NEW PRESENTATION FOR OBSHOR OBSERVATION INTERPOLATOR.

YESSAD Karim.

April 2, 2009

Version 1.

1 Introduction and purpose.

Observation horizontal interpolator is done under routines **COBS+COBSLAG**. Horizontal interpolations are done on upper air fields but also on some surface fields.

In March 2009, Ludovic Auger (CNRM/GMAP) pointed out difficulties to introduce new surface fields in this interpolator. Looking at the code, more potential problems have been raised and the purpose of this technical paper is to list these problems and to bring solutions to rewrite this piece of code more properly and to eliminate potential sources of bugs.

It is not planned in this study to adapt the new surface dataflow to GOMS (that means that individual treatment of surface variables will remain in GOMS dataflow).

2 Shortcomings with the current architecture of COBS+COBSLAG, and bugs or dirty features reported in CY35T2.

2.1 Shortcomings.

We just recall the organigramme under **SCAN2M**:

```
* COBS
* ADD_MODBIAS -> ADD_MODBIAS_TL
* SLEXTPOL
* COBSLAG -> OBSHOR ->
  - SLINT
  - MPOBSEQ_PACK
  - MPOBSEQ
```

Action of these routines:

- **COBS** fills buffer PB1 (like routines **LATTEX** and **LATTES** fill PB1 in the semi-Lagrangian scheme).
- Code under **COBSLAG** performs horizontal interpolations, with technical aspects very similar to code under **CALL_SL** in the semi-Lagrangian scheme.
- This piece of code has a tangent linear and adjoint one.

The main shortcomings of the current code presentation can be listed as follows:

- Pointers to fill PB1 are locally computed in **COBS**, and computed again in **COBSLAG**, contrary to what is done in the semi-Lagrangian scheme where pointers are in a module (**PTRSLB1**) and computed once in a set-up routine (**SUSLB**). There is a risk to have inconsistencies between **COBS** and **COBSLAG**, between the calculation of these pointers and the calculation of NFLDOBB1 done in **SUSC2B** (total number of fields to be interpolated).
- In **COBS** and **COBSLAG**, the different tests done to compute pointers (on LECMWF, forbidden at this level of the code, but also on keys like LCASIG, LDIRCLSMOD, LCANARI, LSOLV) may be not consistent with the ones used to allocate surface fields in routine **SU_SURF_FLDS**. There is a risk to interpolate not allocated surface fields.
- Both previous remarks can be extended to some other local quantities computed under **MPOBSEQ** and its callees (like calculation of IBUFLENP, or filling arrays YDGOM.., YGOM.. which explicitly redefines the order of storing data in local array ZBUFR), and also to some other local quantities computed under **MPOBSEQ_PACK** and its callees (like calculation of IBUFLENP, or filling arrays YDGOM.., YGOM.., or filling PBUFS with an explicit redefinition of the order of storing data in it). Conditions on LECMWF (forbidden at this level of the code), LCASIG, LDIRCLSMOD, LCANARI, LSOLV, also appear in these routines.
- Codes under **COBS** and **COBSLAG** do not take advantage of the new structure of surface fields: in particuliar, individual surface fields, but not groups of surface fields (ex: SOILB, RESVR, ...) appear in these routines, and that makes difficulties to introduce new surface fields in the interpolator. The new surface fields dataflow will in theory allow to do interpolations class by class (global treatment of each class, like we have a global treatment of GFL).

  The GOM dataflow present in the **MPOBSEQ..** routines does not take account at all of the **SU_SURF_FLDS** dataflow (no reference to groups of surface fields, order of storing fields completely different from the one defined by **SU_SURF_FLDS**).
- Additionally, it seems that there is a confusion between NSLWIDE (which allows to dimension the buffer halo for the semi-Lagrangian interpolator) and NOBWIDE (which allows to dimension the buffer halo for the observation interpolator). Routine **SLINT** must use NOBWIDE, and arrays computed (under **SLCSET** called by **SUSC2B**) with NOBWIDE, and the current use of NSLWIDE and of arrays based on NSLWIDE seems to be a provisional strategy in order to avoid to duplicate all the output dataflow of

SLCSET between "SL" and "OB" applications. In **SUSC2B**, the piece of code computing NSLWIDE and NOBWIDE is unclean and there is a severe risk to introduce bugs for some options.

Ideally, all the "OB" output dataflow of **SLCSET** must be well separated from the "SL" one; and calls to **SLCOMM2** and **(E)SLEXTPOL** done under **SLINT** must use only NOBWIDE and the "OB" output dataflow of **SLCSET**.

Routine **LASCAW** is called in the observation interpolator, and uses variables such as NSLEXT, LSLONDEM_ACTIVE which are typically semi-Lagrangian quantities. LSLONDEM concept has no sense in the observation interpolator (there is no semi-Lagrangian trajectory). Ideally, the version of **LASCAW** called by **SLINT** must work with the LSLONDEM=F code, no mask calculation, and a NSLEXT version computed with a halo width equal to NOBWIDE.

Using SLHD variables via modules is also an issue for this call to **LASCAW** because we don't necessary want to compute SLHD cubic weights if they are not required in the observation interpolator (case of a SL2TL run with SLHD SL interpolator, and no-SLHD observation interpolator).

Additional remarks:

- Same remarks are also valid for TL and AD codes.
- Work to solve the NOBWIDE issue can be done separately from the work to solve the other issues.
- Work to adapt the GOM dataflow in order to make it more consistent with the surface dataflow defined by **SU_SURF_FLDS** is out of the scope of this study, but this is one reason why new surface fields are not easy to introduce. The only thing we will try to do is to improve code presentation in order that memory transfers appear in the same order than variable allocations in **SU_SURF_FLDS** (but we don't change the order of storing data in the GOM themselves for the time being!).
- In routine **SLINT**, the second call to **LASCAW** is useless because option IWIS=203 computes both linear weights (required in parts 4 and 5 of **SLINT**) and cubic weights (required in part 3 of **SLINT**).

2.2 Reported bugs or dirty features in CY35T2.

This list is not comprehensive.

- In **COBS**: some lines have been commented about filling PB1 for group VX=VCLIX of surface variables (YPWP, YPWS, YTPC); but the corresponding interpolations have not been removed in **SLINT** (calls to **LAIDLIOBS** using PSXWS, PSXWP, PSXTP in part 5.10). Interpolated values are not stored in GOMS arrays.
- In **COBS**, PB1 is filled for field ORO (group VX=VCLIX of surface variables) but no interpolation is done in **SLINT** for this variable; interpolated values are not stored in GOMS arrays.
- In **COBS**, the following lines (under LFGEL=T) seem bugged:

```
PB1(IROF,IOBB1SAB) = SP_RR(JROF,YSP_RR%YRR(4)%MP0,IBLK)
PB1(IROF,IOBB1SAB) = ZSP_RR5(JROF,YSP_RR%YRR(4)%MP5)
```

  At least YSP_RR%YRR(4) must be replaced by YSP_RR%YIC; use of pointer IOBB1SAB is a bug; this field does not seem to be interpolated in **SLINT**; interpolated value is not stored in GOM arrays.
- In **COBS**, PB1(.,IOBB1WL) is set to 1 everywhere: why interpolate this horizontally constant quantity? What is the physical meaning of "Wl" (meaningful comment to be added in the code)?

3 Proposals to improve consistency with the new surface dataflow and to move all relevant calculations in set-up routines.

3.1 Kinds of interpolated variables, and relevant classes of surface variables for observation interpolator.

There are 6 kinds of interpolated variables:

- GMV upper-air variables (wind, temperature): does not take account of NH variables for the time being.
- GFL variables (for example specific humidity).
- GMVS variables (surface pressure).
- Constants like orography or **YOMGC** constants (OROG = orography, RCORI = Coriolis parameter, GNORDL+GNORDM = components of the unit vector directed towards the geographic Northern pole).
- Surface variables.
- Model errors statistics (CANARI optimal interpolation, code under key LCASIG).

More details about interpolated surface variables:

- Group SB=SOILB:
  - YQ (EC; liquid water content).
- Group SG=SNOWG:
  - YF (content of surface snow).
- Group RR=RESVR:
  - YT (skin temperature (Ts)).
  - YW (MF; superficial reservoir water content (Ws) at MF).
  - YRR(4)=YIC (superficial reservoir ice).
- Group CL=CLS:
  - YTCLS (MF; 2m temperature).
  - YHUCLS (MF; 2m humidity).
- Group WS=WAVES: none.
- Group WW=WAM: none.
- Group EP=EXTRP: none.
- Group X2=XTRP2:
  - YX2(1), YX2(2), YX2(3) and YX2(4) (MF; precipitation fields in CANARI). But what is interpolated is YX2(1)+YX2(2) and YX2(3)+YX2(4)
- Group CI=CANRI: none.
- Group VF=VARSF:
  - YZ0F (gravity $*$ surface roughness length).
  - YALBF (surface shortwave albedo).
  - YEMISF (surface longwave emissivity).
  - YITM (land-sea mask).
  - YVEG (MF; vegetation cover).
  - YCVL (EC; low vegetation cover).
  - YCVH (EC; high vegetation cover).
  - YTVL (EC; low vegetation type).
  - YTVH (EC; high vegetation type).
  - YCI (EC; sea ice fraction).
- Group VP=VCLIP: none.
- Group VV=VCLIV:
  - YARG (silt percentage within soil).
  - YSAB (percentage of sand within the soil).
  - YHV (resistance to evapotranspiration).
  - YZ0H (gravity $*$ roughness length for heat).
- Group VN=VCLIN: none.
- Group VH=VCLIH: none.
- Group VA=VCLIA: none.
- Group VG=VCLIG: none.
- Group VC=VO3ABC: none.
- Group VD=VDIAG: none.
- Group VX=VCLIX:
  - YORO (MF; climatological surface geopotential).
  - YTSC (MF; climatological surface temperature).
  - YSNO (MF; climatological snow cover).
- Group XA=VEXTRA: none.
- Group X2=VEXTR2: none.

Interpolated surface fields are spread among 8 groups.

3.2 New attributes required for surface dataflow.

We need the following additional attributes in the definition of types TYPE_SURF_MTL_3D and TYPE_SURF_MTL_2D:

- LOBSHOR: T vs F: this field is interpolated vs not interpolated in the observation horizontal interpolator.
- CSLINT: interpolator used by surface fields.

Example: YSP_SB%YQ%LOBSHOR=T means that liquid water content (in group SB=SOILB) will be interpolated; YSP_SB%YQ%CSLINT='LAIDLIOBS ' means that interpolator **LAIDLIOBS** will be used for this variable.
The value of these attributes must be computed during the call to SETUP_SFLP3 or SETUP_SFLP2 done under SU_SURF_FLDS. Setting attribute LOBSHOR to .T. requires at least the condition:

```
LOBSC1 .OR. NCONF/100 == 1 .OR. NCONF==701
```

and some conditions on LCANARI, LOBSREF, LECMWF, LSOLV, LFGEL, LCASIG (which are currently spread among several routines and maybe not always consistent). That means that routines SETUP_SFLP3 and SETUP_SFLP2 (in module surface_fields_mix.F90) must be adapted: both routines must contain the additional optional input dummy argument LDOBSHOR, to be able to fill attribute LOBSHOR. In the revised code, conditions on LCANARI, LOBSREF... must appear only in SU_SURF_FLDS to pass .T. or .F. to LDOBSHOR. If the optional argument LDOBSHOR is not present in a call to SETUP_SFLP3 or SETUP_SFLP2, that will be equivalent to have this argument set to .F. .

Example: in class RR=RESVR, we want to specify that, in some conditions, we want to interpolate the superficial reservoir ice (YIC) in the observation interpolator (interpolator=**LAIDLIOBS**). In SU_SURF_FLDS we can see (CY35T2) that this field is allocated only if the following condition is fulfilled:

```
(.NOT.LECMWF).AND.LMPHYS.AND.(.NOT.LLASSIMAROME).AND.((LSOLV.AND.LFGEL).OR.LMSE)
```

Additionally, this field will be interpolated if the following necessary condition (but is it sufficient?) is fulfilled:

```
LOBSC1 .OR. NCONF/100 == 1 .OR. NCONF==701 = T
```

That means that the sequence of lines:

```
IF ((LSOLV.AND.LFGEL).OR.LMSE) THEN
  YSP_RR%YIC => YSP_RR%YRR(YSP_RRD%IPTR)
  CALL SETUP_SFLP2(YSP_RRD,YSP_RR%YIC,CDNAME='SURFRESERV.GLACE', KTRAJ=1, KREQIN=1)
ENDIF
```

will be replaced by something looking like:

```
IF ((LSOLV.AND.LFGEL).OR.LMSE) THEN
  YSP_RR%YIC => YSP_RR%YRR(YSP_RRD%IPTR)
  LLOBSHOR=LOBSC1 .OR. NCONF/100 == 1 .OR. NCONF==701
  CALL SETUP_SFLP2(YSP_RRD,YSP_RR%YIC,CDNAME='SURFRESERV.GLACE', KTRAJ=1, &
   & KREQIN=1, LDOBSHOR=LLOBSHOR, CDSLINT='LAIDLIOBS ')
ENDIF
```

3.3 PTROBB1 pointers and calculation of NFLDOBB1.

It is recommended to replace local pointers IOBB1U, .., IOBB1Z0, .., computed in **COBS** and **COBSLAG** by module pointers which will be computed by a set-up routine called by **SUSC2B**. What is currently done for the **PTRSLB1** pointers of the semi-Lagrangian scheme can be extended to the observation horizontal interpolator pointers. These pointers will be stored and declared in a new deck ptrobb1.F90 and set-up in a new set-up routine **SUOBB** called under **SUSC2B** (at the same location as **SUSLB**). The content of **SUOBB** will look like the one of **SUSLB**.
SUOBB will be called in the following cases:

- Configuration 701.
- Configurations between 101 and 199 (131 for the time being).
- Other configurations: if LOBSC1=T.

In **PTROBB1** we need at least the following pointers:

- MOBBUF1.
- For GMV variables: MOBB1U (3D field), MOBB1V (3D field), MOBB1T (3D field).
- For GFL variables: MOBB1GFL (NGOMGFL 3D fields).
- For GMVS variables: MOBB1SP.
- For constants like orography or **YOMGC** constants: MOBB1OR, MOBB1CO, MOBB1DL, MOBB1DM.

- For surface variables, pointers for each group where there are variables to interpolate: MOBB1SB, MOBB1SG, MOBB1RR, MOBB1CL, MOBB1X2, MOBB1VF, MOBB1VV, MOBB1VX.

  Pointers for individual fields will also be computed (provisionally required when storing data in GOMS and for some other applications).

  - Group SB=SOILB: MOBB1SB_Q.
  - Group SG=SNOWG: MOBB1SG_F.
  - Group RR=RESVR: MOBB1RR_T,MOBB1RR_W,MOBB1RR_IC.
  - Group CL=CLS: MOBB1CL_TCLS, MOBB1CL_HUCLS.
  - Group X2=XTRP2: MOBB1X2_PRWA, MOBB1X2_PRSN.
  - Group VF=VARSF: MOBB1VF_Z0F, MOBB1VF_ALBF, MOBB1VF_EMISF, MOBB1VF_LSM, MOBB1VF_VEG, MOBB1VF_CVL, MOBB1VF_CVH, MOBB1VF_TVL, MOBB1VF_TVH, MOBB1VF_CI.
  - Group VV=VCLIV: MOBB1VV_ARG, MOBB1VV_SAB, MOBB1VV_HV, MOBB1VV_Z0H.
  - Group VX=VCLIX: MOBB1VX_ORO, MOBB1VX_TSC, MOBB1VX_SNO.

- Pointer MOBB1WL.
- Pointers for model errors statistics: MOBB1EZ (3D field), MOBB1EH (3D field), MOBB1EST, MOBB1ESN, MOBB1ET2, MOBB1EH2, MOBB1EV1.

Calculation of NFLDOBB1 will be also done in **SUOBB**, consistently with the calculation of pointers. NFLDOBB1=0 in **SUSC2B** if **SUOBB** is not called.

PTROBB1 must also contain array RPAROB1 for parities over poles (calculation to be done in **SUOBB**): replaces the local array ZPARITY computed in **SLINT** and **SCAN2M**.

The structure of new routine **SUOBB** will look like the one of **SUSLB**:

- Part 1: set **PTROBB1** pointers to NUNDEFLD (order must be the previous one).
- Part 2: compute values of these pointers and print them.
- Part 3: compute parities for extension over pole.

Appendix A gives a prototype for routine **SUOBB**.

3.4 Use of these pre-computed quantities for interpolations, and interpolating surface fields class by class.

Revision of the set-up routines, as mentioned above, will allow a significant simplification of the code under **COBS** and **COBSLAG**.

∗ COBS:

- In part 2.3 add LL5=.NOT.(LOBSC1.OR.LOBSREF.OR.LCANARI)
- In part 3.0, it is desirable (and simpler) to fill the whole PB1 with zeros, rather than a subset of PB1 with 999999999.0 .
- In part 3.1.1, 3.1.2, beginning of part 3.1.3 (constants), and part 3.1.4, local pointers IOBB1.. must be replaced by MOBB1.. .
- Part 3.1.3 for surface fields must be rewritten as follows: group by group, but no comprehensive list of fields must appear inside each group. For example, for group RR=RESVR, we must find the following lines:

```
IF (LL5) THEN
  IFLDRR=0
  DO JFLD=1,YSP_RRD%NUMFLDS
    IF (YSP_RR%YRR(JFLD)%LOBSHOR) THEN
      IFLDRR=IFLDRR+1
!OCL NOVREC
      DO JROF=IST,IEND
        IROF=ISLCORE(JROF)
        PB1(IROF,MOBB1RR+IFLDRR-1)=ZSP_RR5(JROF,YSP_RR%YRR(JFLD)%MP5,IBLK)
      ENDDO
    ENDIF
  ENDDO
ELSE
  IFLDRR=0
  DO JFLD=1,YSP_RRD%NUMFLDS
    IF (YSP_RR%YRR(JFLD)%LOBSHOR) THEN
      IFLDRR=IFLDRR+1
!OCL NOVREC
      DO JROF=IST,IEND
```

```
          IROF=ISLCORE(JROF)
          PB1(IROF,MOBB1RR+IFLDRR-1)=SP_RR(JROF,YSP_RR%YRR(JFLD)%MP0,IBLK)
        ENDDO
      ENDIF
    ENDDO
  ENDIF
```

We can notice some code simplification: no test left on LCANARI, LECMWF, etc.
Caution: for class VX=VCLIX, nothing is done if LL5=T.
A difficulty may be expected with group X2=XTRP2 because fields are gathered by pairs, and interpolations are not done for trajectory.

```
IF (.NOT.LL5) THEN
  IFLDX2=0
  DO JFLD=1,YSP_X2D%NUMFLDS,2
    IF (YSP_X2%YX2(JFLD)%LOBSHOR.AND.YSP_X2%YX2(JFLD+1)%LOBSHOR) THEN
      IFLDX2=IFLDX2+1
!OCL NOVREC
      DO JROF=IST,IEND
        IROF=ISLCORE(JROF)
        PB1(IROF,MOBB1X2+IFLDX2-1)=SP_X2(JROF,YSP_X2%YX2(JFLD)%MP0,IBLK)+&
         & SP_X2(JROF,YSP_X2%YX2(JFLD+1)%MP0,IBLK)
      ENDDO
    ENDIF
  ENDDO
ENDIF
```

- Part 3.1.3 must end by filling PB1(.,MOBB1WL).
- Part 2.2 for local pointers becomes useless (to be removed).

∗ COBSLAG:

- Part 1 must be removed.
- Call to **OBSHOR** must be simplified, pass the whole PB1 once (not each field individually).

∗ OBSHOR:

- Dummy argument interface must be simplified, pass the whole POBB1 once (not each field individually).
- Call to **SLINT** must be simplified, pass the whole POBB1 once (not each field individually). Use an array ZOBBF to pass most of the interpolated surface fields (instead of the individual ZSL.. arrays).
- Pass ZOBBF to **MPOBSEQ_PACK** instead of the ZSL.. arrays. ZSLSST must be kept and renamed into ZOBSST.

∗ **SLINT:** This routine does interpolations: appendix C provides a list of variables which are interpolated and the paragraph where interpolation is done in CY35T2 for each quantity.

Work to be done:

- Dummy argument interface must be simplified, pass the whole POBB1 once (not each field individually).
- Use POBB1 for quantities to be interpolated (with the right pointer of **PTROBB1**).
- Individual arrays storing interpolated quantities must be replaced by a global one (POBBF), dimensioned with NFLDOBB1. This array must be passed to interpolation routines (with some exceptions), with the right pointer of **PTROBB1**.
- The second call to **(E)LASCAW** is useless (computes linear weights already computed by the first one).
- The revised version of **SLINT** must contain:
  - the current contents of parts 1 and 2.
  - horizontal interpolations with weights which do not use land-sea mask (current part 3 and parts 4.2 to 4.4).
  - horizontal interpolations with weights which use land-sea mask (current part 5). For surface variables groups must be treated globally as possible (no reference to individual fields inside each groupe when it can be avoided). This part starts by a recalculation of weights.
- Fill an additional output dummy argument LDINT (dimension NFLDOBB1) saying that fields have been interpolated if .T., and also LDINT_SST (see Appendix B).

Recommended order is, for interpolations with weights which do not use land-sea mask.

- GMV variables: U, V, T (current parts 3.1, 3.2 and 3.3). Choice between **LAIDLIOBS** or **LAIDDIOBS** according to the value of IWIS.

- GFL variables (current parts 3.4 and 4.4). Choice of interpolator according to the value of IWIS and YGOMGFL(JGFL)%CSLINT.

```
(YGOMGFL(JGFL)%CSLINT == 'STANDARD  ').AND.(IWIS == 201) -> LAIDLIOBS
(YGOMGFL(JGFL)%CSLINT == 'STANDARD  ').AND.(IWIS == 203) -> LAIDDIOBS
(YGOMGFL(JGFL)%CSLINT == 'LAIDLIC   ') -> LAIDLIC
```

- GMVS variables (current part 3.6). Choice between **LAIDLIOBS** or **LAIDDIOBS** according to the value of IWIS.
- Constants (current parts 3.7, 4.3, 3.8). Choice between **LAIDLIOBS** or **LAIDDIOBS** according to the value of IWIS, excepted for Coriolis parameter where only **LAIDLIOBS** is allowed. Order: orography, Coriolis parameter, geographical north direction.
- The two 3D model error statistics fields (roots EZ and EH, current part 4.2). Call to **LAIDLIOBS**.

Recommended order is, for interpolations with weights which use land-sea mask.

- Class of surface fields SB=SOILB: call to **LAIDLIOBS**.
- Class of surface fields SG=SNOWG: call to **LAIDLIOBS**.
- Class of surface fields RR=RESVR: call to **LAIDLIOBS**.
- Class of surface fields CL=CLS: call to **LAIDLIOBS**.
- Class of surface fields X2=XTRP2: call to **LAIDLIOBS**.
- Class of surface fields VF=VARSF: call to **LAIDLIOBS** or **LAIDLIC** according to the field.
- Class of surface fields VV=VCLIV: call to **LAIDLIOBS**.
- Class of surface fields VX=VCLIX: call to **LAIDLIOBS**.
- The additional field with pointer MOBB1WL: call to **LAIDLIOBS**.
- The five 2D model error statistics fields (roots EST, ESN, ET2, EH2, EV1; cf. current part 5.12): call to **LAIDLIOBS**.
- Revision of weights and additional interpolations and calculations (cf. current parts 5.13 and 5.14).

To give an example how the code must look like for surface fields, see Appendix B.

∗ **MPOBSEQ_PACK:** This routine copies interpolated values in the different GOM buffers. See appendix D for more details about quantities copied (and order of copy) in the CY35T2 version of **MPOBSEQ_PACK**.

- Dummy argument interface must be simplified, pass the whole POBBF once (not each PSL.. field individually). Pass also POBSST (= current PSLSST).
- Calculation of IBUFLENP must be revised and simplified (this is the number of 2D interpolated fields which must be stored in the GOM arrays): this is probably NFLDOBB1+5 to take account of the NFLDOBB1 fields present in POBBF, of the additional interpolated field stored in POBSST, and of the additional storage of surface values of GMV variables (U, V, T) and specific humidity. No test on LECMWF, LCANARI, etc, must be used any longer to compute IBUFLENP.
- Presentation of part 3 must be improved in order to better show the different classes of variables (GMV, GFL, GMVS, constants, surface, model error statistics). For surface variables we cannot avoid for the time being individual treatment of each variable (because the GOMS dataflow is not ready for treatment group by group): do the copy variable by variable.
  - Part 3.1: copy interpolated GMV and GFL variables in YDGOMUA or YDGOMUA_2D according to the value of IPROF.
  - Part 3.2: copy relevant interpolated quantities in YDGOMS or YDGOMS_2D according to the value of IPROF. Order of copy must be:
    - interpolated bottom level values of GMV.
    - interpolated bottom level values of GFL (humidity only for the time being).
    - interpolated GMVS variables.
    - surface variables of group SB=SOILB (YQ).
    - surface variables of group SG=SNOWG (YF).
    - surface variables of group RR=RESVR (YT, YW, YIC).
    - surface variables of group CL=CLS (YTCLS, YHUCLS).
    - a subset of surface variables of group VF=VARSF (YZ0F).
    - content of POBSST.
    - additional field "WL" (cf. pointer MOBB1WL).
  - Part 3.3: copy relevant interpolated quantities in YGOMEC or YGOMEC_2D according to the value of IPROF.

        * interpolated constants (OR, CO, DL, DM).
        * a subset of surface variables of group VF=VARSF (YALBF, YEMISF, YITM, YVEG, YCVL, YCVH, YTVL, YTVH, YCI).
        * surface variables of group VV=VCLIV (YARG, YSAB, YHV, YZ0H).
    - Part 3.4: copy relevant interpolated quantities in YGOMCAN (2D quantities) or YGOMCANA (3D quantities).
        * surface variables of group X2=XTRP2 (PRWA, PRSN).
        * surface variables of group VX=VCLIX (YTSC, YSNO).
        * model error statistics (EZ, EH, EST, ESN, ET2, EH2, EV1).

  Memory transfers in the GOM arrays must use the content of array LDINT (and of variable LDINT_SST), computed in **SLINT**, and passed to **MPOBSEQ_PACK**, to say if memory transfers must be done or not. No test on LCANARI, LSOLV, LECMWF, LCASIG, etc, must remain in this routine (because of risks of inconsistencies with tests done in **SU_SURF_FLDS**.

- The same kind of work must also be done when filling PBUFS in subroutine **PACK_SEND** (copy interpolated values in buffer SENDBUF5 or SENDBUF). Individial specification of surface fields cannot be avoided but order of storing fields in SENDBUF5 or SENDBUF must be consistent with the one in POBBF (for surface fields the order must be the same one as the order of allocation in **SU_SURF_FLDS** and this is not the case in CY35T2).

∗ **MPOBSEQ:** This routine must be updated according to the modifications introduced in **MPOBSEQ_PACK** in order to have the right storage of variables in the different GOM arrays (more exactly, order of data storage in ZBUFR must be consistent with the one of PBUFS under **MPOBSEQ_PACK**). In particuliar, subroutine **UNPACK_RECV** (contained in **MPOBSEQ**) must be the mirror of subroutine **PACK_SEND** (contained in **MPOBSEQ_PACK**).

∗ **ADD_MODBIAS_TL:** No modification expected in this routine, but it seems that this routine assumes that first 3D fields, then 2D fields, are stored in PB1 (POBB1): in practical this is the case for non-CANARI applications, but not for CANARI applications where 3D model error statistics are stored after some 2D fields. It would be recommended to forbid case LMODERR.AND.NTYPE_MODERR==3.AND.LCANARI.

3.5 TL code.

We just recall the organigramme under **SCAN2MTL**:

```
* COBS
* COBSTL
* ADD_MODBIAS_TL
* SLEXTPOL
* COBSLAG -> OBSHOR ->
  - SLINT
  - MPOBSEQ_PACK
  - MPOBSEQ
```

This code re-uses most of the direct code, only adaptations must be done in **SCAN2MTL** (replace ZPARITY by RPAROB1) and **COBSTL** (local pointers to be replaced by **PTROBB1** ones).

3.6 AD code.

We just recall the organigramme under **SCAN2MAD**:

```
* COBSLAGAD -> OBSHORAD ->
  - MPOBSEQAD
  - MPOBSEQAD_UNPACK
  - SLINTAD
* SLEXTPOLAD
* ADD_MODBIAS_AD
* COBSAD
```

No variable other than GMV, GFL, GMVS, has been currently implemented in the interpolator: only pointers IOBB1.. must be replaced by MOBB1... pointers.

Some constants (DL, DM) and surface variables (in arrays PSLTS, PSLWS, PSLSN, PSLZ0, PSLWL) appear in **MPOBSEQAD_UNPCK** and **MPOBSEQAD**, but with some specific conditions (no use of keys like LECMWF, LCANARI, ..). It is probable that these routines can remain unchanged for the time being.

4 Solving the NOBWIDE issue.

A separate **SLCSET** output dataflow must be used for observation interpolator. Additionally, no LSLONDEM code must be used (that leads to a code simplification).

4.1 SUSC2B.

The following quantities must be added in module **YOMMP**: NOBPROCS, NOBRPT, NOBSPT, NOBSENDPOS, NOBRECVPOS, NOBSENDPTR, NOBRECVPTR, NOBCOMM, NOBEXT, NOBCORE, NAOBB1.

In **SUSC2B**, the call to **SLCSET** using 'OB' and NOBWIDE as input dummy arguments must compute not only NOBSTA, NOBONL, NOBOFF, but also the new NOB.. and NAOBB1 above additional quantities.

When the observation horizontal interpolator is required, NOBWIDE should be set at least at 2.

4.2 SLINT.

Some modifications must be brought to **LASCAW** and **ELASCAW**: variables LSLHD_OLD, LSLONDEM_ACTIVE, NSLEXT, must be passed via dummy arguments.

In **SLINT**:

- The call to **LASCAW** and **ELASCAW** must use: LLSLHD_OLD=.FALSE., LLSLONDEM_ACTIVE=.FALSE., NOBEXT.
- NSLWIDE must be replaced by NOBWIDE.
- NOBOFF, NAOBB1, NOBSTA, NOBPROCS, NOBRPT, NOBSPT, NOBSENDPOS, NOBRECVPOS, NOBSENDPTR, NOBRECVPTR, NOBCOMM, NOBONL must be used instead of NSLOFF ,NFRSTLOFF, NASLB1, NSLSTA, NSLPROCS, NSLRPT, NSLSPT, NSLSENDPOS, NSLRECVPOS, NSENDPTR, NRECVPTR, NSLCOMM, NSLONL.
- Masks MASK_SL1 and MASK_SL2 are not required.
- calls to **SLCOMM2**, **SLEXTPOL** and **ESLEXTPOL** must be removed.

4.3 COBS.

NSLCORE ,NASLB1 must be replaced by NOBCORE, NAOBB1.

4.4 ADD_MODBIAS.

NASLB1 must be replaced by NAOBB1.

4.5 SCAN2M.

In part 4.2:

- remove lines modifying LSLONDEM_ACTIVE.
- remove lines allocating MASK_SL2.
- calls to **SLEXTPOL** and **ESLEXTPOL** must use NOBWIDE, NAOBB1, NOBSTA, NOBONL, NOBOFF, no mask, KTYP=1 (instead of 3); a call to **SLCOMM** with KTYP=0 must be added just before the calls to **SLEXTPOL** and **ESLEXTPOL**, if NPROC>1. Finally, the code of part 4.2 must look like:

```
!*       4.2 DISTRIBUTED MEMORY : COMMUNICATION BETWEEN PROCESSORS

  IF (CDCONF(6:6)  ==  'C'.OR.LOBS) THEN

    CALL GSTATS(8,2)

    IF (NPROC > 1) THEN
      CALL SLCOMM(NAOBB1,NOBPROCS,NOBRPT,&
       & NOBSPT,NOBSENDPOS,NOBRECVPOS,NOBSENDPTR,NOBRECVPTR,NOBCOMM,&
       & IDUMARR,NFLDOBB1,LLINC,0,ZOBBUF1)
    ENDIF

    IF (.NOT.LELAM) THEN
      CALL SLEXTPOL(NOBWIDE,NOBWIDE,&
       & NAOBB1,NDGSAG,NDGENG,NDGSAL,NDGENL,NFLDOBB1,NOBSTA,NOBONL,NOBOFF,&
       & NDGLG,NLOENG,NFRSTLOFF,MYFRSTACTLAT,MYLSTACTLAT,IDUMARR,1,&
       & RPAROB1,ZOBBUF1)
    ELSE
```

```
      CALL ESLEXTPOL(NOBWIDE,NOBWIDE,&
       & NAOBB1,NDGSAG,NDGENG,NDGSAL,NDGENL,NFLDOBB1,NOBSTA,NOBONL,NOBOFF,&
       & NLOENG,NFRSTLOFF,MYFRSTACTLAT,MYLSTACTLAT,IDUMARR,1,&
       & ZOBBUF1)
    ENDIF
  ENDIF
```

- call to COBSLAG must use NAOBB1 instead of NASLB1.

4.6 TL and AD code.

The same kind of modifications must be also done on TL and AD codes.

5 Conclusion and perspectives.

Work has been undertaken to rewrite the observation horizontal interpolator in order to ensure better consistency with the surface dataflow and to avoid redefining the same quantities (tests, pointers) in different parts of the code. We have identified at least two distinct tasks.

The first task allows to set-up some pointers (instead of recomputing them at several places of the code), to reduce the number of dummy arguments of some routines, and to ensure consistency between the surface dataflow of the interpolator and the surface dataflow defined by **SU_SURF_FLDS**. This is the most stringent task to be done in order to allow easier introduction of new surface fields (or evolution of use of the current surface fields) if required in the future.
It would be recommended to give a high priority to this work (target = CY36T1?).

The second task allows to do the halo calculation of the interpolation buffer with a OB-own dataflow which does not interfere with the semi-Lagrangian interpolator one. In particuliar it removes the notion of "on-demand" comminications which has no meaning in the observation interpolator. In semi-Lagrangian runs that may lead to memory saving (since NOBWIDE is significantly lower than NSLWIDE in this case). The revised code looks simpler (DM communications in **SCAN2M** only, not in **SLINT**).
This task is less stringent than the previous one.

Adapting the GOM dataflow was out of the scope of our study. But the GOM dataflow still has a fixed set of surface fields which does not take account of the flexibility of the new surface dataflow defined by **SU_SURF_FLDS**. Introducing new surface variables in the GOM arrays is not convenient with the current dataflow. Adapting the GOM dataflow (for example introducing splitting between the different groups SB=SOILB, RR=RESVR, etc) is a task which may have collateral effects on some other parts of the code which also use a fixed set of surface variables (CANARI for example).

Appendix A: Body of new routine SUOBB.

```
LLP = .TRUE.

!      Part 1: set pointers to NUNDEFLD.

MOBBUF1=NUNDEFLD

! * GMV:
MOBB1U=NUNDEFLD
MOBB1V=NUNDEFLD
MOBB1T=NUNDEFLD

! * GFL:
MOBB1GFL=NUNDEFLD

! * GMVS:
MOBB1SP=NUNDEFLD

! * Constants:
MOBB1OR=NUNDEFLD
MOBB1CO=NUNDEFLD
MOBB1DL=NUNDEFLD
MOBB1DM=NUNDEFLD

! * Surface variables (group SB=SOILB):
MOBB1SB=NUNDEFLD
MOBB1SB_Q=NUNDEFLD

! * Surface variables (group SG=SNOWG):
MOBB1SG=NUNDEFLD
MOBB1SG_F=NUNDEFLD

! * Surface variables (group RR=RESVR):
MOBB1RR=NUNDEFLD
MOBB1RR_T=NUNDEFLD
MOBB1RR_W=NUNDEFLD
MOBB1RR_IC=NUNDEFLD

! * Surface variables (group CL=CLS):
MOBB1CL=NUNDEFLD
MOBB1CL_TCLS=NUNDEFLD
MOBB1CL_HUCLS=NUNDEFLD

! * Surface variables (group X2=XTRP2):
MOBB1X2=NUNDEFLD
MOBB1X2_PRWA=NUNDEFLD
MOBB1X2_PRSN=NUNDEFLD

! * Surface variables (group VF=VARSF):
MOBB1VF=NUNDEFLD
MOBB1VF_ZOF=NUNDEFLD
MOBB1VF_ALBF=NUNDEFLD
MOBB1VF_EMISF=NUNDEFLD
MOBB1VF_LSM=NUNDEFLD
MOBB1VF_VEG=NUNDEFLD
MOBB1VF_CVL=NUNDEFLD
MOBB1VF_CVH=NUNDEFLD
MOBB1VF_TVL=NUNDEFLD
MOBB1VF_TVH=NUNDEFLD
MOBB1VF_CI=NUNDEFLD

! * Surface variables (group VV=VCLIV):
MOBB1VV=NUNDEFLD
MOBB1VV_ARG=NUNDEFLD
MOBB1VV_SAB=NUNDEFLD
MOBB1VV_HV=NUNDEFLD
MOBB1VV_ZOH=NUNDEFLD

! * Surface variables (group VX=VCLIX):
MOBB1VX=NUNDEFLD
MOBB1VX_ORO=NUNDEFLD
MOBB1VX_TSC=NUNDEFLD
MOBB1VX_SNO=NUNDEFLD

! * Additional surface variables:
MOBB1WL=NUNDEFLD

! * Model errors statistics:
MOBB1EZ=NUNDEFLD
MOBB1EH=NUNDEFLD
MOBB1EST=NUNDEFLD
MOBB1ESN=NUNDEFLD
MOBB1ET2=NUNDEFLD
MOBB1EH2=NUNDEFLD
MOBB1EV1=NUNDEFLD

!      Part 2: compute pointers and print them.

IF (LLP) WRITE (NULOUT,'(A)') ' Pointers of PTROBB1:'
IPTX    = 1
```

```
MOBBUF1 = IPTX

! * GMV:

MOBB1U=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1U=     ',IPTX,NFLEVG
IPTX=IPTX+NFLEVG
MOBB1V=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1V=     ',IPTX,NFLEVG
IPTX=IPTX+NFLEVG
MOBB1T=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1T=     ',IPTX,NFLEVG
IPTX=IPTX+NFLEVG

! * GFL:

MOBB1GFL=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1GFL=   ',IPTX,NFLEVG*NGOMGFL
IPTX=IPTX+NFLEVG*NGOMGFL

! * GMVS:

MOBB1SP=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1SP=    ',IPTX,1
IPTX=IPTX+1

! * Constants:

MOBB1OR=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1OR=    ',IPTX,1
IPTX=IPTX+1
MOBB1CO=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1CO=    ',IPTX,1
IPTX=IPTX+1
MOBB1DL=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1DL=    ',IPTX,1
IPTX=IPTX+1
MOBB1DM=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1DM=    ',IPTX,1
IPTX=IPTX+1

! * Surface variables:

! Group SB=SOILB:
IFLDSB=0
IFLDSB_Q=0
DO JFLD=1,YSP_SBD%NUMFLDS
  IF (YSP_SB%YSB(JFLD)%LOBSHOR) THEN
    IFLDSB=IFLDSB+1
    IF (YSP_SB%YSB(JFLD) == YSP_SB%YQ) IFLDSB_Q=IFLDSB
  ENDIF
ENDDO
IF (IFLDSB>0) THEN
  MOBB1SB=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1SB=    ',IPTX,IFLDSB
  IPTX=IPTX+IFLDSB
  IF(IFLDSB_Q > 0) THEN
    MOBB1SB_Q=MOBB1SB+IFLDSB_Q-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1SB_Q=  ',MOBB1SB_Q,1
  ENDIF
ENDIF

! Group SG=SNOWG:
IFLDSG=0
IFLDSG_F=0
DO JFLD=1,YSP_SGD%NUMFLDS
  IF (YSP_SG%YSG(JFLD)%LOBSHOR) THEN
    IFLDSG=IFLDSG+1
    IF (YSP_SG%YSG(JFLD) == YSP_SG%YF) IFLDSG_F=IFLDSG
  ENDIF
ENDDO
IF (IFLDSG>0) THEN
  MOBB1SG=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1SG=    ',IPTX,IFLDSG
  IPTX=IPTX+IFLDSG
  IF(IFLDSG_F > 0) THEN
    MOBB1SG_F=MOBB1SG+IFLDSG_F-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1SG_F=  ',MOBB1SG_F,1
  ENDIF
ENDIF

! Group RR=RESVR:
IFLDRR=0
IFLDRR_T=0
IFLDRR_W=0
IFLDRR_IC=0
DO JFLD=1,YSP_RRD%NUMFLDS
  IF (YSP_RR%YRR(JFLD)%LOBSHOR) THEN
    IFLDRR=IFLDRR+1
    IF (YSP_RR%YRR(JFLD) == YSP_RR%YT) IFLDRR_T=IFLDRR
    IF (YSP_RR%YRR(JFLD) == YSP_RR%YW) IFLDRR_W=IFLDRR
```

```
    IF (YSP_RR%YRR(JFLD) == YSP_RR%YIC) IFLDRR_IC=IFLDRR
  ENDIF
ENDDO
IF (IFLDRR>0) THEN
  MOBB1RR=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1RR=    ',IPTX,IFLDRR
  IPTX=IPTX+IFLDRR
  IF(IFLDRR_T > 0) THEN
    MOBB1RR_T=MOBB1RR+IFLDRR_T-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1RR_T=  ',MOBB1RR_T,1
  ENDIF
  IF(IFLDRR_W > 0) THEN
    MOBB1RR_W=MOBB1RR+IFLDRR_W-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1RR_W=  ',MOBB1RR_W,1
  ENDIF
  IF(IFLDRR_IC > 0) THEN
    MOBB1RR_IC=MOBB1RR+IFLDRR_IC-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1RR_IC=  ',MOBB1RR_IC,1
  ENDIF
ENDIF

! Group CL=CLS:
IFLDCL=0
IFLDCL_TCLS=0
IFLDCL_HUCLS=0
DO JFLD=1,YSP_CLD%NUMFLDS
  IF (YSP_CL%YCL(JFLD)%LOBSHOR) THEN
    IFLDCL=IFLDCL+1
    IF (YSP_CL%YCL(JFLD) == YSP_CL%YTCLS) IFLDCL_TCLS=IFLDCL
    IF (YSP_CL%YCL(JFLD) == YSP_CL%YHUCLS) IFLDCL_HUCLS=IFLDCL
  ENDIF
ENDDO
IF (IFLDCL>0) THEN
  MOBB1CL=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1CL=    ',IPTX,IFLDCL
  IPTX=IPTX+IFLDCL
  IF(IFLDCL_TCLS > 0) THEN
    MOBB1CL_TCLS=MOBB1CL+IFLDCL_TCLS-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1CL_TCLS=  ',MOBB1CL_TCLS,1
  ENDIF
  IF(IFLDCL_HUCLS > 0) THEN
    MOBB1CL_HUCLS=MOBB1CL+IFLDCL_HUCLS-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1CL_HUCLS=  ',MOBB1CL_HUCLS,1
  ENDIF
ENDIF

! Group X2=XTRP2:
IFLDX2=0
IFLDX2_PRWA=0
IFLDX2_PRSN=0
DO JFLD=1,YSP_X2D%NUMFLDS,2
  IF (YSP_X2%YX2(JFLD)%LOBSHOR .AND. YSP_X2%YX2(JFLD+1)%LOBSHOR) THEN
    IFLDX2=IFLDX2+1
    IF (JFLD == 1) IFLDX2_PRWA=IFLDX2
    IF (JFLD == 3) IFLDX2_PRSN=IFLDX2
  ENDIF
ENDDO
IF (IFLDX2>0) THEN
  MOBB1X2=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1X2=    ',IPTX,IFLDX2
  IPTX=IPTX+IFLDX2
  IF(IFLDX2_PRWA > 0) THEN
    MOBB1X2_PRWA=MOBB1X2+IFLDX2_PRWA-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1X2_PRWA=  ',MOBB1X2_PRWA,1
  ENDIF
  IF(IFLDX2_PRSN > 0) THEN
    MOBB1X2_PRSN=MOBB1X2+IFLDX2_PRSN-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1X2_PRSN=  ',MOBB1X2_PRSN,1
  ENDIF
ENDIF

! Group VF=VARSF:
IFLDVF=0
IFLDVF_ZOF=0
IFLDVF_ALBF=0
IFLDVF_EMISF=0
IFLDVF_LSM=0
IFLDVF_VEG=0
IFLDVF_CVL=0
IFLDVF_CVH=0
IFLDVF_TVL=0
IFLDVF_TVH=0
IFLDVF_CI=0
DO JFLD=1,YSD_VFD%NUMFLDS
  IF (YSD_VF%YVF(JFLD)%LOBSHOR) THEN
    IFLDVF=IFLDVF+1
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YZOF) IFLDVF_ZOF=IFLDVF
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YALBF) IFLDVF_ALBF=IFLDVF
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YEMISF) IFLDVF_EMISF=IFLDVF
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YITM) IFLDVF_LSM=IFLDVF
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YVEG) IFLDVF_VEG=IFLDVF
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YCVL) IFLDVF_CVL=IFLDVF
```

```
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YCVH) IFLDVF_CVH=IFLDVF
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YTVL) IFLDVF_TVL=IFLDVF
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YTVH) IFLDVF_TVH=IFLDVF
    IF (YSD_VF%YVF(JFLD) == YSD_VF%YCI) IFLDVF_CI=IFLDVF
  ENDIF
ENDDO
IF (IFLDVF>0) THEN
  MOBB1VF=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF=    ',IPTX,IFLDVF
  IPTX=IPTX+IFLDVF
  IF(IFLDVF_ZOF > 0) THEN
    MOBB1VF_ZOF=MOBB1VF+IFLDVF_ZOF-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_ZOF=  ',MOBB1VF_ZOF,1
  ENDIF
  IF(IFLDVF_ALBF > 0) THEN
    MOBB1VF_ALBF=MOBB1VF+IFLDVF_ALBF-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_ALBF=  ',MOBB1VF_ALBF,1
  ENDIF
  IF(IFLDVF_EMISF > 0) THEN
    MOBB1VF_EMISF=MOBB1VF+IFLDVF_EMISF-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_EMISF=  ',MOBB1VF_EMISF,1
  ENDIF
  IF(IFLDVF_LSM > 0) THEN
    MOBB1VF_LSM=MOBB1VF+IFLDVF_LSM-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_LSM=  ',MOBB1VF_LSM,1
  ENDIF
  IF(IFLDVF_VEG > 0) THEN
    MOBB1VF_VEG=MOBB1VF+IFLDVF_VEG-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_VEG=  ',MOBB1VF_VEG,1
  ENDIF
  IF(IFLDVF_CVL > 0) THEN
    MOBB1VF_CVL=MOBB1VF+IFLDVF_CVL-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_CVL=  ',MOBB1VF_CVL,1
  ENDIF
  IF(IFLDVF_CVH > 0) THEN
    MOBB1VF_CVH=MOBB1VF+IFLDVF_CVH-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_CVH=  ',MOBB1VF_CVH,1
  ENDIF
  IF(IFLDVF_TVL > 0) THEN
    MOBB1VF_TVL=MOBB1VF+IFLDVF_TVL-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_TVL=  ',MOBB1VF_TVL,1
  ENDIF
  IF(IFLDVF_TVH > 0) THEN
    MOBB1VF_TVH=MOBB1VF+IFLDVF_TVH-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_TVH=  ',MOBB1VF_TVH,1
  ENDIF
  IF(IFLDVF_CI > 0) THEN
    MOBB1VF_CI=MOBB1VF+IFLDVF_CI-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VF_CI=  ',MOBB1VF_CI,1
  ENDIF
ENDIF

! Group VV=VCLIV:
IFLDVV=0
IFLDVV_ARG=0
IFLDVV_SAB=0
IFLDVV_HV=0
IFLDVV_ZOH=0
DO JFLD=1,YSD_VVD%NUMFLDS
  IF (YSD_VV%YVV(JFLD)%LOBSHOR) THEN
    IFLDVV=IFLDVV+1
    IF (YSD_VV%YVV(JFLD) == YSD_VV%YARG) IFLDVV_ARG=IFLDVV
    IF (YSD_VV%YVV(JFLD) == YSD_VV%YSAB) IFLDVV_SAB=IFLDVV
    IF (YSD_VV%YVV(JFLD) == YSD_VV%YHV) IFLDVV_HV=IFLDVV
    IF (YSD_VV%YVV(JFLD) == YSD_VV%YZOH) IFLDVV_ZOH=IFLDVV
  ENDIF
ENDDO
IF (IFLDVV>0) THEN
  MOBB1VV=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VV=    ',IPTX,IFLDVV
  IPTX=IPTX+IFLDVV
  IF(IFLDVV_ARG > 0) THEN
    MOBB1VV_ARG=MOBB1VV+IFLDVV_ARG-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VV_ARG=  ',MOBB1VV_ARG,1
  ENDIF
  IF(IFLDVV_SAB > 0) THEN
    MOBB1VV_SAB=MOBB1VV+IFLDVV_SAB-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VV_SAB=  ',MOBB1VV_SAB,1
  ENDIF
  IF(IFLDVV_HV > 0) THEN
    MOBB1VV_HV=MOBB1VV+IFLDVV_HV-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VV_HV=  ',MOBB1VV_HV,1
  ENDIF
  IF(IFLDVV_ZOH > 0) THEN
    MOBB1VV_ZOH=MOBB1VV+IFLDVV_ZOH-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VV_ZOH=  ',MOBB1VV_ZOH,1
  ENDIF
ENDIF

! Group VX=VCLIX:
IFLDVX=0
IFLDVX_ORO=0
```

```
IFLDVX_TSC=0
IFLDVX_SNO=0
DO JFLD=1,YSD_VXD%NUMFLDS
  IF (YSD_VX%YVX(JFLD)%LOBSHOR) THEN
    IFLDVX=IFLDVX+1
    IF (YSD_VX%YVX(JFLD) == YSD_VX%YORO) IFLDVX_ORO=IFLDVX
    IF (YSD_VX%YVX(JFLD) == YSD_VX%YTSC) IFLDVX_TSC=IFLDVX
    IF (YSD_VX%YVX(JFLD) == YSD_VX%YSNO) IFLDVX_SNO=IFLDVX
  ENDIF
ENDDO
IF (IFLDVX>0) THEN
  MOBB1VX=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VX=    ',IPTX,IFLDVX
  IPTX=IPTX+IFLDVX
  IF(IFLDVX_ORO > 0) THEN
    MOBB1VX_ORO=MOBB1VX+IFLDVX_ORO-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VX_ORO=  ',MOBB1VX_ORO,1
  ENDIF
  IF(IFLDVX_TSC > 0) THEN
    MOBB1VX_TSC=MOBB1VX+IFLDVX_TSC-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VX_TSC=  ',MOBB1VX_TSC,1
  ENDIF
  IF(IFLDVX_SNO > 0) THEN
    MOBB1VX_SNO=MOBB1VX+IFLDVX_SNO-1
    IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1VX_SNO=  ',MOBB1VX_SNO,1
  ENDIF
ENDIF

! Additional quantities (what does it contain? is it still useful?):
MOBB1WL=IPTX
IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1WL=    ',IPTX,1
IPTX=IPTX+1

! * Model errors statistics:

IF (LCASIG) THEN
  MOBB1EZ=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1EZ=    ',IPTX,NFLEVG
  IPTX=IPTX+NFLEVG
  MOBB1EH=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1EH=    ',IPTX,NFLEVG
  IPTX=IPTX+NFLEVG
  MOBB1EST=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1EST=   ',IPTX,1
  IPTX=IPTX+1
  MOBB1ESN=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1ESN=   ',IPTX,1
  IPTX=IPTX+1
  MOBB1ET2=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1ET2=   ',IPTX,1
  IPTX=IPTX+1
  MOBB1EH2=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1EH2=   ',IPTX,1
  IPTX=IPTX+1
  MOBB1EV1=IPTX
  IF (LLP) WRITE(NULOUT,'(1X,A12,2I4)') 'MOBB1EV1=   ',IPTX,1
  IPTX=IPTX+1
ENDIF

NFLDOBB1=IPTX-1

!      Part 3: compute parities for extension over pole.

IF (NFLDOBB1>0) THEN
  IF(ALLOCATED(RPAROB1)) DEALLOCATE(RPAROB1) ! To make SUOBB recallable
  ALLOCATE(RPAROB1(NFLDOBB1))
  IF(LLP)WRITE(IU,9) 'RPAROB1  ',SIZE(RPAROB1),SHAPE(RPAROB1)

  ! Initial setting to 1:
  RPAROB1(1:NFLDOBB1)=1.0_JPRB

  ! Change into -1 for U and V (GMV):
  RPAROB1(MOBB1U:MOBB1U+NFLEVG-1)=-1.0_JPRB
  RPAROB1(MOBB1V:MOBB1V+NFLEVG-1)=-1.0_JPRB
ENDIF
```

Appendix B: Body of revised SLINT for surface variables (part 5).

```
!    ------------------------------------------------------------------
!*       5. SURFACE FIELDS INTERPOLATION
!           ---------------------------

!*       5.1  MODIFICATION OF ZDSLAT/ZDSLON DUE TO LAND/SEA MASK

ZLEV(:,:) = 0.0_JPRB

IF ( LCANARI .AND. LSLREJ .AND. YSD_VF%YITM%LOBSHOR) THEN

  DO JROF = 1,KPROMB

    !for land observation
    IF ( KLSOBS(JROF) == 1 .OR. KLSOBS(JROF) == 2 ) THEN
      !* if point 1,1 is land and point 1,2 is sea
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) < 0.5_JPRB ) THEN
        ZDSLON(JROF,1,1) = 0._JPRB
      ENDIF
      !* if point 1,1 is sea and point 1,2 is land
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLON(JROF,1,1) = 1._JPRB
      ENDIF
      !* if both points 1,1 and 1,2 are sea
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) < 0.5_JPRB ) THEN
        ZDSLAT(JROF,1) = 1._JPRB
      ENDIF
      !* if point 2,1 is land and point 2,2 is sea
      IF (POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) < 0.5_JPRB ) THEN
        ZDSLON(JROF,1,2) = 0._JPRB
      ENDIF
      !* if point 2,1 is sea and point 2,2 is land
      IF (POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLON(JROF,1,2) = 1._JPRB
      ENDIF
      !* if both points 2,1 and 2,2 are sea
      IF (POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) < 0.5_JPRB ) THEN
        ZDSLAT(JROF,1) = 0._JPRB
      ENDIF
      !* if four points are sea
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) < 0.5_JPRB ) THEN
        ZLEV(JROF,1) = 2.0_JPRB
      ENDIF

    !for sea observation
    ELSEIF (KLSOBS(JROF) == 0 ) THEN
      !* if point 1,1 is sea and point 1,2 is land
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLON(JROF,1,1) = 0._JPRB
      ENDIF
      !* if point 1,1 is land and point 1,2 is sea
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) < 0.5_JPRB ) THEN
        ZDSLON(JROF,1,1) = 1._JPRB
      ENDIF
      !* if both points 1,1 and 1,2 are land
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLAT(JROF,1) = 1._JPRB
      ENDIF
      !* if point 2,1 is sea and point 2,2 is land
      IF (POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLON(JROF,1,2) = 0._JPRB
      ENDIF
      !* if point 2,1 is land and point 2,2 is sea
      IF (POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) < 0.5_JPRB ) THEN
        ZDSLON(JROF,1,2) = 1._JPRB
      ENDIF
      !* if both points 2,1 and 2,2 are land
      IF (POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
```

```
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLAT(JROF,1) = 0._JPRB
      ENDIF
      !* if four points are land
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZLEV(JROF,1) = 2.0_JPRB
      ENDIF
    ENDIF

  ENDDO

ENDIF

CALL GSTATS(1937,0)

!*       5.2  SURFACE FIELDS: GROUP SB=SOILB.

IFLDSB=0
DO JFLD=1,YSP_SBD%NUMFLDS
  IF (YSP_SB%YSB(JFLD)%LOBSHOR) THEN
    IFLDSB=IFLDSB+1
    CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
     & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
     & POBB1(1,MOBB1SB+IFLDSB-1),POBBF(1,MOBB1SB+IFLDSB-1))
    LDINT(MOBB1SB+IFLDSB-1)=.TRUE.
  ENDIF
ENDDO

!*       5.3  SURFACE FIELDS: GROUP SG=SNOWG.

IFLDSG=0
DO JFLD=1,YSP_SGD%NUMFLDS
  IF (YSP_SG%YSG(JFLD)%LOBSHOR) THEN
    IFLDSG=IFLDSG+1
    CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
     & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
     & POBB1(1,MOBB1SG+IFLDSG-1),POBBF(1,MOBB1SG+IFLDSG-1))
    LDINT(MOBB1SG+IFLDSG-1)=.TRUE.
  ENDIF
ENDDO

!*       5.4  SURFACE FIELDS: GROUP RR=RESVR.

IFLDRR=0
DO JFLD=1,YSP_RRD%NUMFLDS
  IF (YSP_RR%YRR(JFLD)%LOBSHOR) THEN
    IFLDRR=IFLDRR+1
    CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
     & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
     & POBB1(1,MOBB1RR+IFLDRR-1),POBBF(1,MOBB1RR+IFLDRR-1))
    LDINT(MOBB1RR+IFLDRR-1)=.TRUE.
  ENDIF
ENDDO

!*       5.5  SURFACE FIELDS: GROUP CL=CLS.

IFLDCL=0
DO JFLD=1,YSP_CLD%NUMFLDS
  IF (YSP_CL%YCL(JFLD)%LOBSHOR) THEN
    IFLDCL=IFLDCL+1
    CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
     & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
     & POBB1(1,MOBB1CL+IFLDCL-1),POBBF(1,MOBB1CL+IFLDCL-1))
    LDINT(MOBB1CL+IFLDCL-1)=.TRUE.
  ENDIF
ENDDO

!*       5.6  SURFACE FIELDS: GROUP X2=XTRP2.

LL5=.NOT.(LOBSC1.OR.LOBSREF.OR.LCANARI)
IF (.NOT.LL5) THEN
  IFLDX2=0
  DO JFLD=1,YSP_X2D%NUMFLDS,2
    IF (YSP_X2%YX2(JFLD)%LOBSHOR.AND.YSP_X2%YX2(JFLD+1)%LOBSHOR) THEN
      IFLDX2=IFLDX2+1
      CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
       & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
       & POBB1(1,MOBB1X2+IFLDX2-1),POBBF(1,MOBB1X2+IFLDX2-1))
      LDINT(MOBB1X2+IFLDX2-1)=.TRUE.
    ENDIF
  ENDDO
ENDIF
```

```
!*       5.7  SURFACE FIELDS: GROUP VF=VARSF.

IFLDVF=0
DO JFLD=1,YSD_VFD%NUMFLDS
  IF (YSD_VF%YVF(JFLD)%LOBSHOR) THEN
    IFLDVF=IFLDVF+1
    IF (YSD_VF%YVF(JFLD)%CSLINT='LAIDLIOBS ') THEN
      CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
       & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
       & POBB1(1,MOBB1VF+IFLDVF-1),POBBF(1,MOBB1VF+IFLDVF-1))
    ELSEIF (YSD_VF%YVF(JFLD)%CSLINT='LAIDLIC   ') THEN
      CALL LAIDLIC(KASLB1,KPROMB,1,KPROMB,1,IFLDN,IFLDX,&
       & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
       & POBB1(1,MOBB1VF+IFLDVF-1),POBBF(1,MOBB1VF+IFLDVF-1))
    ENDIF
    LDINT(MOBB1VF+IFLDVF-1)=.TRUE.
  ENDIF
ENDDO

!*       5.8  SURFACE FIELDS: GROUP VV=VCLIV.

IFLDVV=0
DO JFLD=1,YSD_VVD%NUMFLDS
  IF (YSD_VV%YVV(JFLD)%LOBSHOR) THEN
    IFLDVV=IFLDVV+1
    CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
     & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
     & POBB1(1,MOBB1VV+IFLDVV-1),POBBF(1,MOBB1VV+IFLDVV-1))
    LDINT(MOBB1VV+IFLDVV-1)=.TRUE.
  ENDIF
ENDDO

!*       5.9  SURFACE FIELDS: GROUP VX=VCLIX.

LL5=.NOT.(LOBSC1.OR.LOBSREF.OR.LCANARI)
IF (.NOT.LL5) THEN
  IFLDVX=0
  DO JFLD=1,YSD_VXD%NUMFLDS
    IF (YSD_VX%YVX(JFLD)%LOBSHOR) THEN
      IFLDVX=IFLDVX+1
      CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
       & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
       & POBB1(1,MOBB1VX+IFLDVX-1),POBBF(1,MOBB1VX+IFLDVX-1))
      LDINT(MOBB1VX+IFLDVX-1)=.TRUE.
    ENDIF
  ENDDO
ENDIF

!*       5.10  ADDITIONAL FIELDS.

CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
 & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
 & POBB1(1,MOBB1WL),POBBF(1,MOBB1WL))
LDINT(MOBB1WL)=.TRUE.

!*       5.11  2D FIELDS FOR MODEL ERRORS STATISTICS (FRENCH OI)

IF ( LCANARI .AND. LCASIG ) THEN

  CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
   & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
   & POBB1(1,MOBB1EST),POBBF(1,MOBB1EST))
  LDINT(MOBB1EST)=.TRUE.

  CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
   & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
   & POBB1(1,MOBB1ESN),POBBF(1,MOBB1ESN))
  LDINT(MOBB1ESN)=.TRUE.

  CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
   & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
   & POBB1(1,MOBB1ET2),POBBF(1,MOBB1ET2))
  LDINT(MOBB1ET2)=.TRUE.

  CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
   & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
   & POBB1(1,MOBB1EH2),POBBF(1,MOBB1EH2))
  LDINT(MOBB1EH2)=.TRUE.

  CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
   & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
   & POBB1(1,MOBB1EV1),POBBF(1,MOBB1EV1))
  LDINT(MOBB1EV1)=.TRUE.
```

```
ENDIF

CALL GSTATS(1937,1)

!*       5.12  SURFACE TEMPERATURE : SPECIAL TREATMENT FOR COASTAL OBS

! Some observations need 2 Ts interpolated fields :
! one for land parameter (T2m,Hu2m,...), and one for sea parameter (SST).
! The land case is OK, now prepare Ts over sea.

PSLSST(1:KPROMB)=POBBF(1:KPROMB,MOBB1RR_T)

IF ( LCANARI .AND. LSLREJ .AND. YSD_VF%YITM%LOBSHOR) THEN

  IDUM1=0

  DO JROF = 1,KPROMB

    IF ( KLSOBS(JROF) == 2 ) THEN

      IDUM1=IDUM1+1

      !* if point 1,1 is sea and point 1,2 is land
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLON(JROF,1,1) = 0._JPRB
      ENDIF
      !* if point 1,1 is land and point 1,2 is sea
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) < 0.5_JPRB ) THEN
        ZDSLON(JROF,1,1) = 1._JPRB
      ENDIF
      !* if both points 1,1 and 1,2 are land
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLAT(JROF,1) = 1._JPRB
      ENDIF
      !* if point 2,1 is sea and point 2,2 is land
      IF (POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) < 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLON(JROF,1,2) = 0._JPRB
      ENDIF
      !* if point 2,1 is land and point 2,2 is sea
      IF (POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) < 0.5_JPRB ) THEN
        ZDSLON(JROF,1,2) = 1._JPRB
      ENDIF
      !* if both points 2,1 and 2,2 are land
      IF (POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZDSLAT(JROF,1) = 0._JPRB
      ENDIF
      !* if four points are land
      IF (POBB1(ILOULI(JROF,1,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,1,2),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,1),MOBB1VF_LSM) > 0.5_JPRB .AND.&
       & POBB1(ILOULI(JROF,2,2),MOBB1VF_LSM) > 0.5_JPRB ) THEN
        ZLEV(JROF,1) = 1.0_JPRB
      ENDIF
    ENDIF
  ENDDO

  CALL GSTATS(1937,0)
  IF (IDUM1 /= 0) THEN
    CALL LAIDLIOBS(KASLB1,KPROMB,1,KPROMB,1,1,IFLDN,IFLDX,&
     & ZDSLAT,ZDSLON(1,1,1),ILOULI,&
     & POBB1(1,MOBB1RR_T),POBSST)
    LDINT_SST=.TRUE.
  ENDIF
  CALL GSTATS(1937,1)
ENDIF

!*       5.14  CONSIDER LAND/SEA MASK  (FRENCH OI)

! We want to reject the parameters of the observations which are surrounded by
! 4 sea points over ground (T2m and Hu2m) and with 4 land points over sea (SST).
! To allow this rejection, we use the PSLEMIS array to "mark" the concerned obs.

IF ( LCANARI .AND. LSLREJ .AND. YSD_VF%YEMISF%LOBSHOR) THEN
  POBBF(1:KPROMB,MOBB1VF_EMISF)= &
   & POBBF(1:KPROMB,MOBB1VF_EMISF)+ZLEV(1:KPROMB,1)*100._JPRB
ENDIF
```

Appendix C: Interpolations currently done in SLINT and dataflow (status CY35T2).

Class of fields	Group	Root (attrib for surf)	POBB1.. (input)	PSL.. (output)	Paragraph	Weights modified by LSM?
GMV		U	POBB1U5	PSLU	3.1	no
		V	POBB1V5	PSLV	3.2	no
		T	POBB1T5	PSLT	3.3	no
GFL			POBB1GFL5	PSLGFL	3.4+4.4	no
GMVS		SP	POBB1SP5	PSLSP	3.6	no
Constants		OR	POBB1OR5	PSLOROG	3.7	no
		CO	POBB1CO5	PSLCORI	4.3	no
		DL	POBB1DL5	PSLDL	3.8	no
		DM	POBB1DM5	PSLDM	3.8	no
Surface P	SB=SOILB	YQ	POBB1WS5	PSLWS	5.3	yes
Surface P	SG=SNOWG	YF	POBB1SN5	PSLSN	5.4	yes
Surface P	RR=RESVR	YT	POBB1TS5	PSLTS	5.2	yes
				PSLSST	5.13	yes
		YW	POBB1WS5	PSLWS	5.3	yes
		YIC	missing	missing		
Surface P	CL=CLS	YTCLS	POBB1TCLS5	PSLTCLS	5.2	yes
		YHUCLS	POBB1HUCLS	PSLHUCLS	5.2	yes
Surface P	X2=XTRP2	YX2(1) +YX2(2)	POBB1PW5	PSLPRWA	5.8	yes
		YX2(3) +YX2(4)	POBB1PS5	PSLPRSN	5.9	yes
Surface D	VF=VARSF	YZOF	POBB1ZO5	PSLZO	5.5	yes
		YALBF	POBB1AL5	PSLALBE	5.7.1	yes
		YEMISF	POBB1EM5	PSLEMIS	5.7.2	yes
		YITM	POBB1LS5	PSLLSMA	5.7.3	yes
		YVEG	POBB1VG5	PSLVEGE	5.7.9	yes
		YCVL	POBB1CVL5	PSLCVL	5.7.6	yes
		YCVH	POBB1CVH5	PSLCVH	5.7.5	yes
		YTVL	POBB1TVL5	PSLTVL	5.7.8	yes
		YTVH	POBB1TVH5	PSLTVH	5.7.7	yes
		YCI	POBB1CI5	PSLCI	5.7.4	yes
Surface D	VV=VCLIV	YARG	POBB1ARG5	PSLARG	5.11.1	yes
		YSAB	POBB1SAB5	PSLSAB	5.11.2	yes
		YHV	POBB1HV5	PSLHV	5.11.3	yes
		YZOH	POBB1ZOH5	PSLZOH	5.11.4	yes
Surface D	VX=VCLIX	YORO	missing	missing		
		YTSC	POBB1XTS	PSLXTS	5.10	yes
		YSNO	POBB1XSN	PSLXSN	5.10	yes
????		WL	POBB1WL5 =1 everywhere	PSLWL	5.6	yes
Model error stats. for CANARI		EZ	POBB1EZ5	PSLEZ	4.2	no
		EH	POBB1EH5	PSLEH	4.2	no
		EST	POBB1EST5	PSLEST	5.12	yes
		ESN	POBB1ESN	PSLESN	5.12	yes
		ET2	POBB1ET2	PSLET2	5.12	yes
		EH2	POBB1EH2	PSLEH2	5.12	yes
		EV1	POBB1EV1	PSLEV1	5.12	yes

Appendix D: Filling GOM in MPOBSEQ_PACK: memory transfers currently done (status CY35T2).

Class of fields	Group	Root (attrib for surf)	PSL.. (input)	GOM (output)	Order of storing in PBUFS
GMV		U	PSLU	YDGOMUA%U	1 (nflevg)
				YDGOMS%UL (l=nflevg)	5
		V	PSLV	YDGOMUA%V	2 (nflevg)
				YDGOMS%VL (l=nflevg)	6
		T	PSLT	YDGOMUA%T	3 (nflevg)
				YDGOMS%TL (l=nflevg)	13
GFL			PSLGFL	YDGOMUA%GFL	4 (nflevg)
				YDGOMS%QL (l=nflevg, for "q")	14
GMVS		SP	PSLSP	YDGOMS%SP	7
Constants		OR	PSLOROG	YGOMEC%ECOROG	18
		CO	PSLCORI	YGOMEC%ECCORI	19
		DL	PSLDL	YGOMEC%ECDL	28
		DM	PSLDM	YGOMEC%ECDM	29
Surface P	SB=SOILB	YQ	PSLWS	YDGOMS%WS	9
Surface P	SG=SNOWG	YF	PSLSN	YDGOMS%SN	10
Surface P	RR=RESVR	YT	PSLTS	YDGOMS%TS	8
			PSLSST	YDGOMS%SST	15
		YW	PSLWS	YDGOMS%WS	9
		YIC	missing	missing	
Surface P	CL=CLS	YTCLS	PSLTCLS	YDGOMS%TCLS	16
		YHUCLS	PSLHUCLS	YDGOMS%HUCLS	17
Surface P	X2=XTRP2	YX2(1) +YX2(2)	PSLPRWA	YGOMCAN%ECPRWA	41
		YX2(3) +YX2(4)	PSLPRSN	YGOMCAN%ECPRSN	42
Surface D	VF=VARSF	YZOF	PSLZO	YDGOMS%ZO	11
		YALBF	PSLALBE	YGOMEC%ECALBE	20
		YEMISF	PSLEMIS	YGOMEC%ECEMIS	21
		YITM	PSLLSMA	YGOMEC%ECLSMA	22
		YVEG	PSLVEGE	YGOMEC%ECVEGE	23 to 27 mf
		YCVL	PSLCVL	YGOMEC%ECCVL	25ec
		YCVH	PSLCVH	YGOMEC%ECCVH	24ec
		YTVL	PSLTVL	YGOMEC%ECTVL	27ec
		YTVH	PSLTVH	YGOMEC%ECTVH	26ec
		YCI	PSLCI	YGOMEC%ECCI	23ec
Surface D	VV=VCLIV	YARG	PSLARG	YGOMEC%ECARG	30
		YSAB	PSLSAB	YGOMEC%ECSAB	31
		YHV	PSLHV	YGOMEC%ECHV	32
		YZOH	PSLZOH	YGOMEC%ECZOH	33
Surface D	VX=VCLIX	YORO	missing	missing	
		YTSC	PSLXTS	YGOMCAN%ECCLITS	43
		YSNO	PSLXSN	YGOMCAN%ECCLISN	44
????		WL	PSLWL	YDGOMS%WL	12
Model error stats. for CANARI		EZ	PSLEZ	YGOMCANA%EMMZ	34 (nflevg)
		EH	PSLEH	YGOMCANA%EMMH	35 (nflevg)
		EST	PSLEST	YGOMCAN%EMMST	36
		ESN	PSLESN	YGOMCAN%EMMSN	37
		ET2	PSLET2	YGOMCAN%EMMT2	38
		EH2	PSLEH2	YGOMCAN%EMMH2	39
		EV1	PSLEV1	YGOMCAN%EMMV1	40

Appendix E: conditions for allocating surface fields, and interpolating them (status CY35T2).

```
LLUSE_OBSHOR=LOBSC1.OR.(NCONF/100 == 1).OR.(NCONF/100 == 7)
LLASSIMAROME=LAROME.AND.(.NOT.(LSCREEN.OR.(NCONF == 131)))
LLFP_CLASSIC=LFPOS.AND.(NFPCLI == 3)
LLFP_SURFEX=LFPOS.AND..NOT.LFPART2.AND.LMSE
```

Group	Root (attrib for surf)	Allocation in SU_SURF_FLDS	Interpolation in SLINT (assumes LLUSE_OBSHOR=T)
SB=SOILB	YQ	LECMWF: always .NOT.LECMWF: LMPHYS.AND.LSOLV.AND.LTPROF	LECMWF: always
SG=SNOWG	YF	LECMWF: always .NOT.LECMWF: LMPHYS.AND.(.NOT.LLASSIMAROME)	LECMWF: always .NOT.LECMWF: always
RR=RESVR	YT	LECMWF: always .NOT.LECMWF: LMPHYS	LECMWF: always .NOT.LECMWF: always
	YW	LECMWF: always .NOT.LECMWF: LMPHYS.AND.(.NOT.LLASSIMAROME)	.NOT.LECMWF: always
	YIC	.NOT.LECMWF: LMPHYS.AND.((LSOLV.AND.LFGEL).OR.LMSE)	
CL=CLS	YTCLS	.NOT.LECMWF: LMPHYS.AND.LDIRCLSMOD	LDIRCLSMOD
	YHUCLS	.NOT.LECMWF: LMPHYS.AND.LDIRCLSMOD	LDIRCLSMOD
X2=XTRP2	YX2(1) +YX2(2) YX2(3) +YX2(4)	???	LCANARI
VF=VARSF	YZOF	LECMWF: always .NOT.LECMWF: LMPHYS.AND.(.NOT.(LMSE.OR.LLASSIMAROME))	LECMWF: always .NOT.LECMWF: always
	YALBF	LECMWF: always .NOT.LECMWF: LMPHYS.AND.(.NOT.(LMSE.OR.LLASSIMAROME))	LECMWF: always .NOT.LECMWF: always
	YEMISF	LECMWF: always .NOT.LECMWF: LMPHYS.AND.(.NOT.(LMSE.OR.LLASSIMAROME))	LECMWF: always .NOT.LECMWF: always
	YITM	LECMWF: always .NOT.LECMWF: LMPHYS.OR.((.NOT.(LMPHYS.OR.LEPHYS)).AND.LSPHLC)	LECMWF: always .NOT.LECMWF: always
	YVEG	.NOT.LECMWF: LMPHYS.AND.(.NOT.((LMSE.AND.LFPART2).OR.LLASSIMAROME))	.NOT.LECMWF: always
	YCVL	LECMWF: always	LECMWF: always
	YCVH	LECMWF: always	LECMWF: always
	YTVL	LECMWF: always	LECMWF: always
	YTVH	LECMWF: always	LECMWF: always
	YCI	LECMWF: always	LECMWF: always
VV=VCLIV	YARG	.NOT.LECMWF: LMPHYS.AND.(LSOLV.OR.LMSE).AND.(.NOT.LLASSIMAROME)	LSOLV
	YSAB	.NOT.LECMWF: LMPHYS.AND.(LSOLV.OR.LMSE).AND.(.NOT.LLASSIMAROME)	LSOLV
	YHV	.NOT.LECMWF: LMPHYS.AND.LSOLV.AND.(.NOT.LLASSIMAROME)	LSOLV
	YZOH	.NOT.LECMWF: LMPHYS.AND.LSOLV.AND.(.NOT.LLASSIMAROME)	LSOLV (1)
VX=VCLIX	YORO	.NOT.LECMWF: LCANARI.OR.LLFP_CLASSIC.OR.LLFP_SURFEX	not interpolated
	YTSC	.NOT.LECMWF: LCANARI.OR.LLFP_CLASSIC.OR.LLFP_SURFEX	LCANARI
	YSNO	.NOT.LECMWF: LCANARI.OR.LLFP_CLASSIC.OR.LLFP_SURFEX .AND.(.NOT.LLASSIMAROME)	LCANARI

(1): interpolated value replaced by 1 if YSD_VDD%NUMFLDS<=8