(jk,jgrd) 101 ij = idxbnbi(jb,jk,jgrd) 102 zubtpecor = zubtpecor + idxvflagu(jb,jgrd) * ua(ii,ij,jk) * elv(ii,ij,jk) * ev_n(ii,ij,jk) 103 104 END DO 105 jgrd = 3 106 DO jb = 1, idxbnlenim(jgrd) 107 DO jk = 1, jkobj 108 ii = idxbnbi(jb,jgrd) 109 jj = idxbnbi(jk,jgrd) 110 val(ii,ij,jk) = ua(ii,ij,jk) - idxvflagu(jb,jgrd) * zubtpecor * umask(ii,ij,jk) 111 ztransit = ztransit - idxvflagu(jb,jgrd) * us(ii,ij,jk) * e2u(ii,ij,jk) * e3u_n(ii,ij,jk) 112 113 END DO 114 115 END DO 116 117 ! The normal velocity correction 118 !----- 119 !IF(nn_voltcl=1) THEN ; zubtpecor = (zubtpecor - z_cflemp) / bdysurftot 120 !ELSE ; zubtpecor = zubtpecor / bdysurftot 121 !END IF 122 123 ! Correction of the total velocity on the unstructured boundary to respect the mass flux conservation 124 !----- 125 !DO ib_bdy = 1, nb_bdy 126 ! idb_idx_bdy(ib_bdy) 127 ! ! 128 ! jgrd = 2 129 ! DO jb = 1, idxbnlenim(jgrd) 130 ! DO jk = 1, jkobj 131 ! ii = idxbnbi(jb,jgrd) 132 ! jj = idxbnbi(jk,jgrd) 133 ! val(ii,ij,jk) = ua(ii,ij,jk) - idxvflagu(jb,jgrd) * zubtpecor * umask(ii,ij,jk) 134 ! ztransit = ztransit - idxvflagu(jb,jgrd) * us(ii,ij,jk) * elv(ii,ij,jk) * ev_n(ii,ij,jk) 135 ! 136 ! END DO 137 ! END DO 138 ! jgrd = 3 139 ! DO jb = 1, idxbnlenim(jgrd) 140 ! DO jk = 1, jkobj 141 ! ii = idxbnbi(jb,jgrd) 142 ! jj = idxbnbi(jk,jgrd) 143 ! val(ii,ij,jk) = ua(ii,ij,jk) - idxvflagu(jb,jgrd) * zubtpecor * umask(ii,ij,jk) 144 ! ztransit = ztransit - idxvflagu(jb,jgrd) * us(ii,ij,jk) * elv(ii,ij,jk) * ev_n(ii,ij,jk) 145 ! 146 ! END DO 147 ! END DO 148 ! ! 149 !END DO 150 !CALL mpp_sum('bdyvol', ztransit) ! sum over the global domain 151 152 ! Check the cumulated transport through unstructured OBC once barotropic velocities corrected 153 !----- 154 IF(lwp .AND. MOD(kt, nnwrite) == 0) THEN 155 156 !IF(lwp) WRITE(nnout,*)'bdy_vol : time step ', kt 157 !IF(lwp) WRITE(nnout,*)'-----' 158 !IF(lwp) WRITE(nnout,*)'cumulate flux BPP', 'z_cflemp', '(m/s)' 159 !IF(lwp) WRITE(nnout,*)'total lateral surface of OBC', 'bdysurftot', '(m2)' 160 !IF(lwp) WRITE(nnout,*)'correction velocity zubtpecor', 'zubtpecor', '(m/s)' 161 !IF(lwp) WRITE(nnout,*)'cumulated transport ztransit', 'ztransit', '(m/s)' 162 163 END IF 164 165 END IF ! ln_vol 166 167 END SUBROUTINE bdy_vol 168 169 200 !< 201 !> ** Purpose : Compute total lateral surface for volume correction 202 !> 203 !> ----- 204 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 205 !> INTEGER :: i, j, idb_idx_bdy, ib 206 !> ! loop indexes 207 208 209 !< 210 !> ** Purpose : Compute total lateral surface for volume correction 211 !> 212 !> ----- 213 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 214 !> INTEGER :: i, j, idb_idx_bdy, ib 215 !> ! loop indexes 216 217 218 !< 219 !> ** Purpose : Compute total lateral surface for volume correction 220 !> 221 !> ----- 222 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 223 !> INTEGER :: i, j, idb_idx_bdy, ib 224 !> ! loop indexes 225 226 227 !< 228 !> ** Purpose : Compute total lateral surface for volume correction 229 !> 230 !> ----- 231 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 232 !> INTEGER :: i, j, idb_idx_bdy, ib 233 !> ! loop indexes 234 235 236 !< 237 !> ** Purpose : Compute total lateral surface for volume correction 238 !> 239 !> ----- 240 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 241 !> INTEGER :: i, j, idb_idx_bdy, ib 242 !> ! loop indexes 243 244 245 !< 246 !> ** Purpose : Compute total lateral surface for volume correction 247 !> 248 !> ----- 249 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 250 !> INTEGER :: i, j, idb_idx_bdy, ib 251 !> ! loop indexes 252 253 254 !< 255 !> ** Purpose : Compute total lateral surface for volume correction 256 !> 257 !> ----- 258 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 259 !> INTEGER :: i, j, idb_idx_bdy, ib 260 !> ! loop indexes 261 262 263 !< 264 !> ** Purpose : Compute total lateral surface for volume correction 265 !> 266 !> ----- 267 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 268 !> INTEGER :: i, j, idb_idx_bdy, ib 269 !> ! loop indexes 270 271 272 !< 273 !> ** Purpose : Compute total lateral surface for volume correction 274 !> 275 !> ----- 276 !> REAL(wp), FUNCTION bdy_segs_surf(phu, phv) 277 !> !----- 278 !> *** ROUTINE bdy_ctt_seg *** 279 !> 280 !> !< 281 !> !> ** Purpose : Compute total lateral surface for volume correction 282 !> 283 !> ----- 284 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 285 !> INTEGER :: i, j, idb_idx_bdy, ib 286 !> ! loop indexes 287 288 289 !< 290 !> ** Purpose : Compute total lateral surface for volume correction 291 !> 292 !> ----- 293 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 294 !> INTEGER :: i, j, idb_idx_bdy, ib 295 !> ! loop indexes 296 297 298 !< 299 !> ** Purpose : Compute total lateral surface for volume correction 300 !> 301 !> ----- 302 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 303 !> INTEGER :: i, j, idb_idx_bdy, ib 304 !> ! loop indexes 305 306 307 !< 308 !> ** Purpose : Compute total lateral surface for volume correction 309 !> 310 !> ----- 311 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 312 !> INTEGER :: i, j, idb_idx_bdy, ib 313 !> ! loop indexes 314 315 316 !< 317 !> ** Purpose : Compute total lateral surface for volume correction 318 !> 319 !> ----- 320 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 321 !> INTEGER :: i, j, idb_idx_bdy, ib 322 !> ! loop indexes 323 324 325 !< 326 !> ** Purpose : Compute total lateral surface for volume correction 327 !> 328 !> ----- 329 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 330 !> INTEGER :: i, j, idb_idx_bdy, ib 331 !> ! loop indexes 332 333 334 !< 335 !> ** Purpose : Compute total lateral surface for volume correction 336 !> 337 !> ----- 338 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 339 !> INTEGER :: i, j, idb_idx_bdy, ib 340 !> ! loop indexes 341 342 343 !< 344 !> ** Purpose : Compute total lateral surface for volume correction 345 !> 346 !> ----- 347 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 348 !> INTEGER :: i, j, idb_idx_bdy, ib 349 !> ! loop indexes 350 351 352 !< 353 !> ** Purpose : Compute total lateral surface for volume correction 354 !> 355 !> ----- 356 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 357 !> INTEGER :: i, j, idb_idx_bdy, ib 358 !> ! loop indexes 359 360 361 !< 362 !> ** Purpose : Compute total lateral surface for volume correction 363 !> 364 !> ----- 365 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 366 !> INTEGER :: i, j, idb_idx_bdy, ib 367 !> ! loop indexes 368 369 370 !< 371 !> ** Purpose : Compute total lateral surface for volume correction 372 !> 373 !> ----- 374 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 375 !> INTEGER :: i, j, idb_idx_bdy, ib 376 !> ! loop indexes 377 378 379 !< 380 !> ** Purpose : Compute total lateral surface for volume correction 381 !> 382 !> ----- 383 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 384 !> INTEGER :: i, j, idb_idx_bdy, ib 385 !> ! loop indexes 386 387 388 !< 389 !> ** Purpose : Compute total lateral surface for volume correction 390 !> 391 !> ----- 392 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 393 !> INTEGER :: i, j, idb_idx_bdy, ib 394 !> ! loop indexes 395 396 397 !< 398 !> ** Purpose : Compute total lateral surface for volume correction 399 !> 400 !> ----- 401 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 402 !> INTEGER :: i, j, idb_idx_bdy, ib 403 !> ! loop indexes 404 405 406 !< 407 !> ** Purpose : Compute total lateral surface for volume correction 408 !> 409 !> ----- 410 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 411 !> INTEGER :: i, j, idb_idx_bdy, ib 412 !> ! loop indexes 413 414 415 !< 416 !> ** Purpose : Compute total lateral surface for volume correction 417 !> 418 !> ----- 419 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 420 !> INTEGER :: i, j, idb_idx_bdy, ib 421 !> ! loop indexes 422 423 424 !< 425 !> ** Purpose : Compute total lateral surface for volume correction 426 !> 427 !> ----- 428 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 429 !> INTEGER :: i, j, idb_idx_bdy, ib 430 !> ! loop indexes 431 432 433 !< 434 !> ** Purpose : Compute total lateral surface for volume correction 435 !> 436 !> ----- 437 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 438 !> INTEGER :: i, j, idb_idx_bdy, ib 439 !> ! loop indexes 440 441 442 !< 443 !> ** Purpose : Compute total lateral surface for volume correction 444 !> 445 !> ----- 446 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 447 !> INTEGER :: i, j, idb_idx_bdy, ib 448 !> ! loop indexes 449 450 451 !< 452 !> ** Purpose : Compute total lateral surface for volume correction 453 !> 454 !> ----- 455 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 456 !> INTEGER :: i, j, idb_idx_bdy, ib 457 !> ! loop indexes 458 459 460 !< 461 !> ** Purpose : Compute total lateral surface for volume correction 462 !> 463 !> ----- 464 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 465 !> INTEGER :: i, j, idb_idx_bdy, ib 466 !> ! loop indexes 467 468 469 !< 470 !> ** Purpose : Compute total lateral surface for volume correction 471 !> 472 !> ----- 473 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 474 !> INTEGER :: i, j, idb_idx_bdy, ib 475 !> ! loop indexes 476 477 478 !< 479 !> ** Purpose : Compute total lateral surface for volume correction 480 !> 481 !> ----- 482 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 483 !> INTEGER :: i, j, idb_idx_bdy, ib 484 !> ! loop indexes 485 486 487 !< 488 !> ** Purpose : Compute total lateral surface for volume correction 489 !> 490 !> ----- 491 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 492 !> INTEGER :: i, j, idb_idx_bdy, ib 493 !> ! loop indexes 494 495 496 !< 497 !> ** Purpose : Compute total lateral surface for volume correction 498 !> 499 !> ----- 500 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 501 !> INTEGER :: i, j, idb_idx_bdy, ib 502 !> ! loop indexes 503 504 505 !< 506 !> ** Purpose : Compute total lateral surface for volume correction 507 !> 508 !> ----- 509 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 510 !> INTEGER :: i, j, idb_idx_bdy, ib 511 !> ! loop indexes 512 513 514 !< 515 !> ** Purpose : Compute total lateral surface for volume correction 516 !> 517 !> ----- 518 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 519 !> INTEGER :: i, j, idb_idx_bdy, ib 520 !> ! loop indexes 521 522 523 !< 524 !> ** Purpose : Compute total lateral surface for volume correction 525 !> 526 !> ----- 527 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 528 !> INTEGER :: i, j, idb_idx_bdy, ib 529 !> ! loop indexes 530 531 532 !< 533 !> ** Purpose : Compute total lateral surface for volume correction 534 !> 535 !> ----- 536 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 537 !> INTEGER :: i, j, idb_idx_bdy, ib 538 !> ! loop indexes 539 540 541 !< 542 !> ** Purpose : Compute total lateral surface for volume correction 543 !> 544 !> ----- 545 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 546 !> INTEGER :: i, j, idb_idx_bdy, ib 547 !> ! loop indexes 548 549 550 !< 551 !> ** Purpose : Compute total lateral surface for volume correction 552 !> 553 !> ----- 554 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 555 !> INTEGER :: i, j, idb_idx_bdy, ib 556 !> ! loop indexes 557 558 559 !< 560 !> ** Purpose : Compute total lateral surface for volume correction 561 !> 562 !> ----- 563 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 564 !> INTEGER :: i, j, idb_idx_bdy, ib 565 !> ! loop indexes 566 567 568 !< 569 !> ** Purpose : Compute total lateral surface for volume correction 570 !> 571 !> ----- 572 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 573 !> INTEGER :: i, j, idb_idx_bdy, ib 574 !> ! loop indexes 575 576 577 !< 578 !> ** Purpose : Compute total lateral surface for volume correction 579 !> 580 !> ----- 581 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 582 !> INTEGER :: i, j, idb_idx_bdy, ib 583 !> ! loop indexes 584 585 586 !< 587 !> ** Purpose : Compute total lateral surface for volume correction 588 !> 589 !> ----- 590 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 591 !> INTEGER :: i, j, idb_idx_bdy, ib 592 !> ! loop indexes 593 594 595 !< 596 !> ** Purpose : Compute total lateral surface for volume correction 597 !> 598 !> ----- 599 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 600 !> INTEGER :: i, j, idb_idx_bdy, ib 601 !> ! loop indexes 602 603 604 !< 605 !> ** Purpose : Compute total lateral surface for volume correction 606 !> 607 !> ----- 608 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 609 !> INTEGER :: i, j, idb_idx_bdy, ib 610 !> ! loop indexes 611 612 613 !< 614 !> ** Purpose : Compute total lateral surface for volume correction 615 !> 616 !> ----- 617 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 618 !> INTEGER :: i, j, idb_idx_bdy, ib 619 !> ! loop indexes 620 621 622 !< 623 !> ** Purpose : Compute total lateral surface for volume correction 624 !> 625 !> ----- 626 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 627 !> INTEGER :: i, j, idb_idx_bdy, ib 628 !> ! loop indexes 629 630 631 !< 632 !> ** Purpose : Compute total lateral surface for volume correction 633 !> 634 !> ----- 635 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 636 !> INTEGER :: i, j, idb_idx_bdy, ib 637 !> ! loop indexes 638 639 640 !< 641 !> ** Purpose : Compute total lateral surface for volume correction 642 !> 643 !> ----- 644 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 645 !> INTEGER :: i, j, idb_idx_bdy, ib 646 !> ! loop indexes 647 648 649 !< 650 !> ** Purpose : Compute total lateral surface for volume correction 651 !> 652 !> ----- 653 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 654 !> INTEGER :: i, j, idb_idx_bdy, ib 655 !> ! loop indexes 656 657 658 !< 659 !> ** Purpose : Compute total lateral surface for volume correction 660 !> 661 !> ----- 662 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 663 !> INTEGER :: i, j, idb_idx_bdy, ib 664 !> ! loop indexes 665 666 667 !< 668 !> ** Purpose : Compute total lateral surface for volume correction 669 !> 670 !> ----- 671 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 672 !> INTEGER :: i, j, idb_idx_bdy, ib 673 !> ! loop indexes 674 675 676 !< 677 !> ** Purpose : Compute total lateral surface for volume correction 678 !> 679 !> ----- 680 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 681 !> INTEGER :: i, j, idb_idx_bdy, ib 682 !> ! loop indexes 683 684 685 !< 686 !> ** Purpose : Compute total lateral surface for volume correction 687 !> 688 !> ----- 689 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 690 !> INTEGER :: i, j, idb_idx_bdy, ib 691 !> ! loop indexes 692 693 694 !< 695 !> ** Purpose : Compute total lateral surface for volume correction 696 !> 697 !> ----- 698 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 699 !> INTEGER :: i, j, idb_idx_bdy, ib 700 !> ! loop indexes 701 702 703 !< 704 !> ** Purpose : Compute total lateral surface for volume correction 705 !> 706 !> ----- 707 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 708 !> INTEGER :: i, j, idb_idx_bdy, ib 709 !> ! loop indexes 710 711 712 !< 713 !> ** Purpose : Compute total lateral surface for volume correction 714 !> 715 !> ----- 716 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 717 !> INTEGER :: i, j, idb_idx_bdy, ib 718 !> ! loop indexes 719 720 721 !< 722 !> ** Purpose : Compute total lateral surface for volume correction 723 !> 724 !> ----- 725 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 726 !> INTEGER :: i, j, idb_idx_bdy, ib 727 !> ! loop indexes 728 729 730 !< 731 !> ** Purpose : Compute total lateral surface for volume correction 732 !> 733 !> ----- 734 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 735 !> INTEGER :: i, j, idb_idx_bdy, ib 736 !> ! loop indexes 737 738 739 !< 740 !> ** Purpose : Compute total lateral surface for volume correction 741 !> 742 !> ----- 743 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 744 !> INTEGER :: i, j, idb_idx_bdy, ib 745 !> ! loop indexes 746 747 748 !< 749 !> ** Purpose : Compute total lateral surface for volume correction 750 !> 751 !> ----- 752 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 753 !> INTEGER :: i, j, idb_idx_bdy, ib 754 !> ! loop indexes 755 756 757 !< 758 !> ** Purpose : Compute total lateral surface for volume correction 759 !> 760 !> ----- 761 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 762 !> INTEGER :: i, j, idb_idx_bdy, ib 763 !> ! loop indexes 764 765 766 !< 767 !> ** Purpose : Compute total lateral surface for volume correction 768 !> 769 !> ----- 770 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 771 !> INTEGER :: i, j, idb_idx_bdy, ib 772 !> ! loop indexes 773 774 775 !< 776 !> ** Purpose : Compute total lateral surface for volume correction 777 !> 778 !> ----- 779 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 780 !> INTEGER :: i, j, idb_idx_bdy, ib 781 !> ! loop indexes 782 783 784 !< 785 !> ** Purpose : Compute total lateral surface for volume correction 786 !> 787 !> ----- 788 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 789 !> INTEGER :: i, j, idb_idx_bdy, ib 790 !> ! loop indexes 791 792 793 !< 794 !> ** Purpose : Compute total lateral surface for volume correction 795 !> 796 !> ----- 797 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 798 !> INTEGER :: i, j, idb_idx_bdy, ib 799 !> ! loop indexes 800 801 802 !< 803 !> ** Purpose : Compute total lateral surface for volume correction 804 !> 805 !> ----- 806 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 807 !> INTEGER :: i, j, idb_idx_bdy, ib 808 !> ! loop indexes 809 810 811 !< 812 !> ** Purpose : Compute total lateral surface for volume correction 813 !> 814 !> ----- 815 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 816 !> INTEGER :: i, j, idb_idx_bdy, ib 817 !> ! loop indexes 818 819 820 !< 821 !> ** Purpose : Compute total lateral surface for volume correction 822 !> 823 !> ----- 824 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 825 !> INTEGER :: i, j, idb_idx_bdy, ib 826 !> ! loop indexes 827 828 829 !< 830 !> ** Purpose : Compute total lateral surface for volume correction 831 !> 832 !> ----- 833 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 834 !> INTEGER :: i, j, idb_idx_bdy, ib 835 !> ! loop indexes 836 837 838 !< 839 !> ** Purpose : Compute total lateral surface for volume correction 840 !> 841 !> ----- 842 !> REAL(wp), DIMENSION(jpi,jpj), INTENT(in) :: phu, phv ! water column thickness at U and V points 843 !> INTEGER :: i, j, idb</pre>

```

86     INTEGER , POINTER :: nbi,nbj           ! short cuts
87     REAL(wp), POINTER :: zflagu,zflagv      ! - -
88
89     ! Compute total lateral surface for volume correction:
90
91     bdy_segs_surf = 0d0
92
93     !igrd = 2 ! Lateral surface at U-points
94     DO ib_bdy = 1,nb_bdy
95       DO iu_ib = 1,ib_bdy
96         DO iu_ib_bdy = 1,ib(ib_bdy)\nb(ib_ib)
97           rbi => idx_bdy(ib_bdy)\nb(ib_ib).igrd
98           rbi => idx_bdy(ib_bdy)\nb(ib_ib).igrd
99           rbi => idx_bdy(ib_bdy)\nb(ib_ib).igrd
100          rbi => idx_bdy(ib_bdy)\nb(ib_ib).igrd
101          bdy_segs_surf = bdy_segs_surf + phu(ib_ib, nbj)
102          bdy_segs_surf = bdy_segs_surf + e2u(nbi, nbj) * ABS(zflagv)
103          bdy_segs_surf = bdy_segs_surf + tmask_i(nbi-1, nbj)
104
105        END DO
106      END DO
107
108      !igrd=3 ! Add lateral surface at V-points
109      DO ib_bdy = 1,nb_bdy
110        DO iv_ib = 1,ib_bdy
111          DO iv_ib_bdy = 1,ib(ib_bdy)\nb(ib_ib)
112            rbi => idx_bdy(ib_bdy)\nb(ib_ib).igrd
113            rbi => idx_bdy(ib_bdy)\nb(ib_ib).igrd
114            rbi => idx_bdy(ib_bdy)\nb(ib_ib).igrd
115            rbi => idx_bdy(ib_bdy)\nb(ib_ib).igrd
116            rbi => idx_bdy(ib_bdy)\nb(ib_ib).igrd
117            bdy_segs_surf = bdy_segs_surf + phv(ib_ib, nbj)
118            bdy_segs_surf = bdy_segs_surf + elv(nbi, nbj) * ABS(zflagv)
119            bdy_segs_surf = bdy_segs_surf + tmask_i(nbi, nbj+1)
120
121          END DO
122        END DO
123
124      ! redirect the time to bdyvol as this variable is only used by bdyvol
125      IF lk_mpp ) CALL mpp_sum('bdyvol', bdy_segs_surf) ! sum over the global domain
126
127    END FUNCTION bdy_segs_surf
128
129 !-----+
130 END MODULE bdyvol

```

- Changes `bdy_vol` to `bdy_vol2d`
- It now takes new arguments:
 - `in_bdy` and `in_vol`
 - after band test `ib(ib_ib)\nb(ib_ib)` `pe2d`
 - ocean depth at `u,v,zhup,zhvp,zhrv,e>phu,phv`
- Remove comment that current depth is constant as it is not now
- Change `z_ofxmp` so it is storables (add `SAVE`)
- at `kt=1`, set emp flux = global sum of (`emp.mfMsf`) * the bdy mask * `e1e2/rau0`
- calculate bdyvol each time step if non linear, calls `bdy_segs_surf`/depth at `u,v`
 - otherwise only do at first time step
- Calculations through open bdy , convert to 2D (remove `k` loop etc) * `zubtpecor` = sum over bdy points, function `fia g(in/out) u baro * e2u * depth * tmask(i)*mask(j+1)` This to remove possibel of land at point one inside even though here isd open bdy
- global sum
- actual correction . split:
 - `nn_vald==1`, then `re_pve.emp / bdysegsurf`
 - otherwise dont just / bdysegsurf
- correct `u2d,v2d`
 - `u2d = u2d + directional flag * zubtpecor * mask`
 - `v2d = v2d + directional flag * zubtpecor * mask`
- Change 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 can 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(in_bdy .AND. in_vol) CALL bdy_vol2d(kt, jn, ua_e, va_e, zhup2_e, zhvp2_e)` once ua,va and u,v a depth available.

– EndoODea – 08 Jan 2019

This topic: Main > ShellSeasCodeReviewBdyVolBugFix

History: 2 - 09 Jan 2019 - 10:54:22 – EndoODea

Copyright © by the contributing authors. All material on this collaboration platform is the property of the contributing authors.

[Send feedback](#)