< Chronologique > < Discussions >

- **From**: Jozef Vivoda <jozef.vivoda@shmu.sk>
- **To**: alabobo2@meteo.fr
- **Subject**: [alabobo2] URGENT: serious bug in quadratic coupling - BUGFIX for cy40t1 and all later cycles
- **Date**: Wed, 15 Feb 2017 15:08:34 +0100

Dear colleagues,

There is an active hidden bug in dynamics of cy40t1.
It appears in LAM models of ARPEGE/IFS code running
with LQCPL=.TRUE. - quadratic time interpolation of coupling files.
The configurations with linear coupling are not affected.

Definition of problem:

The interpolation weights EWB computed in routine "arpifs/module
/elbc0b_mod.F90"
are not consistent with treatment of GMV fields that holds coupling fields.
This can be easily checked in routine "esc2r.F90coupling/external/gp
cou/esc2r.F90".
This influences all prognostic quantities.

The bug is hidden and is active every N-th time step (with
N=TEFRCL/TSTEP)
when condition (NSTEP+1) == N is satisfied.
In this particular time step the data from wrong coupling file is used !
(attached test.png shows the evolution of MSLP in one point close to LBC.
Red curve is cy40t1 without bugfix, blue curve is the result from CY38T1
and green curve is bugfixed solution in CY40T1)

Bug causes erroneous signal in all prognostic fields. This signal is not visible
immediately due to nature of bug. It propagates over whole domain. It spoils
fields globally
in sufficiently long time interval. The bug is present in NH dynamics as well.

(attached MSLPRESSURE_121.png - difference of MSLP between vy 40t1 and
38t1. There is
something close to boundaries, on MSLPRESSURE_0140.png is field 20 time
steps later,
we see oscillation in domain with magnitude +-5 hPa).

Fix:

I am sending you fixed routine "arpifs/module/elbc0b_mod.F90".

I did some small modification to improve readability of code (because
I guess that its limited readability was the source of bug.

At the same time, I fixed also the key LCCPL - cubic coupling interpolations).

Sincerely
Jozef Vivoda
on behalf of Slovakian NWP team

Attachment: test.png
Description: PNG image

Attachment: MSLPRESSURE_0140.png
Description: PNG image

Attachment: MSLPRESSURE_0121.png
Description: PNG image

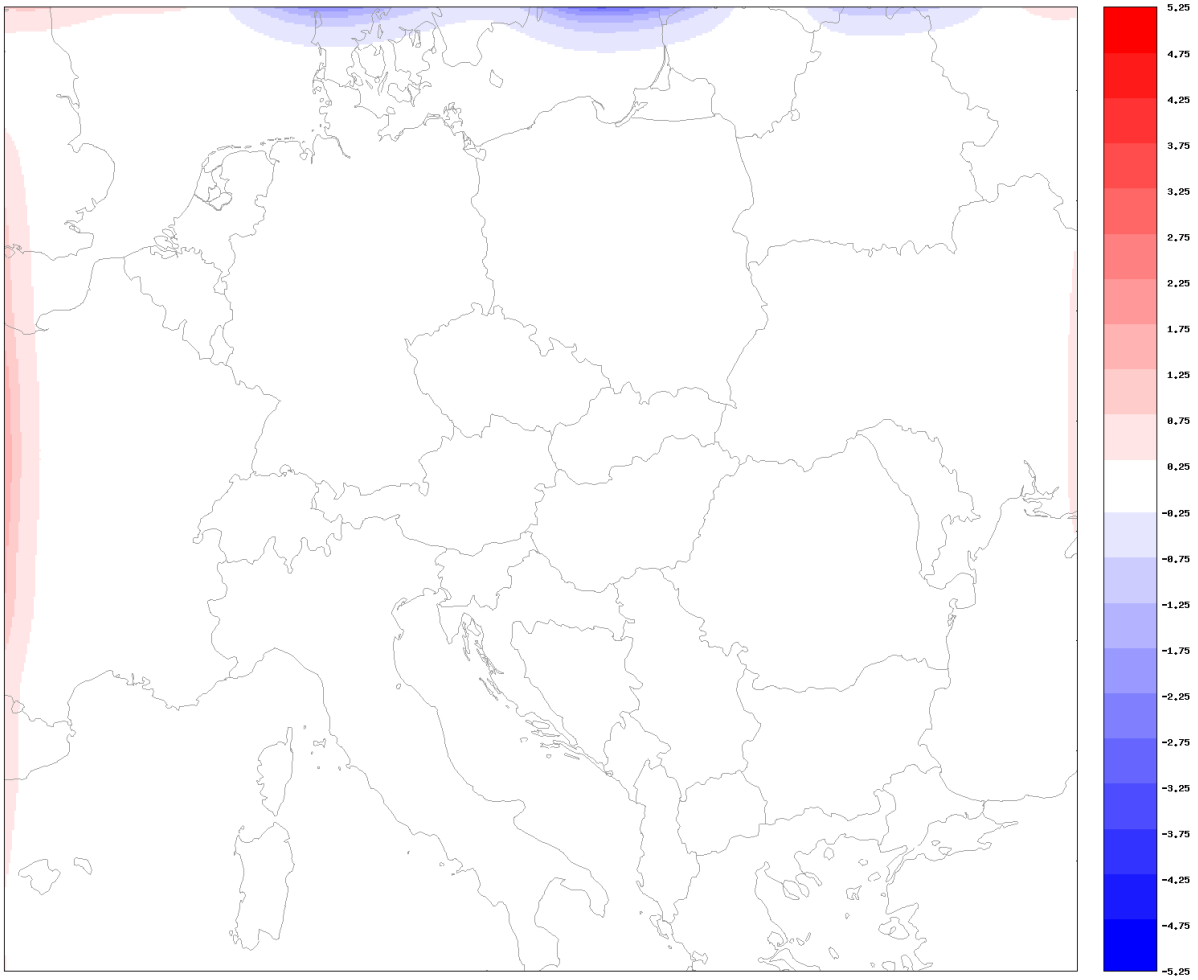
Attachment: elbc0b_mod.F90
Description: Binary data

- **[alabobo2] URGENT: serious bug in quadratic coupling - BUGFIX for cy40t1 and all later cycles**, *Jozef Vivoda, 15/02/2017*

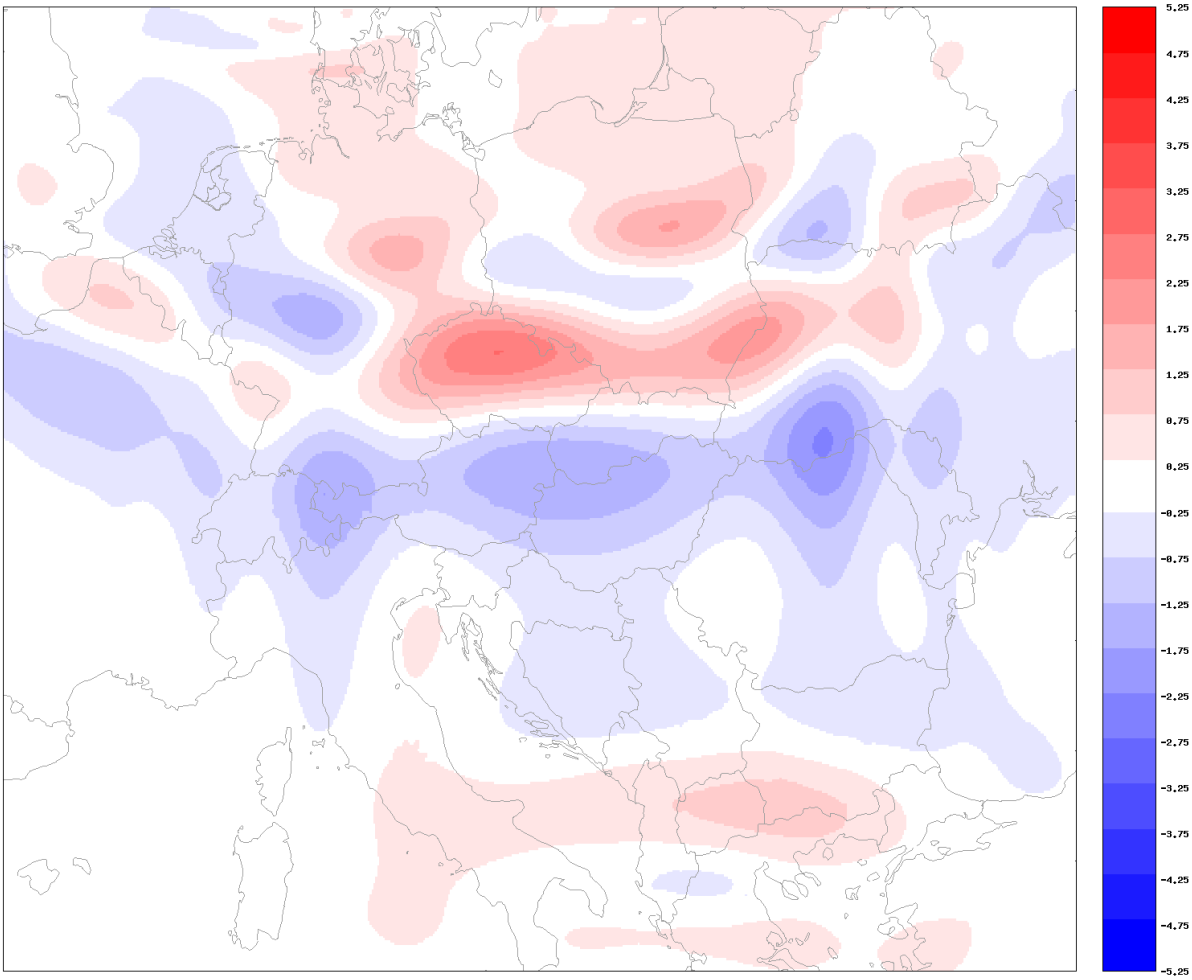
Archives gérées par MHonArc 2.6.16.

§
Powered by Sympa 5.4.7

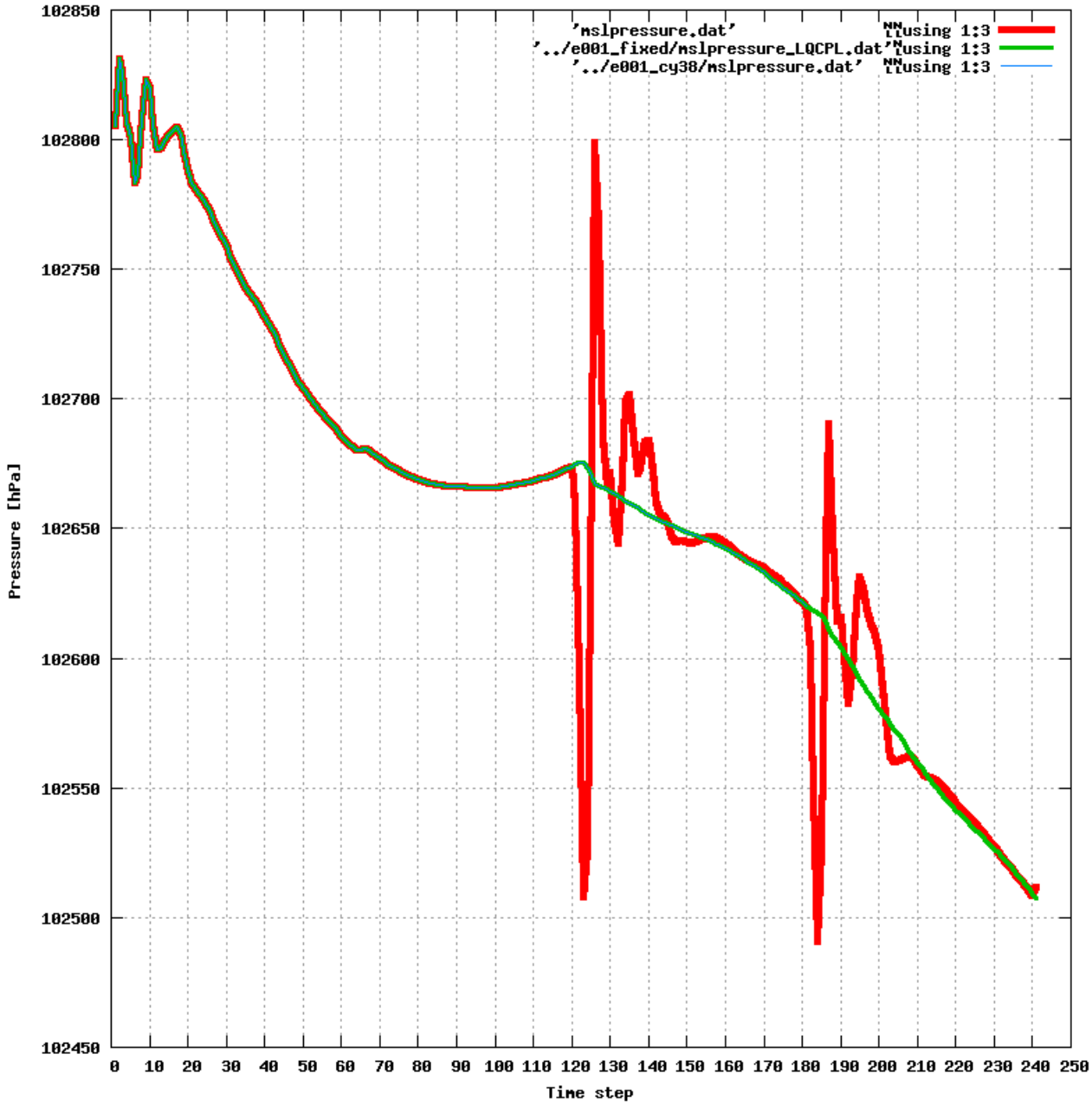
MSLPRESSURE-0121.grb



MSLPRESSURE-0140.grb



Test.png


'mslpressure.dat' using 1:3
'../e001_fixed/mslpressure_LQCPL.dat' using 1:3
'../e001_cy38/mslpressure.dat' using 1:3
Pressure [hPa]
102850
102800
102750
102700
102650
102600
102550
102500
102450
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250
Time step

fixed routine "arpifs/module/elbc0b_mod.F90"
MODULE ELBC0B_MOD

! Purpose :
! -------
!   Forcing a LAM model by another model: part 0B
!   - forcing by lateral boundary conditions
!   - pressure tendency coupling
!   - spectral nudging

! Interface :
! ---------
!   Empty.

! External :
! --------
!   None.

! Method :
! ------
!   See Documentation.

! Reference :
! ---------

! Author :
! ------
!   K. YESSAD (CNRM/GMAP) after YEMBICU, YEMDYN, YEMGT3B, SUEBICU, SUEDYN, SUESC2.

! Modifications :
! -------------
! Original : December 2010
! Daan Degrauwe: Feb 2012 Boyd biperiodization
! B. Bochenek (Oct 2013): Weights for LBC interpolation
!---

USE PARKIND1 , ONLY : JPIM, JPRB
USE YOMHOOK , ONLY : LHOOK, DR_HOOK
USE YEMDIM   , ONLY : NBZONG, NBZONL, NBIPINCIX, NBIPINCIY
USE YOMDFI   , ONLY : NSTDFI, NSTDFIA, RTDFI, RTDFIA
USE YOMDIM   , ONLY : NDGLG, NDGSAL, NDGENL, NDLON, NDGUNG, NDGUXG, NDLUNG, &
 & NDLUXG, NSPEC2, NPROMA, NGPBLKS
USE YOMDIMV , ONLY : NFLEVG, NFLEVL
USE YOMDIMF , ONLY : NFD2D, NS3D
USE YOMCT0   , ONLY : LNHDYN, LOUTPUT, NPRTRV, LRPLANE, LALLOPR, NCONF, NSTOP
USE YOMDYNA , ONLY : LNHX
USE YOMDYN   , ONLY : TSTEP
USE YOMGC   , ONLY : YRGSGEOM_NB
USE YOMGEM   , ONLY : NSTAGP, NGPTOT

```
USE YOMINI   , ONLY : LDFI
USE YOMLUN   , ONLY : NULOUT, NULNAM
USE YOMMP0   , ONLY : MYSETB, MYSETV
USE YOMMP    , ONLY : NSTA, NONL, MYLATS, NPTRFLOFF, MY_REGION_EW,
NBSETSP
USE YOMGMV   , ONLY : YT0, YT1
USE YOM_YGFL , ONLY : YGFL, YGFLC, YQ_NL, YQ
USE ELBC0A_MOD,ONLY : NBICOU, NBICOT, NBICOP, NBICPD, NBICVD, NBICNHX, &
& LTENC, LALLTC, LQCPL, LESPCPL, NEFRSPCPL, SPNUDSP, SPNUDQ, LSPTENC,
LCCPL

IMPLICIT NONE
SAVE

!
=========================================================================
======

!     1.   TYPE DEFINITION
!          ---------------

! Structure for GMVS coupled fields in LTENC option.
TYPE TGMVSTENC
INTEGER(KIND=JPIM) :: MSP       ! surface pressure variable
INTEGER(KIND=JPIM) :: MSPL      ! zonal component of grad(surface pressure variable)
INTEGER(KIND=JPIM) :: MSPM      ! meridian component of grad(surface pressure variable)
INTEGER(KIND=JPIM) :: NDIM      ! number of coupled fields (includes derivatives)
INTEGER(KIND=JPIM) :: NDIMT     ! number of temporally interpolated fields (includes
derivatives)
END TYPE TGMVSTENC

! Structure for GMV coupled fields
TYPE TGMVCPL
! coupled GMV
INTEGER(KIND=JPIM) :: MU        ! U-wind
INTEGER(KIND=JPIM) :: MV        ! V-wind
INTEGER(KIND=JPIM) :: MT        ! temperature
INTEGER(KIND=JPIM) :: MSPD      ! pressure departure variable
INTEGER(KIND=JPIM) :: MSVD      ! vertical divergence variable
INTEGER(KIND=JPIM) :: MNHX      ! NHX term
! derivatives required for linear terms calculation (in ESEIMPLS)
INTEGER(KIND=JPIM) :: MDIV      ! horizontal divergence
INTEGER(KIND=JPIM) :: MTL       ! zonal component of grad(temperature)
INTEGER(KIND=JPIM) :: MTM       ! meridian component of grad(temperature)
INTEGER(KIND=JPIM) :: MSPDL     ! zonal component of grad(pressure departure variable)
INTEGER(KIND=JPIM) :: MSPDM     ! meridian component of grad(pressure departure variable)
INTEGER(KIND=JPIM) :: NDIM      ! number of coupled fields (does not include derivatives)
INTEGER(KIND=JPIM) :: NDIMT     ! number of temporally interpolated fields (includes
derivatives)
END TYPE TGMVCPL

! Structure for GMVS coupled fields
```

```
TYPE TGMVSCPL
INTEGER(KIND=JPIM) :: MSP       ! surface pressure variable
INTEGER(KIND=JPIM) :: MSPL      ! zonal component of grad(surface pressure variable)
INTEGER(KIND=JPIM) :: MSPM      ! meridian component of grad(surface pressure variable)
INTEGER(KIND=JPIM) :: NDIM      ! number of coupled fields (does not include derivatives)
INTEGER(KIND=JPIM) :: NDIMT     ! number of temporally interpolated fields (includes
derivatives)
END TYPE TGMVSCPL

!
=================================================================
======

!     2.    DECLARATIONS
!           ------------

!     2.0   Dimensions.

! JPALFNM  : Dimension for reading alpha function parameters
INTEGER(KIND=JPIM), PARAMETER :: JPALFNM=10

!     2.1   Control frequency of LBC.

! NEFRCL   : frequency of updating the lateral boundary coupling fields.
!            The LBC fields will be updated every NEFRCL time steps.
! TEFRCL   : time interval between two updatings of the lateral boundary fields

INTEGER(KIND=JPIM) :: NEFRCL
REAL(KIND=JPRB) :: TEFRCL

!     2.2   Number of coupled fields, structures for coupled fields.

! YYTGMVSTENC  : contains pointers and number of coupled fields for GMVS in LTENC option
! YYTGMVCPL    : contains pointers and number of coupled fields for GMV
! YYTGMVSCPL   : contains pointers and number of coupled fields for GMVS
! NDIMCPL      : number of GFL fields with true LCOUPLING attribute.
! NGALEF       : total number of coupled fields.

TYPE(TGMVSTENC) :: YYTGMVSTENC
TYPE(TGMVCPL)   :: YYTGMVCPL
TYPE(TGMVSCPL)  :: YYTGMVSCPL
INTEGER(KIND=JPIM) :: NDIMCPL
INTEGER(KIND=JPIM) :: NGALEF

!     2.3   LECOBI.

! LECOBI  : T if there is coupling and biperiodicisation

LOGICAL :: LECOBI

!     2.4   Relaxation coefficients.
```

```
! EALFA_GMV    : relaxation coefficients alpha for GMV.
! EALFA_GMVS   : relaxation coefficients alpha for GMVS.
! EALFA_GFL    : relaxation coefficients alpha for GFL.
! EALFA_TENC   : relaxation coefficients alpha for LTENC (GMVS only).
! EALFAGT3GMV  : ALFA (relax. coef.) of coupling points for GMV
! EALFAGT3GMVS : ALFA (relax. coef.) of coupling points for GMVS
! EALFAGT3GFL  : ALFA (relax. coef.) of coupling points for GFL

REAL(KIND=JPRB),ALLOCATABLE:: EALFA_GMV(:,:)
REAL(KIND=JPRB),ALLOCATABLE:: EALFA_GMVS(:,:)
REAL(KIND=JPRB),ALLOCATABLE:: EALFA_GFL(:,:)
REAL(KIND=JPRB),ALLOCATABLE:: EALFA_TENC(:,:)
REAL(KIND=JPRB),ALLOCATABLE:: EALFAGT3GMV(:,:)
REAL(KIND=JPRB),ALLOCATABLE:: EALFAGT3GMVS(:,:)
REAL(KIND=JPRB),ALLOCATABLE:: EALFAGT3GFL(:,:)

!      2.5   Other variables for grid-point coupling.

! NEDLST        : Nb. of points in E+I area (DM-local var.)
! GMGT3         : GM array of coupling points
! NLATGPP,NLONGPP: global lat,lon indx of a jproma,jgpblks point in gp-space
! NIND_LIST,NIND_LEN: help arrays for memory transfers between
!  (NPROMA,NGPBLKS)-dimensioned arrays and NEDLST-dimensioned arrays.
! GMVCPL, GMVSCPL, GFLCPL: buffers containing the LBC for GMV, GMVS, GFL
! GMVSTENC      : cf. GMVSCPL but for LTENC.
! MGMV0         : index for "GMV" memory transfers between GMV and coupling buffers.
! MGMV1         : index for "GMV" memory transfers between GMVT1 and coupling buffers.
! MGMVS0        : index for "GMVS" memory transfers between GMVS and coupling buffers.
! MGMVS1        : index for "GMVS" memory transfers between GMVT1S and coupling buffers.
! CCFIELD_GMV, CCFIELD_GMVS, CCFIELD_GFL: fields names for EWRLSGRAD.
! EWB           : weights for couplings
! EWBDFIFW      : weights for forward DFI
! EWBDFIBW      : weights for backward DFI

INTEGER(KIND=JPIM) :: NEDLST
REAL(KIND=JPRB),ALLOCATABLE:: GMGT3(:)
INTEGER(KIND=JPIM),ALLOCATABLE:: NLATGPP(:,:)
INTEGER(KIND=JPIM),ALLOCATABLE:: NLONGPP(:,:)
INTEGER(KIND=JPIM),ALLOCATABLE:: NIND_LIST(:,:)
INTEGER(KIND=JPIM),ALLOCATABLE:: NIND_LEN(:)
REAL(KIND=JPRB),ALLOCATABLE:: GMVCPL(:,:,:,:)
REAL(KIND=JPRB),ALLOCATABLE:: GMVSCPL(:,:,:)
REAL(KIND=JPRB),ALLOCATABLE:: GFLCPL(:,:,:,:)
REAL(KIND=JPRB),ALLOCATABLE:: GMVSTENC(:,:,:)
INTEGER(KIND=JPIM),ALLOCATABLE:: MGMV0(:)
INTEGER(KIND=JPIM),ALLOCATABLE:: MGMV1(:)
INTEGER(KIND=JPIM),ALLOCATABLE:: MGMVS0(:)
INTEGER(KIND=JPIM),ALLOCATABLE:: MGMVS1(:)
CHARACTER(LEN=12),ALLOCATABLE:: CCFIELD_GMV(:)
CHARACTER(LEN=12),ALLOCATABLE:: CCFIELD_GMVS(:)
CHARACTER(LEN=12),ALLOCATABLE:: CCFIELD_GFL(:)
REAL(KIND=JPRB),ALLOCATABLE:: EWB(:,:,:)
```

```
REAL(KIND=JPRB),ALLOCATABLE:: EWBDFIFW(:,:,:,:)
REAL(KIND=JPRB),ALLOCATABLE:: EWBDFIBW(:,:,:,:)
!      2.6   Other variables for spectral nudging.

! GT3SPBUF   : buffer for spectral boundary fields
! LSPNUSPDL  : .TRUE. if spectral nudging on Ps is relevent on this MPI task
! NOFFGT3BSP : Supposed to be half the size of GT3SPBUF (which contains 2 sets of fields)
! RNUDTFRAC  : Time fraction for spectral nudging
! LNUDSPGFL  : An array to control if any spectral GFL, for nudging

REAL(KIND=JPRB),ALLOCATABLE :: GT3SPBUF(:)
LOGICAL :: LSPNUSPDL
INTEGER(KIND=JPIM) :: NOFFGT3BSP
REAL(KIND=JPRB) :: RNUDTFRAC
LOGICAL, ALLOCATABLE :: LNUDSPGFL(:)

!
=============================================================================
======

CONTAINS

!      3.    SET-UP

SUBROUTINE SUELBC0B

!-----------------------------------------------------------------------
! Sets-up part 0B of forcing a LAM model by another model
!-----------------------------------------------------------------------

!-----------------------------------------------------------------------

REAL(KIND=JPRB) :: ZHOOK_HANDLE

REAL(KIND=JPRB) :: ZEPA_GMV(JPALFNM)
REAL(KIND=JPRB) :: ZEPA_GMVS(JPALFNM)
REAL(KIND=JPRB) :: ZEPA_GFL(JPALFNM)
REAL(KIND=JPRB),ALLOCATABLE :: ZB(:,:,:),ZEALP(:),ZREPA(:),ZMREPA(:),ZEALFA(:,:,:)
REAL(KIND=JPRB),ALLOCATABLE :: ZGP(:,:,:),ZSPM(:,:)
INTEGER(KIND=JPIM),ALLOCATABLE :: INEAL(:),INNAL(:),INMAL(:)
INTEGER(KIND=JPIM),ALLOCATABLE :: IBICO(:)

INTEGER(KIND=JPIM) :: IENDLON, ISTLON, IBZONGL, ILONG, IPROMA,
IGPBLKS,IGPTOT
INTEGER(KIND=JPIM) :: IA, IGLG, IIA, IJA, IROF, ISUP, IW, IW_TENC, IDLST, ICPL, IND
INTEGER(KIND=JPIM) :: JFLD, JGFL, JK, JA, JIA, JJA, JLON, JGL, JGPBLKS, JROMA
INTEGER(KIND=JPIM) :: IWS_TENC,INSTEP

REAL(KIND=JPRB) :: ZA, ZE, ZO, ZRZONG, ZRZONL, ZXA, ZYA, ZDIV, ZREM, ZWB
REAL(KIND=JPRB) :: ZEPS=1.E-4_JPRB
REAL(KIND=JPRB) :: ZEPS2=1.E-10_JPRB
INTEGER(KIND=JPIM) :: INEFRCLDFI, INEFRCLDFIA, ISWP
```

```
REAL(KIND=JPRB) :: ZT, ZT1, ZT2, ZT3, ZT4
!-----------------------------------------------------------------------

#include "abor1.intfb.h"
#include "ereespe.intfb.h"
#include "esperee.intfb.h"
#include "posnam.intfb.h"
#include "fctez.func.h"

#include "nemelbc0b.nam.h"

!-----------------------------------------------------------------------
IF (LHOOK) CALL DR_HOOK('ELBC0B_MOD:SUELBC0B',0,ZHOOK_HANDLE)
!-----------------------------------------------------------------------

!-----------------------------------------------------------------------

!      Part A: default values.

! * Relaxation coefficients
ZEPA_GMV(:)=2.16_JPRB
ZEPA_GMVS(:)=2.16_JPRB
ZEPA_GFL(:)=5.52_JPRB

! * Control frequency of LBC
TEFRCL=TSTEP

!-----------------------------------------------------------------------

!      Part B: namelist reading.

CALL POSNAM(NULNAM,'NEMELBC0B')
READ(NULNAM,NEMELBC0B)

!-----------------------------------------------------------------------

!      Part C: checkings, modify values.

! * C1: Control frequency of LBC (compute NEFRCL)

! TEFRCL, NEFRCL:
IF (TSTEP > ZEPS) THEN
  ZDIV = TEFRCL/TSTEP
  ZREM = ZDIV - REAL(NINT(ZDIV),JPRB)
  ! 1 second/coupling-interval error is tolerated
  IF(ABS(ZREM) >= 1._JPRB/TSTEP) THEN
    WRITE(UNIT=NULOUT,FMT='(" TEFRCL MUST BE A MULTIPLE ","OF TSTEP")')
    CALL ABOR1('SUELBC0B: TEFRCL MUST BE A MULTIPLE OF TSTEP')
  ELSE
    NEFRCL=NINT(TEFRCL/TSTEP)
  ENDIF
ELSE
```

```
  NEFRCL=1
ENDIF

! * C1.1: Weights for LBC interpolation (compute EWB, EWBDFIFW, EWBDFIBW)

! Fill EWB:
IF ((NSTOP/=0).AND.(NEFRCL/=0)) THEN

  ALLOCATE(EWB(0:NSTOP,1:4,0:9))
  EWB(:,:,:)=0.0_JPRB

  IF (LQCPL) THEN
    DO INSTEP=0,NSTOP
      IF (INSTEP<NEFRCL) THEN
        ! KTIMLEV=0
        EWB(INSTEP,1,0)=REAL(NEFRCL-
MOD(INSTEP,NEFRCL),JPRB)/REAL(NEFRCL,JPRB)
        EWB(INSTEP,2,0)=REAL(MOD(INSTEP,NEFRCL),JPRB)/REAL(NEFRCL,JPRB)
        EWB(INSTEP,3,:)=0.0_JPRB

        ! KTIMLEV=1
        EWB(INSTEP,1,1)=EWB(INSTEP,1,0)-1.0_JPRB/REAL(NEFRCL,JPRB)
        EWB(INSTEP,2,1)=EWB(INSTEP,2,0)+1.0_JPRB/REAL(NEFRCL,JPRB)

        ! KTIMLEV=9
        EWB(INSTEP,1,9)=EWB(INSTEP,1,0)+1.0_JPRB/REAL(NEFRCL,JPRB)
        EWB(INSTEP,2,9)=EWB(INSTEP,2,0)-1.0_JPRB/REAL(NEFRCL,JPRB)
      ELSE
        ISWP = MOD(INSTEP, NEFRCL) + 1

        ZT  = REAL(    ISWP, JPRB)
        ZT1 = REAL(-1*NEFRCL, JPRB)
        ZT2 = REAL( 0*NEFRCL, JPRB)
        ZT3 = REAL(+1*NEFRCL, JPRB)

        EWB(INSTEP,1,1)=          (ZT-ZT2)*(ZT-ZT3) / (          (ZT1-ZT2)*(ZT1-ZT3))
        EWB(INSTEP,2,1)= (ZT-ZT1)         *(ZT-ZT3) / ((ZT2-ZT1)         *(ZT2-ZT3))
        EWB(INSTEP,3,1)= (ZT-ZT1)*(ZT-ZT2)          / ((ZT3-ZT1)*(ZT3-ZT2)          )
      ENDIF
    ENDDO
  ELSEIF (LCCPL) THEN
    DO INSTEP=0,NSTOP
      IF (INSTEP<NEFRCL) THEN
        ! KTIMLEV=0
        EWB(INSTEP,1,0)=REAL(NEFRCL-
MOD(INSTEP,NEFRCL),JPRB)/REAL(NEFRCL,JPRB)
        EWB(INSTEP,2,0)=REAL(MOD(INSTEP,NEFRCL),JPRB)/REAL(NEFRCL,JPRB)
        EWB(INSTEP,3,:)=0.0_JPRB
        EWB(INSTEP,4,:)=0.0_JPRB

        ! KTIMLEV=1
        EWB(INSTEP,1,1)=EWB(INSTEP,1,0)-1.0_JPRB/REAL(NEFRCL,JPRB)
```

```
      EWB(INSTEP,2,1)=EWB(INSTEP,2,0)+1.0_JPRB/REAL(NEFRCL,JPRB)


      ! KTIMLEV=9
      EWB(INSTEP,1,9)=EWB(INSTEP,1,0)+1.0_JPRB/REAL(NEFRCL,JPRB)
      EWB(INSTEP,2,9)=EWB(INSTEP,2,0)-1.0_JPRB/REAL(NEFRCL,JPRB)

    ELSEIF (INSTEP<(2*NEFRCL))THEN

      ISWP = MOD(INSTEP, NEFRCL) + 1

      ZT  = REAL(    ISWP, JPRB)
      ZT1 = REAL(-1*NEFRCL, JPRB)
      ZT2 = REAL( 0*NEFRCL, JPRB)
      ZT3 = REAL(+1*NEFRCL, JPRB)

      EWB(INSTEP,1,1)=          (ZT-ZT2)*(ZT-ZT3) / (          (ZT1-ZT2)*(ZT1-ZT3))
      EWB(INSTEP,2,1)= (ZT-ZT1)         *(ZT-ZT3) / ((ZT2-ZT1)          *(ZT2-ZT3))
      EWB(INSTEP,3,1)= (ZT-ZT1)*(ZT-ZT2)          / ((ZT3-ZT1)*(ZT3-ZT2)          )

    ELSE

      ISWP = MOD(INSTEP, NEFRCL) + 1

      ZT  = REAL(     ISWP, JPRB)

      IF(INSTEP>=(NSTOP-NEFRCL))THEN

        ZT1 = REAL(-1*NEFRCL, JPRB)
        ZT2 = REAL( 0*NEFRCL, JPRB)
        ZT3 = REAL(+1*NEFRCL, JPRB)

        EWB(INSTEP,1,1)= 0.0_JPRB
        EWB(INSTEP,2,1)=          (ZT-ZT2)*(ZT-ZT3) / (          (ZT1-ZT2)*(ZT1-ZT3))
        EWB(INSTEP,3,1)= (ZT-ZT1)         *(ZT-ZT3) / ((ZT2-ZT1)          *(ZT2-ZT3))
        EWB(INSTEP,4,1)= (ZT-ZT1)*(ZT-ZT2)          / ((ZT3-ZT1)*(ZT3-ZT2)          )

      ELSE

        ZT1 = REAL(-1*NEFRCL, JPRB)
        ZT2 = REAL( 0*NEFRCL, JPRB)
        ZT3 = REAL(+1*NEFRCL, JPRB)
        ZT4 = REAL(+2*NEFRCL, JPRB)

        EWB(INSTEP,1,1)=          (ZT-ZT2)*(ZT-ZT3)*(ZT-ZT4) / (          (ZT1-ZT2)*(ZT1-
ZT3)*(ZT1-ZT4))
        EWB(INSTEP,2,1)= (ZT-ZT1)         *(ZT-ZT3)*(ZT-ZT4) / ((ZT2-ZT1)          *(ZT2-
ZT3)*(ZT2-ZT4))
        EWB(INSTEP,3,1)= (ZT-ZT1)*(ZT-ZT2)         *(ZT-ZT4) / ((ZT3-ZT1)*(ZT3-ZT2)
*(ZT3-ZT4))
        EWB(INSTEP,4,1)= (ZT-ZT1)*(ZT-ZT2)*(ZT-ZT3)          / ((ZT4-ZT1)*(ZT4-ZT2)*(ZT4-
ZT3)          )
```

```
      ENDIF

    ENDIF
   ENDDO
  ELSE
   DO INSTEP=0,NSTOP

    ! KTIMLEV=0
    EWB(INSTEP,1,0)=REAL(NEFRCL-MOD(INSTEP,NEFRCL),JPRB)/REAL(NEFRCL,JPRB)
    EWB(INSTEP,2,0)=REAL(MOD(INSTEP,NEFRCL),JPRB)/REAL(NEFRCL,JPRB)

    ! KTIMLEV=1
    EWB(INSTEP,1,1)=EWB(INSTEP,1,0)-1.0_JPRB/REAL(NEFRCL,JPRB)
    EWB(INSTEP,2,1)=EWB(INSTEP,2,0)+1.0_JPRB/REAL(NEFRCL,JPRB)

    ! KTIMLEV=9
    EWB(INSTEP,1,9)=EWB(INSTEP,1,0)+1.0_JPRB/REAL(NEFRCL,JPRB)
    EWB(INSTEP,2,9)=EWB(INSTEP,2,0)-1.0_JPRB/REAL(NEFRCL,JPRB)

   ENDDO

  ENDIF
ENDIF

IF (LDFI) THEN

  ! Fill EWBDFIFW (DFI forward weights):
  ALLOCATE(EWBDFIFW(0:2*NSTDFI,1:2,0:9,0:1))
  INEFRCLDFI=TEFRCL/RTDFI

  DO INSTEP=0,2*NSTDFI
   ! KTIMLEV=0
   EWBDFIFW(INSTEP,1,0,:)=REAL(INEFRCLDFI-MOD(INSTEP,INEFRCLDFI)
+NSTDFI,JPRB)/&
    & REAL(INEFRCLDFI,JPRB)
   EWBDFIFW(INSTEP,2,0,:)=REAL(MOD(INSTEP,INEFRCLDFI)-NSTDFI,JPRB)/&
    & REAL(INEFRCLDFI,JPRB)
   ! KTIMLEV=1, LBIAS=F
   EWBDFIFW(INSTEP,1,1,0)=EWBDFIFW(INSTEP,1,0,0)-
1.0_JPRB/REAL(INEFRCLDFI,JPRB)

EWBDFIFW(INSTEP,2,1,0)=EWBDFIFW(INSTEP,2,0,0)+1.0_JPRB/REAL(INEFRCLDFI,JPRB)
   ! KTIMLEV=1, LBIAS=T
   EWBDFIFW(INSTEP,1,1,1)=REAL(INEFRCLDFI+MOD(INSTEP,INEFRCLDFI)-
NSTDFI,JPRB)/&
    & REAL(INEFRCLDFI,JPRB)+1.0_JPRB/REAL(INEFRCLDFI,JPRB)
   EWBDFIFW(INSTEP,2,1,1)=REAL(-MOD(INSTEP,INEFRCLDFI)+NSTDFI,JPRB)/&
    & REAL(INEFRCLDFI,JPRB)-1.0_JPRB/REAL(INEFRCLDFI,JPRB)
   ! KTIMLEV=9, LBIAS=F

EWBDFIFW(INSTEP,1,9,0)=EWBDFIFW(INSTEP,1,0,0)+1.0_JPRB/REAL(INEFRCLDFI,JPRB)
```

```
   EWBDFIFW(INSTEP,2,9,0)=EWBDFIFW(INSTEP,2,0,0)-
1.0_JPRB/REAL(INEFRCLDFI,JPRB)
   ! KTIMLEV=9, LBIAS=T
   EWBDFIFW(INSTEP,1,9,1)=REAL(INEFRCLDFI+MOD(INSTEP,INEFRCLDFI)-
NSTDFI,JPRB)/&
    & REAL(INEFRCLDFI,JPRB)-1.0_JPRB/REAL(INEFRCLDFI,JPRB)
   EWBDFIFW(INSTEP,2,9,1)=REAL(-MOD(INSTEP,INEFRCLDFI)+NSTDFI,JPRB)/&
    & REAL(INEFRCLDFI,JPRB)+1.0_JPRB/REAL(INEFRCLDFI,JPRB)
  ENDDO

  ! Fill EWBDFIBW (DFI backward weights):
  ALLOCATE(EWBDFIBW(0:2*NSTDFIA,1:2,0:9,0:1))
  INEFRCLDFIA=TEFRCL/RTDFIA

  DO INSTEP=0,2*NSTDFIA

EWBDFIBW(INSTEP,1,0,:)=REAL(INEFRCLDFIA+MOD(INSTEP,INEFRCLDFIA),JPRB)/RE
AL(INEFRCLDFIA,JPRB)
   EWBDFIBW(INSTEP,2,0,:)=-
REAL(MOD(INSTEP,INEFRCLDFIA),JPRB)/REAL(INEFRCLDFIA,JPRB)

   ! KTIMLEV=1, LBIAS=F

EWBDFIBW(INSTEP,1,1,0)=EWBDFIBW(INSTEP,1,0,0)+1.0_JPRB/REAL(INEFRCLDFIA,JPR
B)
   EWBDFIBW(INSTEP,2,1,0)=EWBDFIBW(INSTEP,2,0,0)-
1.0_JPRB/REAL(INEFRCLDFIA,JPRB)
   ! KTIMLEV=1, LBIAS=T
   EWBDFIBW(INSTEP,1,1,1)=REAL(INEFRCLDFIA-
MOD(INSTEP,INEFRCLDFIA),JPRB)/REAL(INEFRCLDFIA,JPRB)&
    & -1.0_JPRB/REAL(INEFRCLDFIA,JPRB)

EWBDFIBW(INSTEP,2,1,1)=REAL(MOD(INSTEP,INEFRCLDFIA),JPRB)/REAL(INEFRCLDFI
A,JPRB)&
    & +1.0_JPRB/REAL(INEFRCLDFIA,JPRB)
   ! KTIMLEV=9, LBIAS=F
   EWBDFIBW(INSTEP,1,9,0)=EWBDFIBW(INSTEP,1,0,0)-
1.0_JPRB/REAL(INEFRCLDFIA,JPRB)

EWBDFIBW(INSTEP,2,9,0)=EWBDFIBW(INSTEP,2,0,0)+1.0_JPRB/REAL(INEFRCLDFIA,JPR
B)
   ! KTIMLEV=9, LBIAS=T
   EWBDFIBW(INSTEP,1,9,1)=REAL(INEFRCLDFIA-
MOD(INSTEP,INEFRCLDFIA),JPRB)/REAL(INEFRCLDFIA,JPRB)&
    & +1.0_JPRB/REAL(INEFRCLDFIA,JPRB)

EWBDFIBW(INSTEP,2,9,1)=REAL(MOD(INSTEP,INEFRCLDFIA),JPRB)/REAL(INEFRCLDFI
A,JPRB)&
    & -1.0_JPRB/REAL(INEFRCLDFIA,JPRB)
  ENDDO

ENDIF
```

```
! * C2: Calculation of YYGMVSTENC, YYTGMVCPL, YYGMVSCPL, NDIMCPL, NGALEF:

! YYGMVSTENC:
IF (LTENC) THEN
  YYTGMVSTENC%MSP=1
  YYTGMVSTENC%MSPL=2
  YYTGMVSTENC%MSPM=3
  YYTGMVSTENC%NDIM=3
  YYTGMVSTENC%NDIMT=3
ELSE
  YYTGMVSTENC%MSP=1
  YYTGMVSTENC%MSPL=1
  YYTGMVSTENC%MSPM=1
  YYTGMVSTENC%NDIM=1
  YYTGMVSTENC%NDIMT=1
ENDIF

! YYTGMVCPL:
IF (NCONF == 701) THEN
  ! derivatives are useless because ESEIMPLS is not called.
  IF (LNHDYN.AND.LNHX) THEN
    YYTGMVCPL%MU   = 1
    YYTGMVCPL%MV   = 2
    YYTGMVCPL%MT   = 3
    YYTGMVCPL%MSPD = 4
    YYTGMVCPL%MSVD = 5
    YYTGMVCPL%MNHX = 6
    YYTGMVCPL%NDIM = 6
    YYTGMVCPL%NDIMT= 6
  ELSEIF (LNHDYN.AND.(.NOT.LNHX)) THEN
    YYTGMVCPL%MU   = 1
    YYTGMVCPL%MV   = 2
    YYTGMVCPL%MT   = 3
    YYTGMVCPL%MSPD = 4
    YYTGMVCPL%MSVD = 5
    YYTGMVCPL%MNHX = 5
    YYTGMVCPL%NDIM = 5
    YYTGMVCPL%NDIMT= 5
  ELSE
    YYTGMVCPL%MU   = 1
    YYTGMVCPL%MV   = 2
    YYTGMVCPL%MT   = 3
    YYTGMVCPL%MSPD = 3
    YYTGMVCPL%MSVD = 3
    YYTGMVCPL%MNHX = 3
    YYTGMVCPL%NDIM = 3
    YYTGMVCPL%NDIMT= 3
  ENDIF
ELSE
! derivatives are useful because ESEIMPLS is called.
  IF (LNHDYN.AND.LNHX) THEN
```

```
    YYTGMVCPL%MU   = 1
    YYTGMVCPL%MV   = 2
    YYTGMVCPL%MT   = 3
    YYTGMVCPL%MSPD = 4
    YYTGMVCPL%MSVD = 5
    YYTGMVCPL%MNHX = 6
    YYTGMVCPL%NDIM = 6
    YYTGMVCPL%MDIV = 7
    YYTGMVCPL%MTL  = 8
    YYTGMVCPL%MTM  = 9
    YYTGMVCPL%MSPDL=10
    YYTGMVCPL%MSPDM=11
    YYTGMVCPL%NDIMT=11
  ELSEIF (LNHDYN.AND.(.NOT.LNHX)) THEN
    YYTGMVCPL%MU   = 1
    YYTGMVCPL%MV   = 2
    YYTGMVCPL%MT   = 3
    YYTGMVCPL%MSPD = 4
    YYTGMVCPL%MSVD = 5
    YYTGMVCPL%MNHX = 5
    YYTGMVCPL%NDIM = 5
    YYTGMVCPL%MDIV = 6
    YYTGMVCPL%MTL  = 7
    YYTGMVCPL%MTM  = 8
    YYTGMVCPL%MSPDL= 9
    YYTGMVCPL%MSPDM=10
    YYTGMVCPL%NDIMT=10
  ELSE
    YYTGMVCPL%MU   = 1
    YYTGMVCPL%MV   = 2
    YYTGMVCPL%MT   = 3
    YYTGMVCPL%MSPD = 3
    YYTGMVCPL%MSVD = 3
    YYTGMVCPL%MNHX = 3
    YYTGMVCPL%NDIM = 3
    YYTGMVCPL%MDIV = 4
    YYTGMVCPL%MTL  = 5
    YYTGMVCPL%MTM  = 6
    YYTGMVCPL%MSPDL= 6
    YYTGMVCPL%MSPDM= 6
    YYTGMVCPL%NDIMT= 6
  ENDIF
ENDIF

! YYGMVSCPL:
IF (NCONF == 701) THEN
  ! derivatives are useless because ESEIMPLS is not called.
  YYTGMVSCPL%MSP=1
  YYTGMVSCPL%NDIM=1
  YYTGMVSCPL%NDIMT=1
ELSE
  ! derivatives are useful because ESEIMPLS is called.
```

```
  YYTGMVSCPL%MSP=1
  YYTGMVSCPL%MSPL=2
  YYTGMVSCPL%MSPM=3
  YYTGMVSCPL%NDIM=1
  YYTGMVSCPL%NDIMT=3
ENDIF

! NDIMCPL:
ICPL=0
DO JGFL=1,YGFL%NUMFLDS
  IF (YGFLC(JGFL)%NCOUPLING /= 0) THEN
    ICPL=ICPL+1
  ENDIF
ENDDO
NDIMCPL=ICPL

! NGALEF:
NGALEF=YYTGMVCPL%NDIM+YYTGMVSCPL%NDIM+NDIMCPL

! * C3: Calculation of IBICO and LECOBI.

ALLOCATE(IBICO(NGALEF))
! GMV:
IBICO(YYTGMVCPL%MU)=NBICOU
IBICO(YYTGMVCPL%MV)=NBICOU
IBICO(YYTGMVCPL%MT)=NBICOT
IF (LNHDYN) IBICO(YYTGMVCPL%MSPD)=NBICPD
IF (LNHDYN) IBICO(YYTGMVCPL%MSVD)=NBICVD
IF (LNHDYN.AND.LNHX) IBICO(YYTGMVCPL%MNHX)=NBICNHX
! GMVS:
IBICO(YYTGMVCPL%NDIM+YYTGMVSCPL%MSP)=NBICOP
! GFL:
DO JFLD=1,NDIMCPL
  IBICO(YYTGMVCPL%NDIM+YYTGMVSCPL%NDIM+JFLD)=1
ENDDO

IF((NBZONL /= 0).OR.(NBZONG /= 0).OR.(NDLUXG /= NDLON).OR.(NDGUXG /= NDGLG))
THEN
  IF ( MAXVAL(IBICO(1:NGALEF))==0 .AND. MINVAL(IBICO(1:NGALEF))==0 ) THEN
    ! no field coupled; LECOBI set to F.
    LECOBI=.FALSE.
  ELSE
    ! at least one field coupled; non-empty coupling zone.
    LECOBI=.TRUE.
  ENDIF
ELSE
  ! empty coupling zone.
  LECOBI=.FALSE.
ENDIF

! * C4: Relaxation coefficients EALFA_GMV, EALFA_GMVS, EALFA_GFL, EALFA_TENC
(former SUEBICU).
```

```
ALLOCATE(EALFA_GMV(NGPTOT+1,YYTGMVCPL%NDIM))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'EALFA_GMV
',SIZE(EALFA_GMV ),SHAPE(EALFA_GMV )

ALLOCATE(EALFA_GMVS(NGPTOT+1,YYTGMVSCPL%NDIM))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)")
'EALFA_GMVS',SIZE(EALFA_GMVS),SHAPE(EALFA_GMVS)

ALLOCATE(EALFA_GFL(NGPTOT+1,NDIMCPL))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'EALFA_GFL
',SIZE(EALFA_GFL ),SHAPE(EALFA_GFL )

IF (LTENC) THEN
  ALLOCATE(EALFA_TENC(NGPTOT+1,YYTGMVSTENC%NDIM))
  IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)")
'EALFA_TENC',SIZE(EALFA_TENC),SHAPE(EALFA_TENC)
ENDIF

IF (LECOBI) THEN

  ! * C4.1: allocations.

  ALLOCATE(ZREPA(NGALEF))
  ALLOCATE(ZMREPA(NGALEF))
  ALLOCATE(ZEALP(NGALEF))
  ALLOCATE(ZEALFA(NDLON,NGALEF,NDGLG))
  ALLOCATE(INEAL(NGALEF))
  ALLOCATE(INNAL(NGALEF))
  ALLOCATE(INMAL(NGALEF))
  ALLOCATE(ZB(NBZONL+1,NBZONG+1,NGALEF))

  ISUP=100
  IBZONGL=MAX(NBZONL-NBIPINCIX,NBZONG-NBIPINCIY)

  ! * C4.2: fill ZREPA.

  ICPL=0
  ZREPA(ICPL+1:ICPL+YYTGMVCPL%NDIM)=ZEPA_GMV(1:YYTGMVCPL%NDIM)
  ICPL=YYTGMVCPL%NDIM
  ZREPA(ICPL+1:ICPL+YYTGMVSCPL%NDIM)=ZEPA_GMVS(1:YYTGMVSCPL%NDIM)
  ICPL=YYTGMVCPL%NDIM+YYTGMVSCPL%NDIM
  ZREPA(ICPL+1:ICPL+NDIMCPL)=ZEPA_GFL(1:NDIMCPL)

  DO JFLD =1,NGALEF
    IF((ZREPA(JFLD) > -2.0_JPRB).AND.(ZREPA(JFLD) < 2.0_JPRB))THEN
      WRITE(UNIT=NULOUT,FMT='(''ERROR ZREPA CANT BE'',F5.2)')ZREPA(JFLD)
      CALL ABOR1('SUELBC0B: ZREPA MUST HAVE A VALUE STRICTLY BETWEEN -2.0
AND 2.0')
    ENDIF
  ENDDO
```

```
  ! * C4.3: fill INEAL,INMAL,INNAL (identical for all coupled fields).

  INEAL(1:NGALEF)=2
  INMAL(1:NGALEF)=1

  IF (IBZONGL>=11 .AND. IBZONGL<=26) THEN
    INNAL(1:NGALEF)=2
    WRITE(NULOUT,*) 'INNAL FORCED TO 2 FOR CONVERGENCE'
  ELSEIF (IBZONGL>=27) THEN
    INNAL(1:NGALEF)=1
    WRITE(NULOUT,*) 'INNAL FORCED TO 1 FOR CONVERGENCE'
  ELSE
    INNAL(1:NGALEF)=3
    WRITE(NULOUT,*) 'INNAL SET TO 3'
  ENDIF

  ! * C4.4: compute auxilary variables ZRZONL, ZRZONG, ZEALP (identical for all coupled
fields).

  IF (NBZONL > NBIPINCIX) THEN
    ZRZONL=1.0_JPRB/REAL(NBZONL-NBIPINCIX,JPRB)
  ENDIF
  IF (NBZONG > NBIPINCIY) THEN
    ZRZONG=1.0_JPRB/REAL(NBZONG-NBIPINCIY,JPRB)
  ENDIF
  DO JFLD =1,NGALEF
    ZEALP(JFLD)=&
     & REAL((INMAL(JFLD)+INNAL(JFLD))**(INMAL(JFLD)+INNAL(JFLD)),JPRB)/&
     &
REAL((INNAL(JFLD)**INNAL(JFLD))*(INMAL(JFLD)**INMAL(JFLD))*INEAL(JFLD))
  ENDDO

  ! * C4.5: compute ZEALFA.

  DO JFLD =1,NGALEF

    IF (IBICO(JFLD) == 0) THEN

      ! --- no coupling applied to this field; we simply set ZEALFA=0 everywhere.
      ZEALFA(1:NDLON,JFLD,1:NDGLG)=0.0_JPRB

    ELSE

      ! --- coupling applied to this field.

      ! * ZEALFA: initialize the center domain to 0. and the outer domain to 1.

      ZEALFA(NDLUNG+NBZONL:NDLUXG-NBZONL,JFLD,NDGUNG+NBZONG:NDGUXG-
NBZONG)=0.0_JPRB
      ZEALFA(1:NDLON,JFLD,1:NDGUNG-1)=1.0_JPRB
      ZEALFA(1:NDLON,JFLD,NDGUXG+1:NDGLG)=1.0_JPRB
      ZEALFA(1:NDLUNG-1,JFLD,NDGUNG:NDGUXG)=1.0_JPRB
```

```
 ZEALFA(NDLUXG+1:NDLON,JFLD,NDGUNG:NDGUXG)=1.0_JPRB

! * compute ZEALFA in the relaxation area.

 IF ((NBZONL > 0).OR.(NBZONG > 0))THEN

   ! Compute ZB:

   IF (NBZONL > 0) THEN
    DO JA=2,NBZONL
     ! relaxation function is 1 in 1:NBIPINCIX
     IF (JA<=NBZONL-NBIPINCIX) THEN
      ZA=REAL(JA-1,JPRB)*ZRZONL
      IF(ZREPA(JFLD) <= -2.0_JPRB) THEN
       ZMREPA(JFLD)=-ZREPA(JFLD)
       ZB(JA,NBZONG+1,JFLD)=FEZBM(ZA,ZMREPA(JFLD))
      ELSE
       ZB(JA,NBZONG+1,JFLD)=FEZBP(ZA,ZREPA(JFLD))
      ENDIF
     ELSE
      ZB(JA,NBZONG+1,JFLD)=1._JPRB
     ENDIF
    ENDDO
   ENDIF

   IF (NBZONG > 0) THEN
    DO JA=2,NBZONG
     ! relaxation function is 1 in 1:NBIPINCIX
     IF (JA<=NBZONG-NBIPINCIY) THEN
      ZA=REAL(JA-1,JPRB)*ZRZONG
      IF(ZREPA(JFLD) <= -2.0_JPRB) THEN
       ZMREPA(JFLD)=-ZREPA(JFLD)
       ZB(NBZONL+1,JA,JFLD)=FEZBM(ZA,ZMREPA(JFLD))
      ELSE
       ZB(NBZONL+1,JA,JFLD)=FEZBP(ZA,ZREPA(JFLD))
      ENDIF
     ELSE
      ZB(NBZONL+1,JA,JFLD)=1._JPRB
     ENDIF
    ENDDO
   ENDIF

   IF ((NBZONL > 0).AND.(NBZONG > 0)) THEN
    DO JIA=2,NBZONL
     IF (JIA<=NBZONL-NBIPINCIX) THEN
      ZXA=REAL(JIA-1,JPRB)*ZRZONL
      DO JJA=2,NBZONG
       IF (JJA<=NBZONG-NBIPINCIY) THEN
        ZYA=REAL(JJA-1,JPRB)*ZRZONG
        ZA=MAX(ZXA,ZYA)
        ZO=ZA
        DO JK=1,ISUP
```

```
          ZE=FEZE(ZA,ZEALP(JFLD),INNAL(JFLD),INMAL(JFLD))
          ZA=(ZXA**ZE+ZYA**ZE)**(1.0_JPRB/ZE)
          ZT=ABS(ZA-ZO)/ZO
          IF (ZT <= ZEPS) EXIT
          IF (JK == ISUP) THEN
            WRITE(NULOUT,*) 'NO CONVERGENCE FOR EALFA'
            CALL ABOR1('SUELBC0B: NO CONVERGENCE FOR EALFA')
          ENDIF
          ZA=0.5_JPRB*(ZA+ZO)
          ZO=ZA
        ENDDO
        IF(ZREPA(JFLD) <= -2.0_JPRB) THEN
          ZMREPA(JFLD)=-ZREPA(JFLD)
          ZB(JIA,JJA,JFLD)=FEZBM(ZA,ZMREPA(JFLD))
        ELSE
          ZB(JIA,JJA,JFLD)=FEZBP(ZA,ZREPA(JFLD))
        ENDIF
      ELSE
        ZB(JIA,JJA,JFLD)=1._JPRB
      ENDIF
    ENDDO
  ELSE
    DO JJA=2,NBZONG
      ZB(JIA,JJA,JFLD)=1._JPRB
    ENDDO
  ENDIF
 ENDDO
ENDIF

! Initialize ZEALFA on the relaxation area

IF (NBZONG > 0)THEN
 DO JLON=NDLUNG+NBZONL,NDLUXG-NBZONL
  ZEALFA(JLON,JFLD,NDGUNG)=1.0_JPRB
  DO JGL=NDGUNG+1,NDGUNG+NBZONG-1
    IA=NDGUNG+NBZONG+1-JGL
    ZEALFA(JLON,JFLD,JGL)=ZB(NBZONL+1,IA,JFLD)
  ENDDO
  ZEALFA(JLON,JFLD,NDGUXG)=1.0_JPRB
  DO JGL=NDGUXG-NBZONG+1,NDGUXG-1
    IA=JGL-NDGUXG+NBZONG+1
    ZEALFA(JLON,JFLD,JGL)=ZB(NBZONL+1,IA,JFLD)
  ENDDO
 ENDDO
ENDIF

IF(NBZONL > 0)THEN
 DO JGL=NDGUNG+NBZONG,NDGUXG-NBZONG
  ZEALFA(NDLUNG,JFLD,JGL)=1.0_JPRB
  DO JLON=NDLUNG+1,NDLUNG+NBZONL-1
    IA=NDLUNG+NBZONL+1-JLON
    ZEALFA(JLON,JFLD,JGL)=ZB(IA,NBZONG+1,JFLD)
```

```
          ENDDO
          ZEALFA(NDLUXG,JFLD,JGL)=1.0_JPRB
          DO JLON=NDLUXG-NBZONL+1,NDLUXG-1
            IA=JLON-NDLUXG+NBZONL+1
            ZEALFA(JLON,JFLD,JGL)=ZB(IA,NBZONG+1,JFLD)
          ENDDO
        ENDDO
      ENDIF

      IF((NBZONL > 0).AND.(NBZONG > 0)) THEN
        DO JLON=NDLUNG+1,NDLUNG+NBZONL-1
          IIA=NDLUNG+NBZONL+1-JLON
          DO JGL=NDGUNG+1,NDGUNG+NBZONG-1
            IJA=NDGUNG+NBZONG+1-JGL
            ZEALFA(JLON,JFLD,JGL)=ZB(IIA,IJA,JFLD)
          ENDDO
          DO JGL=NDGUXG-NBZONG+1,NDGUXG-1
            IJA=JGL-NDGUXG+NBZONG+1
            ZEALFA(JLON,JFLD,JGL)=ZB(IIA,IJA,JFLD)
          ENDDO
        ENDDO
        DO JLON=NDLUXG-NBZONL+1,NDLUXG-1
          IIA=JLON-NDLUXG+NBZONL+1
          DO JGL=NDGUNG+1,NDGUNG+NBZONG-1
            IJA=NDGUNG+NBZONG+1-JGL
            ZEALFA(JLON,JFLD,JGL)=ZB(IIA,IJA,JFLD)
          ENDDO
          DO JGL=NDGUXG-NBZONG+1,NDGUXG-1
            IJA=JGL-NDGUXG+NBZONG+1
            ZEALFA(JLON,JFLD,JGL)=ZB(IIA,IJA,JFLD)
          ENDDO
        ENDDO
        ZEALFA(NDLUNG:NDLUNG+NBZONL-1,JFLD,NDGUNG)=1.0_JPRB
        ZEALFA(NDLUNG:NDLUNG+NBZONL-1,JFLD,NDGUXG)=1.0_JPRB
        ZEALFA(NDLUXG-NBZONL+1:NDLUXG,JFLD,NDGUNG)=1.0_JPRB
        ZEALFA(NDLUXG-NBZONL+1:NDLUXG,JFLD,NDGUXG)=1.0_JPRB
        ZEALFA(NDLUNG,JFLD,NDGUNG:NDGUNG+NBZONG-1)=1.0_JPRB
        ZEALFA(NDLUXG,JFLD,NDGUNG:NDGUNG+NBZONG-1)=1.0_JPRB
        ZEALFA(NDLUNG,JFLD,NDGUXG-NBZONG+1:NDGUXG)=1.0_JPRB
        ZEALFA(NDLUXG,JFLD,NDGUXG-NBZONG+1:NDGUXG)=1.0_JPRB
      ENDIF

      ! Feed the extra-longitudes and latitudes
      ZEALFA(NDLUNG+NBZONL:NDLUXG-
NBZONL,JFLD,NDGUNG+NBZONG:NDGUXG-NBZONG)=0.0_JPRB

    ENDIF ! ((NBZONL > 0).OR.(NBZONG > 0))

  ENDIF ! IBICO(JFLD)

 ENDDO ! JFLD
```

```
! * C4.6: compute EALFA_GMV, EALFA_GMVS, EALFA_GFL, EALFA_TENC from ZEALFA.

! EALFA_GMV:
ICPL=0
DO JFLD=1,YYTGMVCPL%NDIM
  IROF=1
  DO JGL=1,NDGENL
    IGLG=MYLATS(JGL)
    ISTLON=NSTA(NPTRFLOFF+JGL,MYSETB)
    IENDLON=NSTA(NPTRFLOFF+JGL,MYSETB)+NONL(NPTRFLOFF+JGL,MYSETB)-1
    DO JLON=ISTLON,IENDLON
      EALFA_GMV(IROF,JFLD)=ZEALFA(JLON,ICPL+JFLD,IGLG)
      IROF=IROF+1
    ENDDO
  ENDDO
ENDDO

! EALFA_GMVS:
ICPL=YYTGMVCPL%NDIM
DO JFLD=1,YYTGMVSCPL%NDIM
  IROF=1
  DO JGL=1,NDGENL
    IGLG=MYLATS(JGL)
    ISTLON=NSTA(NPTRFLOFF+JGL,MYSETB)
    IENDLON=NSTA(NPTRFLOFF+JGL,MYSETB)+NONL(NPTRFLOFF+JGL,MYSETB)-1
    DO JLON=ISTLON,IENDLON
      EALFA_GMVS(IROF,JFLD)=ZEALFA(JLON,ICPL+JFLD,IGLG)
      IROF=IROF+1
    ENDDO
  ENDDO
ENDDO

! EALFA_GFL:
ICPL=YYTGMVCPL%NDIM+YYTGMVSCPL%NDIM
DO JFLD=1,NDIMCPL
  IROF=1
  DO JGL=1,NDGENL
    IGLG=MYLATS(JGL)
    ISTLON=NSTA(NPTRFLOFF+JGL,MYSETB)
    IENDLON=NSTA(NPTRFLOFF+JGL,MYSETB)+NONL(NPTRFLOFF+JGL,MYSETB)-1
    DO JLON=ISTLON,IENDLON
      EALFA_GFL(IROF,JFLD)=ZEALFA(JLON,ICPL+JFLD,IGLG)
      IROF=IROF+1
    ENDDO
  ENDDO
ENDDO

! EALFA_TENC:
IF (LTENC .AND. LRPLANE) THEN

  ! EALFA_TENC for surface pressure variable:
  IROF=1
```

```
    DO JGL=1,NDGENL
      IGLG=MYLATS(JGL)
      ISTLON=NSTA(NPTRFLOFF+JGL,MYSETB)
      IENDLON=NSTA(NPTRFLOFF+JGL,MYSETB)+NONL(NPTRFLOFF+JGL,MYSETB)-1
      DO JLON=ISTLON,IENDLON
        EALFA_TENC(IROF,YYTGMVSTENC%MSP)=EALFA_GMVS(IROF,YYTGMVSCPL
%MSP)
        IROF=IROF+1
      ENDDO
    ENDDO

    ! EALFA_TENC for horizontal derivatives of surface pressure variable:
    ALLOCATE(ZSPM(1,NSPEC2))
    ALLOCATE(ZGP(NGPTOT,1,3))
    ZGP(1:NGPTOT,1,1)=EALFA_TENC(1:NGPTOT,YYTGMVSTENC%MSP)
    CALL EREESPE(1,1,ZSPM,ZGP(1,1,1))
    CALL ESPEREE(1,1,ZSPM,ZGP(1,1,1),PREELL=ZGP(1,1,2),PREELM=ZGP(1,1,3))
    EALFA_TENC(1:NGPTOT,YYTGMVSTENC%MSPL)=ZGP(1:NGPTOT,1,2)
    EALFA_TENC(1:NGPTOT,YYTGMVSTENC%MSPM)=ZGP(1:NGPTOT,1,3)
    DEALLOCATE(ZGP)
    DEALLOCATE(ZSPM)
    DO JLON=1,NGPTOT
      IF (EALFA_TENC(JLON,YYTGMVSTENC%MSP) ==
1.0_JPRB.OR.EALFA_TENC(JLON,YYTGMVSTENC%MSP) == 0.0_JPRB) THEN
        EALFA_TENC(JLON,YYTGMVSTENC%MSPL) = 0.0_JPRB
        EALFA_TENC(JLON,YYTGMVSTENC%MSPM) = 0.0_JPRB
      ELSEIF (EALFA_TENC(JLON,YYTGMVSTENC%MSP) >
1.0_JPRB.OR.EALFA_TENC(JLON,YYTGMVSTENC%MSP) < 0.0_JPRB) THEN
        CALL ABOR1('SUELBC0B: EALFA_TENC IS OUT OF [0,1]')
      ENDIF
    ENDDO
  ENDIF

  ! * C4.7: deallocations.

  IF (ALLOCATED(ZEALP)) DEALLOCATE(ZEALP)
  IF (ALLOCATED(ZREPA)) DEALLOCATE(ZREPA)
  IF (ALLOCATED(ZMREPA)) DEALLOCATE(ZMREPA)
  IF (ALLOCATED(ZEALFA)) DEALLOCATE(ZEALFA)
  IF (ALLOCATED(INEAL)) DEALLOCATE(INEAL)
  IF (ALLOCATED(INNAL)) DEALLOCATE(INNAL)
  IF (ALLOCATED(INMAL)) DEALLOCATE(INMAL)
  IF (ALLOCATED(ZB)) DEALLOCATE(ZB)

ELSE

  EALFA_GMV(:,:)=0.0_JPRB
  EALFA_GMVS(:,:)=0.0_JPRB
  EALFA_GFL(:,:)=0.0_JPRB
  IF (LTENC) EALFA_TENC(:,:)=0.0_JPRB

ENDIF ! LECOBI
```

```
IF (ALLOCATED(IBICO)) DEALLOCATE(IBICO)

! * C5: Other variables for grid-point coupling

! * C5.1: Allocation and computation NLONGPP, NLATGPP, NIND_LIST and NIND_LEN.

ALLOCATE(NLONGPP(NPROMA,NGPBLKS))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'NLONGPP
',SIZE(NLONGPP ),SHAPE(NLONGPP )
ALLOCATE(NLATGPP(NPROMA,NGPBLKS))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'NLATGPP
',SIZE(NLATGPP ),SHAPE(NLATGPP )
ALLOCATE(NIND_LIST(NPROMA,NGPBLKS))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)")
'NIND_LIST',SIZE(NIND_LIST),SHAPE(NIND_LIST)
ALLOCATE(NIND_LEN(0:NGPBLKS))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)")
'NIND_LEN',SIZE(NIND_LEN),SHAPE(NIND_LEN)

NLATGPP(:,:)=-99999
NLONGPP(:,:)=-99999
IPROMA=0
IGPBLKS=1
DO JGL=NDGSAL,NDGENL
  IGLG=MYLATS(JGL)
  DO JLON=1,NONL(NPTRFLOFF+JGL,MY_REGION_EW)
    ILONG=NSTA(NPTRFLOFF+JGL,MY_REGION_EW)+JLON-1
    IPROMA=IPROMA+1
    IF (IPROMA > NPROMA) THEN
      IPROMA=1
      IGPBLKS=IGPBLKS+1
    ENDIF
    NLATGPP(IPROMA,IGPBLKS)=IGLG
    NLONGPP(IPROMA,IGPBLKS)=ILONG
  ENDDO
ENDDO
IF (IGPBLKS /= NGPBLKS) CALL ABOR1("SUELBC0B: CONFLICT IN NGPBLKS")

!$OMP parallel
!$OMP do private(IND,IGLG,ILONG)
DO JGPBLKS=1,NGPBLKS
  IND=0
  DO JROMA=1,NPROMA
    IF (NLATGPP(JROMA,JGPBLKS) > 0) THEN
      IGLG=NLATGPP(JROMA,JGPBLKS)
      ILONG=NLONGPP(JROMA,JGPBLKS)
      IF (ILONG <= NBZONL .OR. ILONG > NDLUXG-NBZONL &
       & .OR. IGLG <= NBZONG .OR. IGLG > NDGUXG-NBZONG) THEN
        IND=IND+1
        NIND_LIST(IND,JGPBLKS)=JROMA
      ENDIF
```

```
    ENDIF
  ENDDO
  NIND_LEN(JGPBLKS)=IND
ENDDO
!$OMP end parallel

NIND_LEN(0)=0
DO JGPBLKS=2,NGPBLKS
  NIND_LEN(JGPBLKS)=NIND_LEN(JGPBLKS)+NIND_LEN(JGPBLKS-1)
ENDDO

! * C5.2: Computation of NEDLST.

NEDLST=NIND_LEN(NGPBLKS)

! * C5.3: Allocation and computation of GMGT3, EALFAGT3GMV, EALFAGT3GMVS,
EALFAGT3GFL.

ALLOCATE(GMGT3(NEDLST))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'GMGT3
',SIZE(GMGT3 ),SHAPE(GMGT3 )

ALLOCATE(EALFAGT3GMV(NEDLST,YYTGMVCPL%NDIM))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A12,' ALLOCATED ',8I8)") 'EALFAGT3GMV
',SIZE(EALFAGT3GMV),SHAPE(EALFAGT3GMV)

ALLOCATE(EALFAGT3GMVS(NEDLST,YYTGMVSCPL%NDIM))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A12,' ALLOCATED ',8I8)")
'EALFAGT3GMVS',SIZE(EALFAGT3GMVS),SHAPE(EALFAGT3GMVS)

ALLOCATE(EALFAGT3GFL(NEDLST,NDIMCPL))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A12,' ALLOCATED ',8I8)") 'EALFAGT3GFL
',SIZE(EALFAGT3GFL),SHAPE(EALFAGT3GFL)

IGPTOT=0
IDLST=0
DO JGL=NDGSAL,NDGENL
  IGLG=MYLATS(JGL)
  DO JLON=1,NONL(NPTRFLOFF+JGL,MY_REGION_EW)
    ILONG=NSTA(NPTRFLOFF+JGL,MY_REGION_EW)+JLON-1
    IGPTOT=IGPTOT+1
    IF (ILONG <= NBZONL .OR. ILONG > NDLUXG-NBZONL &
     & .OR. IGLG <= NBZONG .OR. IGLG > NDGUXG-NBZONG) THEN
      ! point is outside C-zone==>it shouldbe coupled
      IDLST=IDLST+1
      EALFAGT3GMV(IDLST,:)=EALFA_GMV(IGPTOT,:)
      EALFAGT3GMVS(IDLST,:)=EALFA_GMVS(IGPTOT,:)
      EALFAGT3GFL(IDLST,:)=EALFA_GFL(IGPTOT,:)
      GMGT3(IDLST)=YRGSGEOM_NB%GM(IGPTOT)
    ENDIF
  ENDDO
ENDDO
```

```
! * C5.4: Allocation of GT3SPBUF, GMVCPL, GMVSCPL, GFLCPL, GMVSTENC.
IF (LCCPL) THEN
  IW=4
  IW_TENC=4
ELSEIF (LQCPL) THEN
  IW=3
  IW_TENC=3
ELSEIF (LTENC.AND.LALLTC) THEN
  IW=3
  IW_TENC=3
ELSE
  IW=2
  IW_TENC=2
ENDIF

ALLOCATE (GMVCPL(NEDLST,NFLEVG,YYTGMVCPL%NDIMT,IW))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'GMVCPL
',SIZE(GMVCPL   ),SHAPE(GMVCPL   )
ALLOCATE (GMVSCPL(NEDLST,YYTGMVSCPL%NDIMT,IW))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'GMVSCPL
',SIZE(GMVSCPL  ),SHAPE(GMVSCPL  )
ALLOCATE (GFLCPL(NEDLST,NFLEVG,NDIMCPL,IW))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'GFLCPL
',SIZE(GFLCPL   ),SHAPE(GFLCPL   )

ALLOCATE (GMVSTENC(NEDLST,YYTGMVSTENC%NDIMT,IW_TENC))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'GMVSTENC
',SIZE(GMVSTENC ),SHAPE(GMVSTENC )

! * C5.5: Allocation and computation of MGMV0, MGMV1, MGMVS0, MGMVS1:

ALLOCATE (MGMV0(YYTGMVCPL%NDIMT))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'MGMV0
',SIZE(MGMV0    ),SHAPE(MGMV0    )
MGMV0(YYTGMVCPL%MU)=YT0%MU
MGMV0(YYTGMVCPL%MV)=YT0%MV
MGMV0(YYTGMVCPL%MT)=YT0%MT
IF (LNHDYN) MGMV0(YYTGMVCPL%MSPD)=YT0%MSPD
IF (LNHDYN) MGMV0(YYTGMVCPL%MSVD)=YT0%MSVD
IF (LNHDYN.AND.LNHX) MGMV0(YYTGMVCPL%MNHX)=YT0%MNHX
IF (NCONF /= 701) THEN
  MGMV0(YYTGMVCPL%MDIV)=YT0%MDIV
  MGMV0(YYTGMVCPL%MTL)=YT0%MTL
  MGMV0(YYTGMVCPL%MTM)=YT0%MTM
  IF (LNHDYN) MGMV0(YYTGMVCPL%MSPDL)=YT0%MSPDL
  IF (LNHDYN) MGMV0(YYTGMVCPL%MSPDM)=YT0%MSPDM
ENDIF

ALLOCATE (MGMV1(YYTGMVCPL%NDIM))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'MGMV1
',SIZE(MGMV1    ),SHAPE(MGMV1    )
```

```
MGMV1(YYTGMVCPL%MU)=YT1%MU
MGMV1(YYTGMVCPL%MV)=YT1%MV
MGMV1(YYTGMVCPL%MT)=YT1%MT
IF (LNHDYN) MGMV1(YYTGMVCPL%MSPD)=YT1%MSPD
IF (LNHDYN) MGMV1(YYTGMVCPL%MSVD)=YT1%MSVD
IF (LNHDYN.AND.LNHX) MGMV1(YYTGMVCPL%MNHX)=YT1%MNHX

ALLOCATE (MGMVS0(YYTGMVSCPL%NDIMT))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'MGMVS0
',SIZE(MGMVS0   ),SHAPE(MGMVS0   )
MGMVS0(YYTGMVSCPL%MSP)=YT0%MSP
IF (NCONF /= 701) THEN
  MGMVS0(YYTGMVSCPL%MSPL)=YT0%MSPL
  MGMVS0(YYTGMVSCPL%MSPM)=YT0%MSPM
ENDIF

ALLOCATE (MGMVS1(YYTGMVSCPL%NDIM))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'MGMVS1
',SIZE(MGMVS1   ),SHAPE(MGMVS1   )
MGMVS1(YYTGMVSCPL%MSP)=YT1%MSP

! * C5.6: Allocation and computation of CCFIELD_GMV, CCFIELD_GMVS, CCFIELD_GFL:

ALLOCATE (CCFIELD_GMV(YYTGMVCPL%NDIM))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A12,' ALLOCATED ',8I8)") 'CCFIELD_GMV
',SIZE(CCFIELD_GMV),SHAPE(CCFIELD_GMV)
CCFIELD_GMV(YYTGMVCPL%MU)='LSCGRAD_UUUU'
CCFIELD_GMV(YYTGMVCPL%MV)='LSCGRAD_VVVV'
CCFIELD_GMV(YYTGMVCPL%MT)='LSCGRAD_TEMP'
IF (LNHDYN) CCFIELD_GMV(YYTGMVCPL%MSPD)='LSCGRAD_PDEP'
IF (LNHDYN) CCFIELD_GMV(YYTGMVCPL%MSVD)='LSCGRAD_VDIV'
IF (LNHDYN.AND.LNHX) CCFIELD_GMV(YYTGMVCPL%MNHX)='LSCGRAD_NHXX'

ALLOCATE (CCFIELD_GMVS(YYTGMVSCPL%NDIM))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A12,' ALLOCATED ',8I8)")
'CCFIELD_GMVS',SIZE(CCFIELD_GMVS),SHAPE(CCFIELD_GMVS)
CCFIELD_GMVS(YYTGMVSCPL%MSP)='LSCGRAD_SURP'

ALLOCATE (CCFIELD_GFL(NDIMCPL))
IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A12,' ALLOCATED ',8I8)") 'CCFIELD_GFL
',SIZE(CCFIELD_GFL),SHAPE(CCFIELD_GFL)
CCFIELD_GFL(1:NDIMCPL)='LSCGRAD_HUMQ'

! * C6: Other variables for spectral nudging

IWS_TENC=2

IF (LESPCPL) THEN
  NOFFGT3BSP=NSPEC2*(NS3D*NFLEVL+NFD2D)
  RNUDTFRAC=SIGN(1._JPRB,TSTEP)/REAL(NEFRSPCPL,JPRB)
  LSPNUSPDL=(SPNUDSP > ZEPS2).AND.(MYSETV==NBSETSP)
ELSE
```

```
  NOFFGT3BSP=0
  RNUDTFRAC=0._JPRB
  LSPNUSPDL=.FALSE.
ENDIF

IF (LESPCPL) THEN
  IF (LSPTENC) IWS_TENC=3
  ALLOCATE(GT3SPBUF(IWS_TENC*NOFFGT3BSP))
  IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)") 'GT3SPBUF
',SIZE(GT3SPBUF ),SHAPE(GT3SPBUF )
  ALLOCATE(LNUDSPGFL(MAX(1,YGFL%NUMSPFLDS)))
  IF (LALLOPR) WRITE(NULOUT,"(1X,'ARRAY ',A10,' ALLOCATED ',8I8)")
'LNUDSPGFL',SIZE(LNUDSPGFL),SHAPE(LNUDSPGFL)
  LNUDSPGFL(:)=.FALSE.
  IF ((SPNUDQ>ZEPS2).AND.YQ%MPSP>0.AND.YQ_NL%LSP) LNUDSPGFL(YQ
%MPSP)=.TRUE.
ENDIF

!-----------------------------------------------------------------------

!      Part D: printings.

WRITE(NULOUT,*) ' --- PRINTINGS IN SUELBC0B --- '

! * D1: Control frequency of LBC.
WRITE(UNIT=NULOUT,FMT='(" Frequency of LBC: ")')
WRITE(UNIT=NULOUT,FMT='("  TEFRCL = ",F10.1," NEFRCL = ",I15)') TEFRCL,NEFRCL

! * D2: Number of coupled fields.
WRITE(UNIT=NULOUT,FMT='(" Grid-point coupling: ")')
WRITE(UNIT=NULOUT,FMT='("  nb of GMV fields with temporal interpolation: YYTGMVCPL
%NDIMT = ",I4)') YYTGMVCPL%NDIMT
WRITE(UNIT=NULOUT,FMT='("  nb of coupled GMV fields: YYTGMVCPL%NDIM = ",I4)')
YYTGMVCPL%NDIM
WRITE(UNIT=NULOUT,FMT='("  nb of GMVS fields with temporal interpolation:
YYTGMVSCPL%NDIMT = ",I4)') YYTGMVSCPL%NDIMT
WRITE(UNIT=NULOUT,FMT='("  nb of coupled GMVS fields: YYTGMVSCPL%NDIM = ",I4)')
YYTGMVSCPL%NDIM
WRITE(UNIT=NULOUT,FMT='("  nb of coupled GFL fields: NDIMCPL = ",I4)') NDIMCPL
WRITE(UNIT=NULOUT,FMT='("  total nb of coupled fields: NGALEF = ",I4)') NGALEF

! * D3: LECOBI.
WRITE(NULOUT,'("  LECOBI = ",L2)') LECOBI

! * D4: Relaxation coefficients EALFA_GMV, EALFA_GMVS, EALFA_GFL, EALFA_TENC.
IF (LECOBI) THEN
  IF (LOUTPUT) THEN
    WRITE(UNIT=NULOUT,FMT='(" Relaxation coefficients: ")')
    DO JFLD=1,YYTGMVCPL%NDIM
      WRITE(UNIT=NULOUT,FMT=' ("  JFLD = ",I3)') JFLD
      WRITE(UNIT=NULOUT,FMT=' ("  EALFA_GMV FOR JFLD")')
      DO JGL=1,NDGENL,100
```

```
        IGLG=MYLATS(JGL)
        WRITE(UNIT=NULOUT,FMT='(2X,14(1X,E8.2))')
(EALFA_GMV(JLON+NSTAGP(JGL),JFLD),JLON=1,NDLON,100)
      ENDDO
    ENDDO
    DO JFLD=1,YYTGMVSCPL%NDIM
      WRITE(UNIT=NULOUT,FMT=' ("  JFLD = ",I3)') JFLD
      WRITE(UNIT=NULOUT,FMT=' ("  EALFA_GMVS FOR JFLD")')
      DO JGL=1,NDGENL,100
        IGLG=MYLATS(JGL)
        WRITE(UNIT=NULOUT,FMT='(2X,14(1X,E8.2))')
(EALFA_GMVS(JLON+NSTAGP(JGL),JFLD),JLON=1,NDLON,100)
      ENDDO
    ENDDO
    DO JFLD=1,NDIMCPL
      WRITE(UNIT=NULOUT,FMT=' ("  JFLD = ",I3)') JFLD
      WRITE(UNIT=NULOUT,FMT=' ("  EALFA_GFL FOR JFLD")')
      DO JGL=1,NDGENL,100
        IGLG=MYLATS(JGL)
        WRITE(UNIT=NULOUT,FMT='(2X,14(1X,E8.2))')
(EALFA_GFL(JLON+NSTAGP(JGL),JFLD),JLON=1,NDLON,100)
      ENDDO
    ENDDO
    IF (LTENC.AND.LRPLANE) THEN
      DO JFLD=1,YYTGMVSTENC%NDIM
        WRITE(UNIT=NULOUT,FMT=' ("  JFLD = ",I3)') JFLD
        WRITE(UNIT=NULOUT,FMT=' ("  EALFA_TENC FOR JFLD")')
        DO JGL=1,NDGENL,100
          IGLG=MYLATS(JGL)
          WRITE(UNIT=NULOUT,FMT='(2X,14(1X,E8.2))')
(EALFA_TENC(JLON+NSTAGP(JGL),JFLD),JLON=1,NDLON,100)
        ENDDO
      ENDDO
    ENDIF
  ENDIF
ENDIF

! * D5: Other variables for grid-point coupling.
WRITE(UNIT=NULOUT,FMT='(" Other variables for grid-point coupling: ")')
WRITE(UNIT=NULOUT,FMT='("  NEDLST = ",I8)') NEDLST
WRITE(UNIT=NULOUT,FMT='("  MGMV0 = ",20(1X,I2))') MGMV0(1:YYTGMVCPL
%NDIMT)
WRITE(UNIT=NULOUT,FMT='("  MGMVS0 = ",20(1X,I2))') MGMVS0(1:YYTGMVSCPL
%NDIMT)
WRITE(UNIT=NULOUT,FMT='("  MGMV1 = ",20(1X,I2))') MGMV1(1:YYTGMVCPL%NDIM)
WRITE(UNIT=NULOUT,FMT='("  MGMVS1 = ",20(1X,I2))') MGMVS1(1:YYTGMVSCPL
%NDIM)

! * D6: Other variables for spectral nudging.
IF (LESPCPL) THEN
  WRITE(UNIT=NULOUT,FMT='(" Other variables for spectral nudging: ")')
  WRITE(UNIT=NULOUT,FMT='("  NOFFGT3BSP = ",I8)') NOFFGT3BSP
```

```
  WRITE(UNIT=NULOUT,FMT='("  RNUDTFRAC = ",E20.14)') RNUDTFRAC
  WRITE(UNIT=NULOUT,FMT='("  LSPNUSPDL = ",L2)') LSPNUSPDL
  WRITE(UNIT=NULOUT,FMT='("  LNUDSPGFL = ",20(1X,L2))') LNUDSPGFL(:)
ENDIF

WRITE(NULOUT,*) ' '

!-----------------------------------------------------------------------
IF (LHOOK) CALL DR_HOOK('ELBC0B_MOD:SUELBC0B',1,ZHOOK_HANDLE)
END SUBROUTINE SUELBC0B

SUBROUTINE DEALLOCATE_ELBC0B

!-----------------------------------------------------------------------
! deallocates 'ELBC0B' arrays
!-----------------------------------------------------------------------

IMPLICIT NONE

REAL(KIND=JPRB) :: ZHOOK_HANDLE

!     ------------------------------------------------------------------
IF (LHOOK) CALL
DR_HOOK('ELBC0B_MOD:DEALLOCATE_ELBC0B',0,ZHOOK_HANDLE)
!     ------------------------------------------------------------------

IF (ALLOCATED(EALFA_GMV)) DEALLOCATE(EALFA_GMV)
IF (ALLOCATED(EALFA_GMVS)) DEALLOCATE(EALFA_GMVS)
IF (ALLOCATED(EALFA_GFL)) DEALLOCATE(EALFA_GFL)
IF (ALLOCATED(EALFA_TENC)) DEALLOCATE(EALFA_TENC)
IF (ALLOCATED(EALFAGT3GMV)) DEALLOCATE(EALFAGT3GMV)
IF (ALLOCATED(EALFAGT3GMVS)) DEALLOCATE(EALFAGT3GMVS)
IF (ALLOCATED(EALFAGT3GFL)) DEALLOCATE(EALFAGT3GFL)

IF (ALLOCATED(GMVCPL  )) DEALLOCATE(GMVCPL)
IF (ALLOCATED(GMVSCPL )) DEALLOCATE(GMVSCPL)
IF (ALLOCATED(GFLCPL  )) DEALLOCATE(GFLCPL)
IF (ALLOCATED(GMVSTENC)) DEALLOCATE(GMVSTENC)
IF (ALLOCATED(NLATGPP )) DEALLOCATE(NLATGPP)
IF (ALLOCATED(NLONGPP )) DEALLOCATE(NLONGPP)
IF (ALLOCATED(NIND_LIST)) DEALLOCATE(NIND_LIST)
IF (ALLOCATED(NIND_LEN)) DEALLOCATE(NIND_LEN)
IF (ALLOCATED(GMGT3   )) DEALLOCATE(GMGT3)
IF (ALLOCATED(GT3SPBUF)) DEALLOCATE(GT3SPBUF)
IF (ALLOCATED(LNUDSPGFL)) DEALLOCATE(LNUDSPGFL)

IF (ALLOCATED(MGMV0  )) DEALLOCATE(MGMV0)
IF (ALLOCATED(MGMV1  )) DEALLOCATE(MGMV1)
IF (ALLOCATED(MGMVS0  )) DEALLOCATE(MGMVS0)
IF (ALLOCATED(MGMVS1  )) DEALLOCATE(MGMVS1)

IF (ALLOCATED(CCFIELD_GMV)) DEALLOCATE(CCFIELD_GMV)
```

```
IF (ALLOCATED(CCFIELD_GMVS)) DEALLOCATE(CCFIELD_GMVS)
IF (ALLOCATED(CCFIELD_GFL)) DEALLOCATE(CCFIELD_GFL)

!     ---------------------------------------------------------------

IF (LHOOK) CALL
DR_HOOK('ELBC0B_MOD:DEALLOCATE_ELBC0B',1,ZHOOK_HANDLE)
END SUBROUTINE DEALLOCATE_ELBC0B

!
=================================================================================
======

END MODULE ELBC0B_MOD
```