Martina Tudor, Toulouse, October 2003, phasing

tudor@cirus.dhz.hr

A short description of physics/dynamics interface in the new data flow

CONTENTS:

0. introduction

During autumn 2003 phasing, GMV and GFL structures were introduced to the model data flow. GMV array is used for u, v, T, vort, div and other 3D variables. GMVS is used surface pressure and other dynamical 2D variables. GFL is used for q, L, I, O3 and other 3D variables.

In the call to physics parameterisation package, it is required to know the values of some surface variables and a few pseudo historical 3D variables from the previous time-step. They are transferred through PGP array using GPPBUF that fortunately remained after GMV GFL revolution. GPPBUF buffer (PGP arrays; ZGP in cpg) was designed to contain surface data that are transferred from one time-step to the next. It is read in cpg_gp, the values used/modified in mf_phys, some are diagnosed in cpg_dia and all of them written to GPPBUF in cpg_end. The obvious intruders into PGP array are a few arrays of 3D fields. The 3D variables stored in surface field buffer GPPBUF are (according to the names given in aplpar) PEMTD, PEMTU and PTRS0 (used in cloud and radiation I/O for ECMWF physics), PFPLCH and PFPLSH (convective and stratiform precipitation fluxes). The vertical dimension of the last two variables is 0:KLEV meaning that they are defined on KLEV+1 levels (this is valid for the first 3 when used in MF physics package too). The corresponding pointers in GPPBUF buffer (PGP array) are MVREMTD, MVREMTU, MVRTSW for radiation and MVRRCV and MVRRST for precipitation fluxes. Additionally one 3D field (PNEBH, pointer MVNNEBH) contained a 3D field and additional levels where some additional 2D fields were put.

1. Prior to phasing

The 3D field, stored in the surface field buffer, that contained a 3D field (of precipitation fluxes on NFLEV+1 levels) and a few 2D fields on additional model levels was PNEBH. The vertical extent of PNEBH was NVCLIN (KNEBH in aplpar). It varied depending on the configuration of the physics parameterisation package. This was confusing. In the spring 2003 the first attempt to make clear interface for physics paremeretisations was made. The solution was to call the new routine, aplparint, in the place of aplpar, from mf_phys. Inside aplparint, PNEBH was split into its components prior to the aplpar call. The components are passed to aplpar as separate arguments. The components are collected back to PNEBH before the end of aplparint. The data-flow of these variables was changed only below aplparint. Above mf_phys (meaning in cpg and elsewhere) data flow remained unchanged.

The only purpose of aplparint was to split PNEBH into its components: PPCCH (convective precipitation flux), that is 3D on KLEV+1 levels, and 2D variables PTCCH, PSCCH, PBCCH and PPBLH (total convective cloudiness, top of convective layer, bottom of convective layer and PBL height). mf_phys calls aplparint in the place of aplpar, PNEBH (PGP(1,MVNNEBH) in mf_phys) is passed as an argument 3D variable, defined on KNEBH levels (NVCLIN in mf_phys).

However, this solution was also confusing and impractical, so further steps were undertaken during phasing. Furthermore, convective precipitation flux is also stored in PFPLCH, therefore PPCCH duplicated PFPLCH field. So it was replaced by PFPLCH in acnebr and aplpar, and removed from aplparint. Further details about the use of these variables can be read in Appendix B.

2. Split of PNEBH

A special case of 3D variable stored in surface field buffer is PNEBH that is defined on NVCLIN levels (KNEBH in aplpar), and this size varies depending on the model configuration. After the modifications from the previous chapter, PNEBH contained only different 2D variables on different levels – so it is a 3D array containing 2D variables. The corresponding pointer in GPPBUF buffer (PGP array) was MVNNEBH.

These 2D variables were put into GPPBUF as separate fields, just like all other 2D surface fields (define pointer in ptrgpd and calculate it in sugpprp). The only purpose of aplparint was to split PNEBH into its 2D components, aplparint became obsolete. aplparint subroutine, PNEBH and the corresponding pointer MVNNEBH were removed. Pointers to the new fields in GPPBUF buffer (PGP array) are:

Variable (in aplpar)	Pointer (in ptrgpd)	
PTCCH	MVTCCH	Convective cloudiness
PSCCH	MVSCCH	Top of the convective layer
PBCCH	MVBCCH	Bottom of the convective layer
PPBLH	MVPBLH	PBL height

NVCLIN is no longer used in MF physics but it is used in ECMWF part, so it remained there but was removed from MF part. New pointers/arrays were added to VCLIH part of ptrgpd (with 3D part of precipitation fluxes).

The corresponding part of the set-up in sudim1 was also modified. NVCLIH value is modified according to the values of LNEBCO, LGWDC and LNEBR (see Chapter 3.1).

3. 3D fields from surface field buffer to GFL

3D fields are obvious intruders into the surface field buffer. The new data structures were set-up; GMV, GMVS and GFL. The GMV and GMVS structures contain variables involved in the semi implicit part of the computations (u, v, T, ps, vort, div …) and the GFL structure contains other 3D variables (q, L, I, O3, …). The 3D variables stored in the surface fields buffer, that are used by MF only, were moved to the GFL structure.

The 3D variables stored in GPPBUF are downward longwave emissivity, upward longwave emissivity and shortwave transmisivity (used in cloud and radiation I/O for ECMWF physics), and convective and stratiform precipitation fluxes (used in MF physics). The last two variables are fluxes defined on NFLEVG+1 levels, from 0 to NFLEVG. The GFL structure is designed in a way that the fields in GFL structure are all 3D with the vertical extent always equal to the number of levels in the model. The precipitation fluxes always have values equal to zero on level 0. This means only the lower NFLEVG have to be put into GFL structure. This is not valid for the ECMWF fluxes used in radiation scheme (that can be defined on more levels). The 3D fluxes stored in the surface field buffer used in MF physics only (the precipitation fluxes PFPLCH and PFPLSH) were moved to GFL structure. Other 3 remained in the surface field buffer.

The new 3D variables introduced in the GFL structure are CPF and SPF (convective and stratiform precipitation fluxes). The variables are purely grid-point and used only in physics parameterisation package. It is not required to write this variables to a file nor to read them from file. At first time-step they are set to zero. At the end of the call to MF physics package, they are stored to GFLT1. At the end of scan2mdm GFLT1 is copied to GFL for these fields. At the next time-step, these fields are used as input fields for MF physics from GFL (and again as output in GFLT1).

It is not actually necessary for these variables to have t+dt representation. They could be stored to the GFL array on output of MF physics package, not involving GFLT1 part. But there is a problem described in Appendix C.

1. **Introduction of the new 3D variables into GFL – coding**

   The following steps have been taken:

   Define pointers YCPF and YSPF as TYPE_GFL_COMP and YCPF_NL and YSPF_NL as TYPE_GFL_NAML in module **yom_ygfl**, increase the size of JPNAMED_GFL for the number of new variables (from 5 to 7 in this case). Example:

   ```
   TYPE(TYPE_GFL_COMP),POINTER :: YCPF ! Convective precipitation flux

   TYPE(TYPE_GFL_COMP),POINTER :: YSPF ! Stratiform precipitation flux

   …

   TYPE(TYPE_GFL_NAML) :: YCPF_NL ! Convective precipitation flux

   TYPE(TYPE_GFL_NAML) :: YSPF_NL ! Stratiform precipitation flux
   ```

   Add YCPF_NL and YSPF_NL to **namgfl.h**.

   Define YFACPF and YFASPF as TYPE(FAD) in module **yomfa**. Example

   ```
   TYPE(FAD) :: YFACPF ! Convective precipitation flux

   TYPE(FAD) :: YFASPF ! Stratiform precipitation flux
   ```

   Add YFACPF and YFASPF to **namfa.h**.

   Define names for the components in **sufa**. Example:

   ```
   YFACPF =YSUFAD('CV_PREC_FLUX ',.TRUE.)

   YFASPF =YSUFAD('ST_PREC_FLUX ',.TRUE.)
   ```

   Add calls to DEFINE_GFL_COMP that will set the attributes for these components in **sugfl**. Example:

   ```
   YCPF=>YGFLC(JPGFL-5)

   YSPF=>YGFLC(JPGFL-6)
   ```

   YGFLC is the array that contains descriptors of all GFL components.

   …

```
IF(YCPF_NL%LGP)THEN

IGFLPTR=YGFL%NUMFLDS+1

YCPF=>YGFLC(IGFLPTR)

CALL DEFINE_GFL_COMP(YCPF,YFACPF%CLNAME,YCPF_NL%IGRBCODE,

.TRUE.,YCPF_NL%LREQIN,.FALSE.,.FALSE.,LLT1,.TRUE.)

ENDIF

IF(YSPF_NL%LGP)THEN

IGFLPTR=YGFL%NUMFLDS+1

YSPF=>YGFLC(IGFLPTR)

CALL DEFINE_GFL_COMP(YSPF,YFASPF%CLNAME,YSPF_NL%IGRBCODE,

.TRUE.,YSPF_NL%LREQIN,.FALSE.,.FALSE.,LLT1,.TRUE.)

ENDIF
```

In the call to DEFINE_GFL_COMP the following attributes are assigned to the variable (the first argument is the pointer to the variable – YCPF): name (YFACPF%CLNAME), grib-code (YCPF_NL%IGRBCODE), does it have grid-point or spectral representation (LDGP=.TRUE.), is it required in input (YCPF_NL%LREQIN), does it have derivatives (LDERS=.FALSE., this can be true only for spectral fields), does it need to be present in trajectory (LD5=.FALSE.), does it need to be present in GFLT1 (LDT1=LLT1), all these arguments are obligatory and the last argument is optional: LDGPINGP is true if the grid-point field is input as grid-point.

There are similar calls under LSP. There are always two calls to DEFINE_GFL_COMP per variable, one if it is grid-point and another if it is spectral.

Add calls to SET_GFL_ATTR that will set the additional attributes for these components in **sudyn**.

```
IF(YCPF%LACTIVE) THEN

CALL SET_GFL_ATTR(YCPF,LDADV=.FALSE.,LDT9=LLT9,LDSLP=LSLPHY,

LDCOUPLING=YCPF_NL%LCOUPLING,CDSLINT=YCPF_NL%CSLINT)

ENDIF

IF(YSPF%LACTIVE) THEN

CALL SET_GFL_ATTR(YSPF,LDADV=.FALSE.,LDT9=LLT9,LDSLP=LSLPHY,

LDCOUPLING=YSPF_NL%LCOUPLING,CDSLINT=YSPF_NL%CSLINT)

ENDIF
```

In the call to SET_GFL_ATTR the following attributes are assigned to the variable (the first argument is the pointer to the variable, other arguments are optional): is the field advected (LDADV), does it have t-dt representation (LDT9), does it have SL physics representation (LDSLP), is it adjusted at t (LDADJUST0), is it adjusted at t+dt (LDADJUST1), is it coupled (LDCOUPLING), is it biperiodised (LDBIPER) and the last defines the type of semi-Lagrangian interpolation (CDSLINT).

Add set-up of XXX_NL GFL attributes in **sudim1**, also add some control IF(…) CALL ABORT in **sudim1** , and elsewhere if you find necessary.

```
IF (.NOT.LECMWF .AND. ((NCONF == 1).OR.(NCONF == 801)

.OR.(NCONF == 601).OR.(NCONF == 131)).AND.LMPHYS) THEN

IF(LNEBCO.OR.LGWDC) NVCLIH=NVCLIH+3 ! PTCCH,PSCCH,PBCCH

IF(LNEBR) NVCLIH=NVCLIH+1 ! PPBLH

IF(LRRGUST) THEN

YCPF_NL%LGP=.TRUE.

YSPF_NL%LGP=.TRUE.

ELSEIF((LNEBN.OR.LNEBR).AND.(.NOT.LRRGUST)) THEN

YCPF_NL%LGP=.TRUE.

ENDIF

ENDIF

IF(YCPF_NL%LGP.AND.YCPF_NL%LSP) THEN

WRITE(NULERR,'(''BOTH YCPF_NL%LGP AND YCPF_NL%LSP TRUE'')')

CALL ABOR1(' SUDIM1 ')

ENDIF

IF(YSPF_NL%LGP.AND.YSPF_NL%LSP) THEN

WRITE(NULERR,'(''BOTH YSPF_NL%LGP AND YSPF_NL%LSP TRUE'')')

CALL ABOR1(' SUDIM1 ')

ENDIF

IF((LNEBN.OR.LNEBR.OR.LRRGUST).AND..NOT.YCPF_NL%LGP) THEN

WRITE(NULERR,'(''LNEBN.OR.LNEBR.OR.LRRGUST REQUIRES YCPF_NL%LGP'')')

CALL ABOR1(' SUDIM1 ')

ENDIF

IF(LRRGUST.AND..NOT.YSPF_NL%LGP) THEN

WRITE(NULERR,'(''LRRGUST REQUIRES YSPF_NL%LGP'')')

CALL ABOR1(' SUDIM1 ')

ENDIF
```

The final step is to use the variables.

2. Usage of variables stored in GFL arrays

Example 1: how to pass a variable stored in GFL structure from cpg to mf_phys as an argument.

```
USE YOMGFL , ONLY : GFL
USE YOM_YGFL , ONLY : YQ, YL, YI, YCPF, YSPF
```

… and then put into arguments:

```
GFL(1,1,YCPF%MP,IBL),GFL(1,1,YSPF%MP,IBL)
```

IBL is the index of the JKGLO loop. Each NPROMA packet has a different one.

YCPF%MP is the pointer to time t basic field of convective precipitation flux.

Example 2: This is an example how to use variables stored in GFL arrays more directly. It is not necessary to have them as arguments. In this part of the code t1 values of the variable are copied to t0 if the variable has t1 representation and if it is (purely) gridpoint. (similar to what is done in scan2mdm)

```
USE YOMGFL , ONLY : GFL, GFLT1

USE YOM_YGFL , ONLY : YGFL

USE YOM_YGFL , ONLY : YQ, YL, YI, YCPF, YSPF

…

DO JGFL=1,YGFL%NUMFLDS

IF(YGFLC(JGFL)%LGP.AND.YGFLC(JGFL)%LT1) THEN

DO JLEV=1,NLEV

DO JLON=KSTART,KEND

GFL(JLON,JLEV,YGFLC(JGFL)%MP,IBL) = GFLT1(JLON,JLEV,YGFLC(JGFL)%MP1,IBL)

ENDDO

ENDDO

ENDIF

ENDDO
```

IBL is the numerator of the NPROMA packet in JKGLO loop.

3. A few additional remarks

These variables were stored in the surface fields buffer, therefore the pointers MVRRCV and MVRRST were removed from ptrgpd and sugpprp. aplparint was removed. The part of the set-up was changed (the new set-up is already shown in the previous section).

The variables represent fluxes defined between model levels (on NFLEVG+1 levels, from 0 to NFLEVG). GFL is defined in such a way that all variables have 1:NFLEVG vertical extent. The nature of these fluxes is that they are equal to zero on the top of the atmosphere. Therefore only NFLEVG levels of the variables have to be stored to GFL. The fluxes that enter aplpar are defined on NFLEVG+1 levels (0:NFLEVG). Therefore, local arrays in mf_phys are allocated (if necessary) with 0:NFLEVG vertical extent and filled with data from GFL from 1 to NFLEVG and zero on level 0. These arrays are passed to aplpar as arguments. The new values are copied to GFLT1 before the local arrays are allocated. The time-stepping of these fields is done at the end of scan2mdm, where the GFLT1 values are copied to GFL.

The allocation of local variables requires more allocated memory. The only cure could be to have these fluxes not defined on the top of the atmosphere, but this contradicts the rule that all the fluxes have 0:NFLEVG vertical representation (for 3D fluxes).

The code changes shown in the examples in this chapter do not reflect the developments in the following Chapter.

4. PC scheme

The topic was phasing of the physics-dynamics interface for the predictor-corrector scheme. However, as will be described, the development that was to be phased relied on a structure (GT9 buffer) that was replaced by a different one (GFL and GMV). This required more a new development than phasing.

1. GT9 solution - Summary of the work done in Prague

The PC scheme was coded in a following way: if the physics package was called, the physics tendencies were calculated, the accumulated and instantaneous fluxes were diagnosed, but at the end of the corrector step, the variables were not updated with physics tendencies. This lead to a strange situation, for example for precipitation, the water in the precipitation was never subtracted from the moisture in the air. Therefore, it was raining, but the moisture was not removed from the 3D moisture fields.

To use physics tendencies during PC scheme properly, the following code changes are introduced:

Physics tendencies are transferred using GT9 buffer. The size of GT9 buffer is increased in sudim2. PCXPTT9 arrays contain tendency of variable X after the call to physics and are stored in GT9 buffer. In this way, physics tendencies are transported from predictor to corrector and the final values are updated with physics tendencies. The scheme works in a way that if the trajectories are re-calculated, physics tendency values are re-interpolated.

During predictor: In cpg, after the call to diagnostics, pt2gt9 stores the tendencies from physics to PCXPTT9 array, these arrays are written to GT9 buffer in cpg_end using sc2wrt.

During corrector: GT9 buffer is read by sc2rdt9 called from cpg_gp and PCXPTT9 arrays are filled. If the physics package is not called, gt92pt is called from cpg, the physics tendencies are obtained by copying PCXPTT9 arrays into the physics tendency arrays.

modified routines: adiab/cpg.F90, adiab/cpg_gp.F90, adiab/cpg_end.F90, setup/sudim2.F90, utility/sc2rdt9.F90, utility/sc2wrt9.F90

new routines: adiab/gt92pt.F90, adiab/pt2gt9.F90

However, GT9 buffer is no longer used. New technical solution was required.

2. GMV/GFL solution - coding

In **type_gmvs** MCUPT, MCVPT, MCTPT and MCSPPT pointers are defined in TYPE_T9 (for U and V wind component, temperature and surface pressure).

In **gmv_subs** pointers are defined if LPC_FULL and LMPHYS. MCUPT, MCVPT, MCTPT belong to GMV. MCSPPT belongs to GMVS. All are defined for t-dt part of GMV structure, namely YT9.

In **yomgfl**, GFLPT is defined as a new main GFL array. It will contain the physics contributions to GFL variables.

In **type_gfls** new descriptors, attribute and pointer are defined:

NUMFLDSPT is the number of GFL variables that will have physics contribution stored in GFLPT.

NDIMPT is the dimension of GFLPT.

LPT is an attribute that is true if the phy. tend. of the variable is transferred using GFLPT.

MPPT is the pointer.

In **gfl_subs**, NUMFLDSPT, NDIMPT, LPT and MPPT are initialised in define_gfl_comp, LPT attribute is used as argument to set_gfl_attr where MPPT is defined (assigned a value) if LPT=TRUE.

In **susc2b**, GFLPT is allocated (if necessary).

In **gp_model**, abor1 is called if LPC_FULL and LMPHYS and not allocated GFLPT.

In **cpg**, GFLPT is used. New routine, cpg_pt stores or retrieves data from GFLPT.

3. GMV/GFL solution – usage

User might be confused by an army of dimensions, attributes, pointers etc. This is how it works. The following GFL attributes can be set through namelist:

CNAME ARPEGE field name

IGRBCODE GRIB code

LADV Field advected or not

LREQIN Field required in input

LGPINGP GP field input as GP

LGP Field exists and of grid-point type

LSP Field exists and of spectral type

LCDERS Derivatives required (spectral only)

LT9 Field in t-dt GFL

LT1 Field in t+dt GFL

LT5 Field in trajectory GFL

LSLP Field in S.L. physics GFL

LPT Field used in PC physics.

LCOUPLING True if field is coupled by Davies relaxation

CSLINT S.L interpolation "type"

This means (in theory) that each of these properties for each of the variables in the GFL structure can be set through namelist NAMGFL. For example if you want to store Q in PC physics to GFLPT so the physics tendency would be transferred from predictor to corrector, you (have to) set in the namelist:

```
NAMGFL

YQ_NL%LPT=.TRUE.,

/
```

The default for LPT for each variable is FALSE. NAMGFL is read in sudim1, called from su0yoma. sugfl and sudyn, setup routines where calls that define the GFL component and set-up its attribute are done later in the set-up stage. The default needs to be set before reading the namelist NAMGFL in sudim1.

An attribute like this is not available for GMV and GMVS variables. The physics tendencies for them are always used in the diabatic PC scheme. If one wishes not to be so, for one or more of these variables, new switches should be introduced to the code.

4. **Arrival/departure point combination**

   The code provided had a possibility of combining arrival and departure point physics tendencies using semi-lagrangian buffer 2 (SLB2). The GMV and GFL implementation does not support this solution. The SLBUF2 structure has been reduced, the previously stored copies of t-dt and t+dt quantities have been removed. The pointers MSLB2U1, MSLB2V1, MSLB2T1, MSLB2Q1, MSLB2L1, MSLB2I1 and MSLB2SP1 do not exist any more. Those quantities are now in GMVT1, GMVT1S and GFLT1.

   Examples:

   PB2(1,MSLB2U1) -> PGMVT1(1,1,YT1%MU)

   PB2(1,MSLB2V1) -> PGMVT1(1,1,YT1%MV)

   PB2(1,MSLB2T1) -> PGMVT1(1,1,YT1%MT)

   ...

   PB2(1,MSLB2Q1) -> PGFLT1(1,1,YQ%MP1)

   PB2(1,MSLB2L1) -> PGFLT1(1,1,YL%MP1)

   PB2(1,MSLB2I1) -> PGFLT1(1,1,YI%MP1)

   …

   PB2(1,MSLB2SP1) -> PGMVT1S(1,YT1%MSP)

   The code combining arrival and departure point physics tendencies is not in the public branch yet. The public branch code is not fully validated yet.

5. **One 'mysterious buffer' used in PC scheme**

A small part of the data flow in PC scheme is described here. It involves the data flow of the variables stored in GPNHBUF. This buffer was designed to store a few additional variables required by the old PC scheme. Now, the same buffer is used to transfer a few additional arrays used in PC scheme if the scheme is pseudo-second order decentered. This buffer survived the GMV/GFL revolution. However, its future existence is not certain.

cpg

`ZGPNH(:,:,:)` buffer is defined

If (LNHDYN and LPC_OLD) or LPC_NOTR or (LPC_FULL and LPC_XDIT)

ZGPNH(NPROMA,NFGPNH,KBLKS)is allocated

In call **cpg_gp** (passed as a single argument; called PGPNH)

1. this buffer is filled:

if (not LDLNHSITER)

   if (LPC_FULL and LPC_XDIT) and CDCONF=S)

   the PGPNH arrays are filled from GPNHBUF

if LDLNHSITER

   the PGPNH arrays are filled from GPNHBUF

```
      if LTRAJNH and LNHDYN and LPC_OLD

      call WRITTRAJM ( …,PGPNH(1,KGPN3Y), PGPNH(1,KGPNRR), PGPNH(1,KGPNGL),
      PGPNH(1,KGPNGM), PGPNH(1,KGPNWW),…)

      if LPC_NOTR

call ESTR_READ(.,.,PGPNH, PB2(1,MSLB2PCU1), PB2(1,MSLB2PCV1), PB2(1,MSLB2PCT1),
PB2(1,MSLB2PCPD1), PB2(1,MSLB2PCVD1), PB2(1,MSLB2PCSP1))
```

also for the old set-up of the PC scheme under LPC_OLD there is initialisation of auxiliary variables for case LDLNHSITER=T

CONCLUSION: depending on the PC configuration used in the run, the PGPNH array (ZGPNH in cpg) is filled by different variables

In the call **cpg_dyn** ZGPNH is passed as 10 arguments:

2. 4 belong to the old PC scheme `ZGPNH(1,IGPN3Y,ILL), ZGPNH(1,IGPNRR,ILL), ZGPNH(1,IGPNGL,ILL), ZGPNH(1,IGPNGM,ILL)`, these variables are PE3Y, PERR, PEGHL and PEGHM in cpg_dyn
3. and other 6 are used if LPC_FULL and LPC_XDIT: `ZGPNH(1,IGPNH_U,ILL), ZGPNH(1,IGPNH_V,ILL), ZGPNH(1,IGPNH_T,ILL), ZGPNH(1,IGPNH_PD,ILL), ZGPNH(1,IGPNH_VD,ILL), ZGPNH(1,IGPNH_SP,ILL)`, these variables are PCXNLT99 variables in cpg_dyn
4. the pointers IGPNH_* used in the new PC scheme are calculated in following part of the cpg:

```
IF ((LNHDYN.AND.LPC_OLD).OR.LPC_NOTR.OR.(LPC_FULL.AND.LPC_XIDT)) THEN

! * if lpc_notr=true, lpc_full=true or lpc_xidt=true:

! not yet coded for case lgwadv=true (in this case the

! buffer associated with IGPNH_VD contains interlayer data).

CALL ESTR_POINTERS(IGPNH,IGPN3Y,IGPNRR,IGPNGL,IGPNGM,IGPNWW

,IGPNH_U , IGPNH_V , IGPNH_T, IGPNH_PD, IGPNH_VD, IGPNH_SP

)

ENDIF
```

These fields could be moved to GMV and GMVS. In setup_t9, pointers MCXNLT99 should be defined (X stands for U, V, T, VD, PD etc.). The GMV structure should be introduced in the place of these variables in the code. Some old parts of the code, where variables are read from and written to GPNHBUF should be removed. Similar work is required for LPC_OLD and LPC_NOTR arrays.

APPENDIX A

GMV, GMVS and GFL arrays, dimensions, descriptors, attributes and pointers (from model documentation).

MODULE YOMGFL

GFL grid-point arrays

For the part of GFL that maps into spectral space see yomsp.F90 (SPGFL etc.)

All arrays have the layout (NPROMA, NFLEVG, "number of variables", NGPBLKS)

GFL- main GFL array holding t0 and t1 GFL variables

GFLT1- GFL array for t+dt quantities

GFLSLP- GFL array for use in semi-lagrangian physics

GFL5- trajectory GFL array

GFL_DEPART- used in 3-D FGAT (LIDMODEL)

GFLPT- used in diabatic predictor – corrector scheme

MODULE TYPE_GFLS

Derived types for describing the GFL structure. The descriptors themselves (YGFL and YGFLC) can be found in module yom_ygfl.

TYPE TYPE_GFLD

Overall descriptor, dimensioning etc.

NUMFLDS! Number of GFL fields

NDERS ! Number of horizontal derivatives fields

NUMSPFLDS ! Number of spectrally represented GFL fields

NUMGPFLDS ! Number of grid-point GFL fields

NUMFLDS9 ! Number of GFL fields in (t-dt) part

NUMFLDS1 ! Number of GFL fields in (t+dt) array

NUMSPFLDS1 ! Number of spectrally represented GFL fields (t+dt)

NUMFLDS5 ! Number of GFL fields (trajectory)

NUMFLDSSLP ! Number of GFL fields (S.L. phys.)

NUMFLDS_SPL ! Number of GFL fields (S.L. spline interpolation)

NUMFLDSPT ! Number of GFL fields in GFLPT (phy.tend.)

NDIM ! Dimension of main array holding GFL fields(GFL)

NDIM0 ! Dimension of t0 part of GFL

NDIM9 ! Dimension of t-dt part of GFL

NDIM1 ! Dimension of t+dt array (GFLT1)

NDIM5 ! Dimension of traj. GFL array (GFL5)

NDIMSLP ! Diminsion of S.L. phys. GFL array (GFLSLP)

NDIM_SPL ! Dim. of arrays holding GFL fields (S.L.spline int.)

NDIMPT ! Dimension of phy. tend. GFL array (GFLPT)

END TYPE TYPE_GFLD

TYPE TYPE_GFL_COMP

Individual field descriptor

CNAME ! ARPEGE field name

IGRBCODE ! GRIB code

LADV ! Field advected or not

LREQIN ! Field required in input

LGPINGP ! GP field input as GP

LGP ! Field exists and of grid-point type

LSP ! Field exists and of spectral type

LCDERS ! Derivatives required (spectral only)

LACTIVE ! Field in use

LT9 ! Field in t-dt GFL

LT1 ! Field in t+dt GFL

LT5 ! Field in trajectory GFL

LSLP ! Field in S.L. physics GFL

LPT ! Field in PC phy. tend. GFL (GFLPT)

LAM specific attributes (Arome/Aladin)

LADJUST0 ! True if field is thermodynamically adjusted at t (immediatly after inverse spectral transforms)

LADJUST1 ! True if field is thermodynamically adjusted at t+dt (after SL interpolations and NL residuals)

LCOUPLING ! True if field is coupled by Davies relaxation

LBIPER ! True if field must be biperiodised inside the transforms

End LAM specific attributes (Arome/Aladin)

CSLINT ! S.L interpolaion "type"

MP ! Basic field "pointer"

MPL ! zonal derivative "pointer"

MPM ! Meridional derivative "pointer"

MP9 ! Basic field "pointer" t-dt

MP1 ! Basic field "pointer" t+dt

MP5 ! Basic field "pointer" trajectory

MP5L ! zonal derivative "pointer" trajectory

MP5M ! Meridional derivative "pointer" trajectory

MPSLP ! Basic field "pointer" S.L physics

MPSP ! Basic field "pointer" spectral space

MP_SPL ! Basic field "pointer" spline interpolation

MPPT ! Phisics tendency “pointer”

POINTER :: **PREVIOUS** ! Pointer to previously def. field

END TYPE TYPE_GFL_COMP

TYPE TYPE_GFL_NAML

Individual field descriptor for namelist input

CNAME ARPEGE field name

IGRBCODE GRIB code

LADV Field advected or not

LREQIN Field required in input

LGPINGP GP field input as GP

LGP Field exists and of grid-point type

LSP Field exists and of spectral type

LCDERS Derivatives required (spectral only)

LT9 Field in t-dt GFL

LT1 Field in t+dt GFL

LT5 Field in trajectory GFL

LSLP Field in S.L. physics GFL

LPT Field used in PC physics.

LCOUPLING True if field is coupled by Davies relaxation

CSLINT S.L interpolaion "type"

END TYPE TYPE_GFL_NAML

END MODULE TYPE_GFLS

MODULE YOM_YGFL

Contains the descriptors of GFL arrays

INTEGER_M,PARAMETER:: **JPGFL**=105 ! Max number of GFL fields

INTEGER_M,PARAMETER:: **JPNAMED_GFL**=7 ! Number of currently pre-defined components of GFL

INTEGER_M :: **NGFL_EXT**

TYPE(TYPE_GFLD) :: **YGFL** ! GFL general descriptor, for info about content see comments in type declaration module : type_gfls.F90

TYPE(TYPE_GFL_COMP),TARGET:: **YGFLC(JPGFL)** ! General descriptor of all components

TYPE(TYPE_GFL_COMP),POINTER:: **YQ** ! Specific humidity

TYPE(TYPE_GFL_COMP),POINTER:: **YI** ! Ice water

TYPE(TYPE_GFL_COMP),POINTER:: **YL** ! Liquid water

TYPE(TYPE_GFL_COMP),POINTER:: **YA** ! Cloud fraction

TYPE(TYPE_GFL_COMP),POINTER:: **YO3** ! Ozone

! Extra fields

TYPE(TYPE_GFL_COMP),POINTER:: **YEXT**(:) ! Extra fields

! Fluxes

TYPE(TYPE_GFL_COMP),POINTER:: **YCPF** ! Convective precipitation flux

TYPE(TYPE_GFL_COMP),POINTER:: **YSPF** ! Stratiform precipitation flux

TYPE(TYPE_GFL_NAML) :: **YQ_NL** ! Specific humidity

TYPE(TYPE_GFL_NAML) :: **YI_NL** ! Ice water

TYPE(TYPE_GFL_NAML) :: **YL_NL** ! Liquid water

TYPE(TYPE_GFL_NAML) :: **YA_NL** ! Cloud fraction

TYPE(TYPE_GFL_NAML) :: **YO3_NL** ! Ozone

! Extra fields

TYPE(TYPE_GFL_NAML) :: **YEXT_NL(JPGFL-JPNAMED_GFL)** ! Extra fields

! Fluxes

TYPE(TYPE_GFL_NAML) :: **YCPF_NL** ! Convective precipitation flux

TYPE(TYPE_GFL_NAML) :: **YSPF_NL** ! Stratiform precipitation flux

END MODULE YOM_YGFL

MODULE GFL_SUBS

GFL_SUBS contains routines to do basic manipulations of GFL descriptors

SUBROUTINE **DEFINE_GFL_COMP**(YDGFLC,CDNAME,KGRIB,LDGP,LDREQIN, LDERS,LD5,LDT1,LDGPINGP)

Set-up individual GFL field: Basic allocation of GFL descriptor structure (on first call). Set-up basic attributes of individual GFL component.

YDGFLC - field handle

CDNAME - field ARPEGE name

KGRIB - GRIB code

LDGP - if TRUE grid-point field

LDREQIN - TRUE if required in input

LDERS - TRUE if derivatives required (only possible for spectral field)

LD5 - TRUE if field needs to be present in trajectory (T5)

LD1 - TRUE if field needs to be present in t+dt array (GFLT1)

SUBROUTINE **SET_GFL_ATTR**(YDGFLC,LDADV,LDT9,LDSLP,LDPT,LDADJUST0,
LDADJUST1,LDCOUPLING,LDBIPER,CDSLINT)

Add further attributes to GFL components previously set-up by call to DEFINE_GFL_COMP.

LDADV - TRUE if field to be advected

LDT9 - TRUE if field present in t-dt

LDSLP - TRUE if field present S.L. physics

LPT - TRUE if field used in PC physics.

LDADJUST0 - TRUE if field to be adjusted at t

LDADJUST1 - TRUE if field to be adjusted at t+dt

LDCOUPLING - TRUE if field to be coupled

LDBIPER - TRUE if field to be biperiodised

CDSLINT - S.L. interpolator

MODULE YOMGMV

Module containing t, t-dt, t+dt grid-point arrays (apart from GFL) for dynamics and all "pointers" for accessing elements in the arrays. GMV and GMVS are permanently allocated, GMVT1 and GMVT1S temporarily.

REAL_B,ALLOCATABLE :: **GMV(:,:,:,:)** ! Multilevel fields at t and t-dt

REAL_B,ALLOCATABLE :: **GMVS(:,:,:)** ! Single level fields at t snd t-dt

REAL_B,ALLOCATABLE :: **GMVT1(:,:,:,:)** ! Multilevel fields at t+dt

REAL_B,ALLOCATABLE :: **GMVT1S(:,:,:)** ! Single level fields at t+dt

REAL_B,ALLOCATABLE :: **GMV5(:,:,:,:)** ! Multilevel fields trajectory

REAL_B,ALLOCATABLE :: **GMV5S(:,:,:)** ! Single level fields trajectory

The following two fields are for 3-D FGAT (LIDMODEL)

REAL_B,ALLOCATABLE :: **GMV_DEPART (:,:,:,:)** ! Multilevel fields departure

REAL_B,ALLOCATABLE :: **GMVS_DEPART (:,:,:)** ! Single level fields departure

INTEGER_M :: **NDIMGMV** ! Third dim. of GMV (number of variables)

INTEGER_M :: **NDIMGMVS** ! Second dim. GMVS (number of variables)

TYPE(TYPE_T0) :: **YT0** ! Pointers to time t quantities

TYPE(TYPE_T9) :: **YT9** ! Pointers to time t-dt quantities

TYPE(TYPE_T1) :: **YT1** ! Pointers to time t+dt quantities

TYPE(TYPE_PH9) :: **YPH9** ! Pointers to physics time t-dt quantities

TYPE(TYPE_T0) :: **YT5** ! Pointers to trajectory quantities

TYPE(TYPE_AUX) :: **YAUX** ! Pointers to auxiliary SI-related quantities (NHS model)

END MODULE YOMGMV

MODULE TYPE_GMVS

TYPE TYPE_T0

Pointers to arrays at time t and some dimensions

NDIM – number of 3D fields in T0 part of GMV

NDIMS – number of 2D fields in T0 part of GMVS

NDIMUV – number of 3D fields describing wind components (depends on LVOR, LADER and LUVDER)

Field pointers in GMV

MU - U-component of the wind on layers at t

MV - V-component of the wind on layers at t

MT - temperature on layers at t

MTL - semi-reduced zonal derivative of the t temperature.

MTM - semi-reduced merid derivative of the t temperature.

MDIV - semi-reduced divergence "(M**2 D')" on layers at t.

MVOR - semi-reduced vorticity "(M**2 zeta')" on layers at t.

MUL - semi-reduced zonal derivative of U-wind at t.

MVL - semi-reduced zonal derivative of V-wind at t.

MSPD – pressure departure on layers at t

MSPDL – zonal derivative of pressure departure on layers at t

MSPDM – meridional derivative of pressure departure on layers at t

MSVD - vert.div. on layers at t

MSVDL – zonal derivative of vertical divergence on layers at t

MSVDM – meridional derivative of vertical divergence on layers at t

MEDOT - etadot at t.

Field pointers in GMVS

MSP – surface pressure at t

MSPL – zonal derivative of surface pressure at t

MSPM – meridional derivative of surface pressure at t

END TYPE TYPE_T0

TYPE TYPE_T9

Pointers to arrays at time t-dt and some dimensions

NDIM – number of 3D fields in T9 part of GMV

NDIMS – number of 2D fields in T9 part of GMVS

Field pointers in GMV

MU - U-component of the wind on layers at t-dt.

MV - V-component of the wind on layers at t-dt.

MT - temperature on layers at t-dt.

MTL - semi-reduced zonal derivative of the t-dt temperature.

MTM - semi-reduced merid derivative of the t-dt temperature.

MEDOT - etadot at t-dt.

MUNL - nonlinear term in U-equn at t-dt. (predictor)

MVNL - nonlinear term in V-equn at t-dt. (predictor)

MTNL - nonlinear term in T-equn at t-dt. (predictor)

MSPNL - nonlinear term in continuity equn at t-dt. (predictor)

MVCASRS "a/rs" on layers at t-dt. (input variable for SL3TL scheme) (input/output variable for SL2TL scheme)

MVWVNL - nonlinear term in "vwv"-equn at t-dt. (predictor)

MSPDNL - nonlinear term in "P-hat"-equn at t-dt. (predictor)

MYNL - nonlinear term in Y-corr. s.-i. at t-dt. (predictor)

MDIV - semi-reduced divergence "(M**2 D')" on layers at t-dt.

MVOR - semi-reduced vorticity "(M**2 zeta')" on layers at t-dt.

MUL - semi-reduced zonal derivative of U-wind at t-dt

MVL - semi-reduced zonal derivative of V-wind at t-dt.

MSPD – pressure departure on layers at t-dt

MSPDL – zonal derivative of pressure departure on layers at t-dt

MSPDM – meridional derivative of pressure departure on layers at t-dt

MSVD – vert.div. on layers at t-dt.

MSVDL – zonal derivative of vertical divergence on layers at t-dt

MSVDM – meridional derivative of vertical divergence on layers at t-dt

MSVDAUX – auxiliary variable "d3" for vert.div.(NVDVAR=4)

MCUNL - nonlinear term in U-eq. at t. (corrector)

MCVNL - nonlinear term in V-eq. at t. (corrector)

MCTNL - nonlinear term in T-eq. at t. (corrector)

MCSPNL - nonlinear term in continuity-eq. at t (corrector)

MCVWVNL - nonlinear term in "vwv"-eq. at t. (corrector)

MCSVDAUX - term d3+X-d4 at t used if d4 in NH. (corrector)

MCSPDNL - nonlinear term in "P-hat"-eq. at t. (corrector)

MNHX - term X at t-dt if d4 used in NH.

MCUPT – phy. tend. term in U-eq. at t. (corrector)

MCVPT - phy. tend. term in V-eq. at t. (corrector)

MCTPT - phy. tend. term in T-eq. at t. (corrector)

Field pointers in GMVS

MSP – surface pressure at t-dt

MSPL – zonal derivative of surface pressure at t-dt

MSPM – meridional derivative of surface pressure at t-dt

MVWVTOPNL – top nonlinear term in "vwv"-eq.

MCVWVTOPNL – top nonlinear term in "vwv"-eq. at t. (corrector)

MCSPPT - phy. tend. term of surface pressure (corrector)

END TYPE TYPE_T9

TYPE TYPE_T1

Pointers to arrays at time t+dt and some dimensions

NDIM – number of 3D fields in T1 part of GMV

NDIMS – number of 2D fields in T1 part of GMVS

Field pointers in GMVT1

MU – U-component of the wind on layers at t+dt

MV – V-component of the wind on layers at t+dt

MT – temperature on layers at t+dt

MSPD – pressure departure at t+dt

MSVD – vertical divergence at t+dt

MVOR – vorticity at t+dt

MDIV – divergence at t+dt

MSVDAUX – pointer to d3 at time "t+dt" in ZGT1AUX

Field pointers in GMVT1S

MSP – surface pressure at t+dt

MVWVTOP – zonal derivative of surface pressure at t+dt

END TYPE TYPE_T1

TYPE TYPE_PH9

Pointers to arrays at time "t-dt" adiabatic tendencies used as input for ECMWF physics

Field pointers in GMV

INTEGER_M :: **MU -** U-component of the wind

INTEGER_M :: **MV** – V-component of the wind

INTEGER_M :: **MT** – temperature

Field pointers in GMVS

INTEGER_M :: **MSP** – surface pressure

END TYPE TYPE_PH9

TYPE TYPE_AUX

Pointers to arrays for the NHS LAM auxiliary quantities

NDIM - dimension

Field pointers in GMV

MSVDAUX - pointer to d3 at time "t" in ZGPINAUX

MSVDAUXL - pointer to zonal der. of d3 at time "t" in ZGPINAUX

MSVDAUXM - pointer to merid.der. of d3 at time "t" in ZGPINAUX

END TYPE TYPE_AUX

END MODULE TYPE_GMVS

MODULE GMV_SUBS

SUBROUTINE SETUP_GMV

CALL **SETUP_T0**(YT0)

CALL **SETUP_T9**

CALL **SETUP_T1**

CALL **SETUP_PH9**

CALL **SETUP_AUX**

IF(NCONF/100 == 1.OR.NCONF == 801.OR.NCONF == 601.OR.

NCONF == 401.OR.NCONF == 501) THEN

CALL **SETUP_T0**(YT5)

ENDIF

END SUBROUTINE SETUP_GMV

APPENDIX B

Pointers MVGCHSS, MVGEVAP, MVGICFR, MVGIRUP, MVGSOUP, MVGTAUX, MVGTAUY and MVVALS were cleaned from mf_phys; from ‘USE PTRGPD’ and MSLB2VVEL from ‘USE PTRSLB2’ (variables were removed so the pointers were no longer used). These pointers were not cleaned from ptrgpd since they are used in updclidm.

Initialisation of PFPLCH, PFPLSH, PTCCH, PSCCH, PBCCHand PPBLH:

- PFPLCH is initialised if (LNEBN or LNEBR or LRRGUST) and KSTEP=0 because:
    - it is used in ACHMT if LRRGUST (called if LVDIF or LHMTO or LGWD),
    - is used in ACCOEFK if LRRGUST (called if (LVDIF or LGWD) and not LNEBR),
    - is used in ACNEBN as PFPLC (called if LNEBN) and
    - is used in ACNEBR if LNEBCO (called if LNEBR)

- PFPLSH is initialised if LRRGUST and KSTEP=0 because
    - it is used in ACHMT if LRRGUST (called if LVDIF or LHMTO or LGWD) and
    - is used in ACCOEFK if LRRGUST (called if (LVDIF or LGWD) and not LNEBR)

- PTCCH, PSCCH and PBCCH are initialized if (LNEBCO or LGWDC) and KSTEP=0 because
    - they are used in ACNEBT if LNEBCO (called if LNEBT),
    - diagnosed in ACNEBC called if LNEBCO
    - and used in ACDRAC called if LGWDC (except PTCCH).

- PPBLH is initialised if LNEBR and KSTEP=0 because
    - it is used in ACNEBR called if LNEBR and
    - diagnosed in ACPBLH called if LVDIF and LNEBCO and LNEBR

updating of PFPLCH and PFPLSH

- PFPLCH is updated at the end of APLPAR if LNEBN or LNEBR or LRRGUST
    - if LFPCOR with ZFPCOR (“precipitations plus lisse”)
    - otherwise with PFPLCL+PFPLCN (convective precipitations as rain and snow)

- PFPLSH is updated at the end of APLPAR if LRRGUST
    - with PFPLCL+PFPLCN (stratiform precipitations as rain and snow)

APPENDIX C

The new variables CPF and SPF are present in GFL and GFLT1. This might not be necessary. If at the end of mf_phys, new values stored in local variables would be copied directly to GFL instead of GFLT1, then they would not require t+dt representation, they would not be present in GFLT1 and would not ask for additional memory. The time-stepping done in scan2mdm does not require modification.

However, this implementation gives a ‘bus error’ in call_sl. It concerns the dimensioning of data structures involved in semi-lagrangian computations. The number of fields in GFL buffer is used when pointers for semilagrangian buffer 1 (SLB1) are calculated. If the fields are not advected, they should not be present in SLB1. The set-up of SLB1 is done in suslb. There you can find the following part of the code:

```
INUMGFL=YGFL%NUMFLDS

IF(INUMGFL > 0) THEN

MSLB1GFL9 = IPTX
```

```
IF (LLP) WRITE(IU,'(1X,A14,2I4)') 'MSLB1GFL9 = ',IPTX,ILVP2*INUMGFL

IPTX = IPTX + ILVP2*INUMGFL
```

Better solution would be

```
IF(INUMGFL > 0) THEN

MSLB1GFL9 = IPTX

IF (LLP) WRITE(IU,'(1X,A14,2I4)') 'MSLB1GFL9 = ',IPTX,ILVP2*INUMGFL

DO JGFL=1,INUMGFL

IF(YGFLC(JGFL)%LADV) IPTX = IPTX + ILVP2

ENDDO
```

(actually, that part with WRITE should also be modified to reflect the true size of GFL part of SLB1).

MSLB1GFL9 is also used later in the subroutine.

```
DO JGFL=1,INUMGFL

IGFL9=MSLB1GFL9+(JGFL-1)*ILVP2-1

RPARSL1(JF+IGFL9)=_ONE_

ENDDO
```

Should be

```
IGFL=0

DO JGFL=1,INUMGFL

IF(YGFLC(JGFL)%LADV) THEN

IGFL=IGFL+1

IGFL9=MSLB1GFL9+(IGFL-1)*ILVP2-1

RPARSL1(JF+IGFL9)=_ONE_

ENDIF

ENDDO
```

The modifications should be consistent through the whole set-up of SLB1. However, this is not enough. Numerous arrays in semilagrangian computations have the size set to YGFL%NUMFIELDS, meaning the total number of GFL fields, disregarding the fact that some of them might not be advected. In gp_model, call_sl and lapineb there are following arrays:

`PGFLT1(NPROMA,NFLEVG,YGFL%NUMFLDS,NGPBLKS)` in call_sl, this should be:

`PGFLT1(NPROMA,NFLEVG,YGFL%NDIM1,NGPBLKS)` and then the bus error in call_sl is avoided however, there are further problems.

`ZPBGFLP9(NPROMA,NFLEVG,YGFL%NUMFLDS,NGPBLKS)` in gp_model is

`PGFLP9 (NPROMA,NFLEVG,YGFL%NUMFLDS,NGPBLKS)` in call_sl, it is combined with

`ZGFL9(KPROMA,NFLEVG,YGFL%NUMFLDS)` in lapineb:

```
DO JGFL=1,YGFL%NUMFLDS

IF(YGFLC(JGFL)%LADV) THEN

PGFLT1(KSTART:KPROF,1:NFLEVG,YGFLC(JGFL)%MP1)=

PGFLT1(KSTART:KPROF,1:NFLEVG,YGFLC(JGFL)%MP1)+

ZGFL9(KSTART:KPROF,1:NFLEVG,JGFL)

ENDIF

ENDDO
```

etc.

The size of the arrays used in SL is YGFL%NUMFLDS. Therefore, new variables are kept in GFLT1. However, some further steps could be taken to reduce the memory requirements (and probably the computation cost). Number NUMFLDSADV could be defined, this one would be equal to the number of GFL fields that are advected. Then, all the arrays involved in advection would have the size YGFL%NUMFLDSADV.