Documentation of a new diagnostic dataflow for DDH

Olivier Riviere (GMAP/COOPE)
olivier.riviere@meteo.fr

16 décembre 2008

This documentation is valid for cy35t1 only and will be updated for cy35t2. At the time this documentation is being written some small bugfixes have being introduced so feel free to contact me if any doubt... It is complementary to the already existing DDH documentation.

1 Summary

A new coding approach has been proposed for extracting diagnostics from the Arome/MesoNH physical parametrisations. It can be used in other parts of the IFS/ARPEGE software. Physical quantities are recorded into a flexible data structure in the parametrisations, and readable by higher level routines. The data structure (a linked list of ad hoc Fortran 90 types) is automatically allocated and indexed as needed by low-level routines, so that physicists can freely choose which quantities they want to record, and how they want to process them. This technical approach greatly simplifies software clarity and maintenance.

Main applications are (1) to provide the ALADIN consortium with easy access to various Arome/MesoNH physical quantities at the level of the physics calling interface and (2) to replace existing DDH in Arpege/Aladin/Alaro if satisfying results are obtained after intensive testing.

This technical document is based on the proposal agreed by Aladin partners. Since cy35t1, the software is included in the common releases of the code and replaces the existing Arome's DDH.

2 Achievements-Future developments

The software is developped progressively and is expected to replace the existing DDH dataflow in the different models after a period of testing. User's feedbacks will be very important to trace potential weaknesses of the present code. Here is the timetable of foreseen code evolutions :

- cy35t1 : new dataflow available in Arome only for 3D fields. For Arpege/Aladin/Alaro, old DDH structures are kept.
- cy35t2 : new dataflow can be used in all models but by default old dataflow is used only in Arpege/Aladin/Alaro. 2D fields are available in the new dataflow.
- 2009 : intensive testing period whith expected improvements in the code. Renewing of DDH operators for horizontal averaging may be necessary.
- 2010 : complete switch to new dataflow ? (Would affect IFS code also...)

3 General basics of the new dataflow

This section describes the content of file xrd/module/ddh_mix.F90 which contains all the functionalities of the new dataflow. It can be thought as an externalized functionality of the code.

3.1 Description

The dataflow consists in self allocatable structures similar to GFL but more flexible. This subsection describes how they are defined, the possible architecture of the code being discussed in section 5. Each extracted quantity (variable, flux, tendencies...) will be characterized through a Fortran 90 structure type (named here DDHFLEX) which defines several attributes corresponding to this quantity.

The structure type named DDHFLEX is given here :

```
TYPE DDHFLEX
  CHARACTER(LEN=11)::CNAME !name of field
  CHARACTER(LEN=1)::CFLUX !'F' if flux 'V' if variable 'T' if tendency
  CHARACTER(LEN=3)::CMOD  ! 'ARP','ARO': name of model
  LOGICAL:: LKDDH !TRUE if to be stored into DDH
  ! rfield has to be a pointer because allocatable not allowed in structure type
  REAL(KIND=JPRB),DIMENSION(:,:),POINTER:: RFIELD  ! value of retrieved field
  INTEGER(KIND=JPIM):: NFIELDIND! position of flux in ddh array
END TYPE DDHFLEX
```

Following attributes are used :

- CNAME is the name of the field as it will appear in the output file. CNAME has to respect the following conventions :
  - First letter has to be either 'F' for a flux, 'V' for a variable or 'T' for a tendency.
  - Second and third letter describes the conservation equation to which the budget applies (see DDH documentation for details) : CT (temperature), QV (water vapour), ...
- CFLUX is a sting that informs about the nature of the quantity stored in the structure :
  - CFLUX='F' for a flux
  - CFLUX='T' for a tendency
  - CFLUX='V' for a variable
- CMOD gives information on the model's name
  - CMOD='ARO' for AROME
  - CMOD='ARP' for ARPEGE, ALADIN and ALARO (by default but if you wish other labels can be introduced)
- LKDDH is a flag set to .TRUE. if the field has to be processed by DDH operators and to .FALSE. otherwise.
- RFIELD is a pointer corresponding to the value of the field (it will be explained later why it has to be a pointer)
- NFIELDIND is an integer that gives the number of the processed field within the list of all fields.

These attributes are important because they document the structure content itself (important for debugging purposes) and they determine which operations the extracted field will undergo at the place where it is recorded, before being stored (for instance conversion from potential temperature to temperature...)

The various extracted fields are gathered into an allocatable array of structure of type DDH, called here RDDH_DESCR and whose last dimension corresponds to the total number of extracted fields :

```
TYPE(DDHFLEX),ALLOCATABLE,DIMENSION(:):: RDDH_DESCR
```

The attribute *allocatable* being forbidden inside a type structure, the field is not directly stored inside RDDH_DESCR but defined through a pointer to a large array called RDDH_FIELD :

```
REAL,DIMENSION(:,:,:),ALLOCATABLE,TARGET::RDDH_FIELD ! target of RFIELD
! first two dims are the same as PFIELD, the third being the number of stored fields
```

3.2 Extracting a field from the physics

For adding a field into the diagnostics, you only need to call subroutine ADD_FIELD_3D and that's all! The first argument of ADD_FIELD_3D will be the field to store and the others will correspond to the associated attributes (for instance "call ADD_FIELD3D(field_to_store,'name_of_field','F','CT'....)")

Arguments of ADD_FIELD_3D(PMAT,CDNAME,CDFLUX,CDMOD,LDINST,LDDH) are the following :

- PMAT : the array to be stored. It has to be with levels in the same order than in Arpege part of the code. If you are in a .mnh subroutine just use subroutine INVERT_VLEV.MNH before calling ADD_FIELD_3D in order to have levels ordered as in Arpege.
- CDNAME : name of field. It is constructed the following way :
  - CDNAME(1) : 'F' if flux ,'T' if tendency,'V' if variable
  - CDNAME(1 :2) : type of variable ('CT','QI','QV','QR',...)
  - CDNAME(3 :) : name of process
- CDFLUX : 'F' if flux ,'T' if tendency,'V' if variable
- CDMOD : 'ARO' if AROME, 'ARP' otherwise (but you may add some other label if you wish)
- LDINST :'TRUE' if instaneous field
- LDDH :'TRUE' if field is stored to be in DDH

When using add_field_3D it is extremely important to have the right attributes in the right order. So be careful! Have a look at xrd/module/ddh_mix.F90 if any doubt.

Here are some examples :

```
CALL ADD_FIELD_3D(ZTMPAF,'VQI','V','ARP',.TRUE.,.TRUE.)

CALL ADD_FIELD_3D(ZTMPAF(:,:),CLNAME,'T','ARP',.TRUE.,.TRUE.)

CALL ADD_FIELD_3D(PFRSO(:,:,1),'FCTRAYSO','F','ARP',.TRUE.,.TRUE.)
```

ADD_FIELD performs the following tasks :

- when in the code a specific field is supplied as argument for the first time in the execution, the last dimension of arrays DDH_FIELD and DDH_DESCR is incremented in order to add space for the new field to store. The code determines if a field is encountered for the first time by testing the field's name. This reallocation of arrays may slow the code and fragment memory during the first time step, but it avoids going through complicated setups. One could also preallocate the arrays according to a first guess of the dimensions, as chosen by the user.

– at every time step the field is stored in RDDH_FIELD through the pointer RFIELD
– at every time step, some transformations are done on the field according to its nature (and documented by its attributes), for instance conversion from θ to T... These operations also depend on the physics used (Meso-NH, Arpege...). Here it will be possible for users to add parts corresponding to specific needs, and to document them through attributes.

4 User's guide

4.1 Using DDH products included in documentation

The DDH documentation helds as a reference for the formulation of the budget equations and for the list of terms present by default in the DDH files. If you just need these products, just set the DDH namelist according to your need and you just have to plot the ddh files using the ddhtoolbox.

4.2 Adding terms to the already existing DDH products

You just have to call ADD_FIELD_3D (Make sure that you have imported this function by adding in your file USE DDH_MIX,ONLY :ADD_FIELD_3D) If you want to add a term to an existing budget equation, just use the same name for the variable ('CT','QR'...) than in the rest of the code. Otherwise you are free to introduce a new name. If you are in a .mnh subroutine, you have to proceed in two steps :

– First you have to transform your array on NLEV+2 levels to an array on NLEV levels in reverse order (to go from the "MNH" word to "Arpege" word). There is a subroutine dedicated to this transformation INVERT_VLEV.MNH
– Then use ADD_FIELD_3D on the transformed array.

4.3 Using the dataflow for extracting terms from the physics but not for DDH

It is possible by just setting LDDH to .FALSE. when calling ADD_FIELD_3D to use the flexible dataflow for retrieving fields out from the physics and use them elsewhere. Once the field is stored using ADD_FIELD_3D, you just have to go through the flexible structure once to have the index MYINDEX of your field that you can use later on by accessing RDDH_FIELD(:, :,MYINDEX) :

```
DO II=1,NTOTFIELD
```

```
  IF (RDDH_DESCR(II)\%CNAME=='MYNAME') THEN
   MYINDEX=RDDH_DESCR(II)\%NFIELDINF
  ENDIF
ENDDO

% your field is stored in RDDH\_FIELD(:,:,MYINDEX)
```

For the time being the previous lines of code are not in the common cycles, if you feel that there should be just send an email to the DDH team.

4.4 Plotting results

Use the ddhtoolbox : see DDH's main documentation for details.

5 Architecture of the code

Subroutine ADD_FIELD_3D and associated modules are in xrd/module/ddh_mix.F90. This subroutine contains all elements for using the new dataflow.

However the use of the new dataflow in the part of the Arome code originating form Méso-NH required some interfacing described in the following subsection.

5.1 Application to DDH in Arome

The new dataflow is used in Arome since cy35t1 for DDH diagnostics. Méso-NH code already uses its own diagnostics through the sophisticated budgets and advantage is taken form the work already performed there in order to avoid duplication of effort. MNH's budgets are called through the call of the subroutine budget after each process. This subroutine is able to perform operations on the stored quantity. In order to keep the maximum level of compatibility between MNH and Arome code, it was chosen to keep the calls to budget unchanged in the Méso-NH code and to write a new budget subroutine that would be called in Arome instead of the budget from MASDEV. This subroutine, located in /mpa/micro/externals, supresses first and last level of MNH fields and reverses the order of the vertical levels.

5.2 Organization of the data flow

The DDH diagnostic facility performs some domain averaging and budget computation after the diagnostic extraction. These operations are performed at each

timestep, after the physics computations, so that the raw recorded fields are accessible as NPROMA packets at the level of APLPAR/APL_AROME, where they may be used for other purposes.

For the DDH domain averaging, the Arpege subroutine cpcuddh.F90 (see DDH documentation for more details) is used and averaged fields are then written into file in ppfidh.F90 (which will be simplified since now with the self-documented structure, a loop on all elements in DDH_DESCR can generate the names of the fields to be written into the DDH file). The subroutine cpcuddh.F90 uses arrays (hdcvbx stored in module yomtddh) whose size is computed in setups (the total number of fluxes/tendencies depend on the options used for physics). Since these setups are no longer used with the new data flow, these arrays are allocated with an estimated size (larger than expected value) for the time being but we are thinking at a way to have them reallocated or initialized elsewhere in the code after a dummy call to the code that only computes the total size of DDH arrays (like the call to stepo from cnt4.F90 if CFU/XFU diagnostics are switched on).

Figure 1 summarizes the new data flow (which is the same for Arpege and Arome) within a time step.

6 Application to DDH in Arome

6.1 Architecture of the code

In Arome there are two different ways to have terms in the DDH products. The first is to use ADD_FIELD_3D after a call INVERT_VLEV as shown previously. The second possibility is to use the budgets from Méso-NH as in the first version of the DDH code in Arome. We have used a combination of the two methodes in order to take advantage of the validation performed by the Méso-NH team of the budget packages.

- Variables are stored in cpdyddh.F90 using ADD_FIELD_3D since the part of the code is common with Arpege/Aladin/Alaro.
- Within APL_AROME, adjustment and radiation are retrieved using ADD_FIELD_3D and other processes through budgets from Méso-NH.

Interfacing with Méso-NH budgets works the following way :

- ARO_SUINTBUDGET stores quantities (Exner function...) necessary to transform tendencies from Méso-NH (in θ,r variables) to tendencies in (T,q) variables into the module MODDB_INTBUDGET
- ARO_STARTBU stores initial values of tendencies for each variable
- Within Méso-NH, subroutine "BUDGET" is called. The BUDGET subroutine from Méso-NH is replaced by a new subroutine (/mpa/micro/internals/-budget.mnh) called the same way with the same arguments that transforms

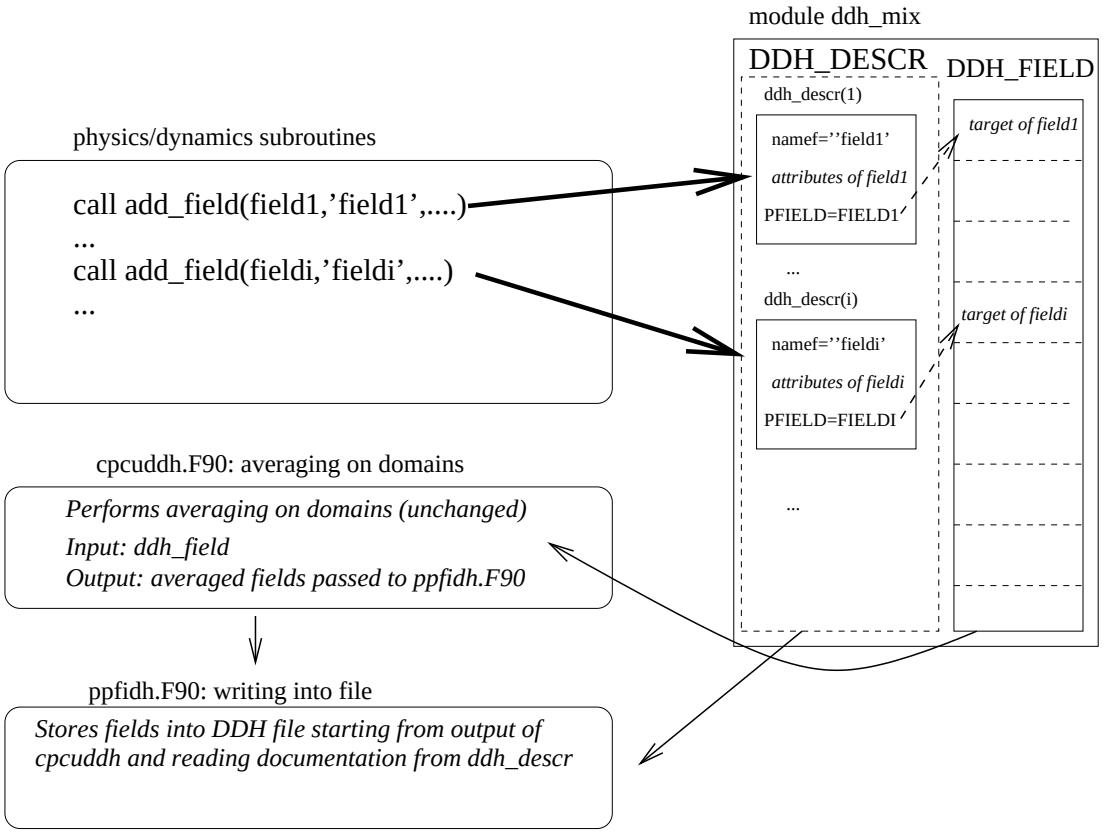

FIG. 1 – Organization of the data flow within a time step. Subroutine ADD_FIELD stores the field and the associated description into DDH_DESCR after possible transformations (bold arrows). Averaging on the domains is performed as in Arpege in cpcuddh.F90, the output being written into file in ppfidh.F90 using the description of the fields stored as attribute in DDH_DESCR.

tendencies of Méso-NH variables (θ...) to tendencies on the desired variables (c_pT...) and skips the Méso-NH processing.

6.2 Maintenance

If the budget package in Méso-NH is mantained there is nothing to do for mantaining the code except in the followng situations :

- **New species are added in Arome.**
  - In this case, a label for it first has to be introduced.
    If this is an hydrometeor you have to add an entry to CLVARNAME in APL_AROME (it corresponds to the names of hydrometeors ordered the same way than in PTENDR) and report it coherently in MODDB_INTBUDGET. Increase also by one dimension of TAB_VARMULT array. **Beware to use the same ordering of variable than in Méso-NH calls to budget ! ! !**
    If this is not an hydrometeor, it may not be present in the Méso-NH budgets and thus we recommend to use combination of INVERT_VLEV and ADD_FIELD_3D.
  - The transformation applied to this field has to be defined.
    In ARO_SUINTBUDGET, increase by one the last dimension of TAB_VARMULT and have it pointing on the TCON2 (equal to PQDM) since it is an hydrometeor.
  - In APL_AROME, check that loops on last dimension on PTENDR include this new hydrometer.
  - In CPDYDDH just use ADD_FIELD_3D to add the value of the variable corresponding to your new hydrometeor.
- **Order of subroutines is changed in APL_AROME.** In this case make sure that ARO_STARTBU and ARO_SUNINTBUDGET are called at the right place.

6.3 Validation

Validation has been performed in Arome on 1D (thanks to Sylvie Malardel) and 3D cases using cy35t1. Two small bugs in cy35t1 have been corrected. 1D cases allow to have closed budgets (since dynamical terms are zero) and easier interpretation of physical processes.

7 Remaining issues specific to the new dataflow

Some issues are still to be dealt with in the new dataflow :

- Performances. If faster in Arome than the old code, there is still room for improvement in terms of computational performances.
- OpenMP. The code has to be tested and optimized for OpenMP parallelization. For the time being, validation has only been done on the NEC platform from Météo-France.
- Elarging the flexibility of the code to the DDH operators for domain averaging. Some arrays like PDDHCV_TOT still have to be initalized at the beginning of the time step and thus we don't fully benefit from the flexibility of the new dataflow. Thinking about how to deal with that without affecting the part of he code used by ECMWF is ongoing.

8 Conclusion

This new version of the dataflow offers not only more facilities to add new quantities in the diagnostics but also more flexibility in terms of possible uses of these diagnostics. For developers, since the new code is considerably smaller and readable than the current one in Arome, it will be easier to debug and maintain when physics evolve in the future. We also expect an increase in the code's performance for Arome's DDH since the Meso-NH budgets part of the code (with a lot of unused (in Arome) options slowing the code) will be skipped. An another important aspect is that this tool, after being successfully implemented in Arome can be used in Arpege/Aladin [1]. Before going on with further work to upgrade this prototype version towards a beta version, discussion between the different possible users of this type of diagnostics is needed in order to raise possible new issues and needs regarding what different users would like these structures to offer.

[1] In fact it is easier to implement in Arpege than in Arome