STATUS OF DIFFERENT INTERPOLATORS CODED IN ARPEGE/IFS CY39: TOWARDS AN UNIFICATION AND EXTERNALISATION?

Karim YESSAD

September 10, 2012

Version 8.

1 Introduction.

This paper tries to do an inventory of the different interpolators (containing horizontal interpolations, and optionally vertical interpolations) in the code of ARPEGE/IFS. The vertical interpolator used in FULL-POS and in the observation interpolator is not referenced here. Additional remarks will be provided, and we will try to specify the main features required in a "universal" interpolator if such one could be coded in the future.

Modifications compared to the previous version of this document:

- Reference cycle is pre-CY39.

2 List of interpolators.

∗ **Two main approaches:** To sum-up we can say that interpolators can be coded according two main approaches.

Approach 1/ In the first one, weights are computed separately for each direction: zonal, meridian, and vertical if required. Interpolations do first zonal interpolations, then meridian interpolations, and finally when required vertical interpolations. Vertical interpolations may be done first in very specific cases (interpolator LAITVSPCQM for example). This approach is used in the semi-Lagrangian interpolator.

Approach 2/ In the second one, all weights are computed at the same time, and the zonal or meridian character of weights is lost. Interpolations simply do multiplications by weights, but at the stage of interpolations the information about zonal or meridian type of interpolation is lost. This approach is used in the FULL-POS horizontal interpolator, and it would be probably the right approach for interpolators used in coarse mesh physics (coarse grid does not depend on the timestep; horizontal interpolations only).

What is the best approach? it depends on what we want to do.

- The first approach reduces the number of multiplications, especially if there are vertical interpolations, and is recommended when weights must be computed several times (for example at each timestep). It is particularly recommended in the following case: weights which must be recomputed at each timestep, and 3D interpolations (for example 32 points interpolations). Example: interpolations used in the semi-Lagrangian interpolator.
- The second approach increases the number of multiplications and weights, but this shortcoming is not too misleading if there are only 4 points or 12 points horizontal interpolations. It is particularly recommended in the following case: weights which must be computed once (in the set-up), set of weights with land-sea mask and set of weights without land-sea mask, horizontal interpolations only. Example: interpolations used in the FULL-POS horizontal interpolator.

* **Interpolators used in ARPEGE/IFS:** This table gives routines involved in the calculation of interpolation grid, weights, and interpolations.

Application	Kind of approach	Interpol grid calcul.	Weights calcul.	Interpolations	Available for LELAM?
Semi-Lag	1	(E)LASCAW	(E)LASCAW	LAITRI,LAITLI LAIDDI,etc	yes
Observation horizontal interpolator	1	(E)LASCAW	(E)LASCAW	LAIDDIOBS LAIDLIOBS LAIDLIC	yes
EC radiation on coarse grid NRADINT=1	1	RDSCAW	RDSCAW	LAIDDIOBS LAIDLIOBS	not yet
EC physics on coarse grid	2	SUCOAPHY	SUCOAPHY	code in EC_PHYSG	not yet
FULLPOS	2	FPSCAW	SU(E)HOW1 SU(E)HOW2 SU(E)HOWLSM	FPINT12 FPINT4	yes
Conf 923	2	(E)INTER..	(E)INTER..	(E)INTER..	yes
H2L and L2H in ECMWF 4DVAR	1	GRID_BICONSERV GRID_BICUBIC GRID_BILINEAR			no
Obs pre-processing	1	SUHIFCE	SUHIFCE	SUHIFCE	no
SURFEX	2	HORIBL_SURF			yes

Additionally, some of these interpolators require a halo calculation (setup: SLCSET+SLRSET; memory communications in SLCOMM.. + (E)SLEXTPOL).

Application	Set-up	Communications COMM	Communications EXTPOL
Semi-Lag	SLCSET +SLRSET	SLCOMM +SLCOMM2 +SLCOMM2A	(E)SLEXTPOL
Observation horizontal interpolator	SLCSET +SLRSET	SLCOMM +SLCOMM2 +SLCOMM2A	(E)SLEXTPOL
EC radiation on coarse grid NRADINT=1	SLCSET +SLRSET	SLCOMM +SLCOMM2 +SLCOMM2A	(E)SLEXTPOL
EC physics on coarse grid	SLCSET +SLRSET	SLCOMM +SLCOMM2 +SLCOMM2A	(E)SLEXTPOL
FULLPOS	SLCSET +SLRSET	SLCOMM +SLCOMM2 +SLCOMM2A	(E)SLEXTPOL
Conf 923	none	none	none
H2L and L2H in ECMWF 4DVAR	none	none	none
Obs pre-processing	none	none	none
SURFEX	none	none	none

∗ Code architecture for interpolators if weights must be redefined at each timestep: We often find the following structure (example: semi-Lagrangian scheme):

- Set-up:
  - Define a halo width N..WIDE and call to SLCSET+SLRSET in the set-up to compute quantities allowing to fill the halo.

  Code body:
  - Fill interpolator buffers.
  - Compute halo (call to SLCOMM-type routines and to (E)SLEXTPOL).
  - Compute the position of interpolation points.
  - Compute the interpolation grid and weights.
  - Do interpolations.

∗ Code architecture for interpolators if weights must be computed once in the set-up: We often find the following structure (example: horizontal interpolator of FULL-POS):

- Set-up:
  - Define a halo width N..WIDE and call to SLCSET+SLRSET in the set-up to compute quantities allowing to fill the halo.
  - Compute the position of interpolation points.
  - Compute the interpolation grid and weights.

  Code body:
  - Fill interpolator buffers.
  - Compute halo (call to SLCOMM-type routines and to (E)SLEXTPOL).
  - Do interpolations.

3 Some ideas to have an universal externalised interpolator.

3.1 Weights calculation and interpolations.

From we have seen at the previous section, we can see that there are too many interpolators which do things which are not so different, and the risk is that people code new interpolators each time they code a new application requiring interpolators. It is time to think about a sort of externalised library containing a limited list of interpolators for all purposes (which may be implemented in a couple of years).

We can say that this library will contain:

- routines computing interpolation grid and weights for the first approach.
- routines computing interpolation grid and weights for the second approach.
- some routines doing interpolations for the first approach.
- some routines doing interpolations for the second approach.

About weights we need at least (but this list is not comprehensive):

- the possibility to take account or not of diffusive interpolations (SLHD).
- the possibility to take account or not of the land-sea mask or of the land ratio.

Routines like (E)LASCAW, SU(E)HOW1, SU(E)HOW2 and SU(E)HOWLSM can be re-used.

About interpolations we need at least (but this list is not comprehensive):

- bilinear interpolations.
- 12 points interpolations (with or without quasi-monotonic option).
- pseudo-interpolation taking the closest grid-point value.
- trilinear interpolations.
- 32 points interpolations (with or without quasi-monotonic option).
- 3D interpolations with continuously differentiable vertical interpolations (for example VFE-spline like in LAITVSPCQM).

The code of routines like LAIDDI, LAIDDIOBS, LAIDLIC, LAIDLI, LAIDLIOBS, LAITLI, LAITRI, LAITVSPCQM, FPINT4, FPINT12, can be re-used.

3.2 Halo calculation and communications.

Additionally, halo calculation routines may be also externalised. Merging SLCOMM, SLCOMM2 and SLCOMM2A is desirable. After that, the following routines can be externalised: SLCSET, SLRSET, SLCOMM, SLCOMM2, SLCOMM2A, SLEXTPOL, ESLEXTPOL, SLEXTPOLAD.

3.3 Steps for externalisation.

- Already done in CY39:
  - Move the "externalisable interpolator" routines in a new directory in the main library (ifs/interpol and ald/interpol).
  - YOMLUN quantities: use YOMLUN_IFSAUX version.
  - YOMCST quantities: use YOMCST_IFSAUX version.
  - KPDUP (YOMLASCAW) changed into JPDUP (EINT_MOD).
  - RFAA to RFDD, YRVSLETA, RIPI, RSLD, RSLDW, R3DTW moved in EINT_MOD. Set-up of these quantities is now in "ifs/interpol" routines.
  - Preliminary other actions to reduce forbidden dependencies.
  - Preliminary cleanings in (E)LASCAW.. to make the code more compact (remove useless KWIS=102 in particuliar).
- Following steps: go towards externalisation (targets CY40, CY41). Work can go into two directions: introduce new structures for weights, and remove forbidden dependencies. In particuliar:
  - Quantities used only in the interpolator: move them in EINT_MOD (example: MTAGSLAG, NADMAP, RSASIGN).
  - Some quantities used both in the externalisable part and in the non-externalisable part can be duplicated (one version in EINT_MOD).
  - Purely semi-Lagrangian features must not be hard coded in LASCAW itself, if possible (for example on-demand communications).
  - For interpolation routines, a sort of external interface must be written (we must have few external routines).
- Further steps: rewrite (E)LASCAW in a more flexible way according to future requirements:
  - Keep KWIS or not?
  - Find a way to add new weights without an excessive lengthening of LASCAW.
  - New data structure for weights.
  - Simpler calculation of KL0 and KLH0; can we find something in order that LASCAW can be used with a more flexible ordering in input array PB1?
  - Number of rows in each side of interpolation point: find a simple way to implement possibility of having more than 2 rows (use of variable NSTENCILWIDE).
  - Is there a possibility to unify the use of different geometries (spherical geometry, plane projection, irregular or regular zonal/meridian spacing)?
  - Keys controlling SLHD: do simplifications.
  - Constraints on vertical coordinate: keep the current one or allow more flexible ones?
  - TL and AD codes must be rewritten too.
- Further steps: complementary tasks:
  - ECMWF radiation: merge RDSCAW and LASCAW (use the new LASCAW).
  - Observation interpolator: rename SLINT_CANARI into LASCAWLSM (does similar actions as SUHOWLSM), externalise LASCAWLSM.
  - Ensure homogeneous variable denotation between the different routines (example: KST and KPROF for the bounds of horizontal points treated, PXSL for input array to be interpolated, PXF for output interpolated array). That may help for further routines reunification.
  - Obs pre-processing: rewrite SUHIFCE as a call to externalised LASCAW and calls to externalised LAIDLI.
  - ECMWF physics on coarse grid: rewrite SUCOAPHY by calling externalised FPSCAW and SUHOW... routines. Interpolations done in routine EC_PHYSG must be done by calling externalised interpolation routines (extended versions of externalised FPINT4 or FPINT12).

  - H2L and L2H in ECMWF 4DVAR: rewrite the GRID_.. routines by calling externalised routines for grid calculation, weights calculation and interpolations. Approach 1 or 2 can be used: if approach 2 is chosen, rewrite the GRID_.. routines by calling externalised FPSCAW and SUHOW... routines, and calling externalised interpolation routines (extended versions of externalised FPINT4 or FPINT12).

- Conf 923: wait and see; some parts may become obsolete during the following years, and the remaining part may be candidate for externalisation in project UTI.

4 More discussions about rewriting LASCAW.

4.1 Why rewriting this routine.

The current design of this routine dates from years 1991-1992 and is outdated. The current design, valid for the 90's ARPEGE/IFS environment, is written to match the following constraints:

- reduce the number of DO loops (but with the code evolutions since 1992 some DO-loops have been splitted).
- identify a subset of interpolations combinations (different values for variable KWIS); for each value of KWIS, do calculations in a minimal number of DO loops.
- limited choice of interpolations.
- use limited to the semi-Lagrangian scheme.
- design rather adapted to vectorial machines.

Shortcomings of this design:

- not easy to introduce new interpolations.
- too linked with semi-Lagrangian scheme.
- too many repetitions of same pieces of code.
- too long code.

The new design we want to give to this routine must be compliant with the following constraints.

- possibility to use it in different parts of the code requiring interpolations.
- possibility to use it in pieces of code external to ARPEGE/IFS.
- no forbidden dependency (when passing variables).
- introduction of new weights must be easy.
- no hard coded feature linked to semi-Lagrangian scheme; in particuliar no hard coded feature linked to LSLONDEM (LSLONDEM has no sense for observation or radiation interpolator).
- some flexibility for vertical coordinate; no explicit reference to "pressure coordinate"; according to choices enabled, calculation of vertical weights may be more or less expensive.
- some flexibility about horizontal geometry (in particuliar muse we unify plane/spherical geometry?).
- avoid repetition of the same pieces of code.
- good performances on different machines, in particuliar scalar machines.

There are some points which require more discussion.

4.2 Horizontal geometry.

Two kinds of horizontal geometries are currently considered:

- LASCAW: spherical geometry, with the following assumptions:
  - coordinates are defined by latitudes, longitudes.
  - grid-points are zonally regularly spaced.
  - grid-points are meridionally slightly-irregularly spaced.
  - interpolation point is not bounded.
  - meridian aligment on longitudes is not necessarily ensured.
- ELASCAW: plane projection geometry, with the following assumptions:
  - coordinates are defined by abscissa, ordinate.
  - grid-points are zonally regularly spaced.

- grid-points are meridionally regularly spaced.
- interpolation point is bounded.
- meridian aligment on longitudes is ensured.

That means that for example users which want use a grid with irregular zonal spacing cannot use any of the routines LASCAW or ELASCAW; additionally only a subset of users has access to ELASCAW.

Do we want to have more flexibility in the future?

4.3 Vertical coordinate.

Vertical coordinate used for vertical interpolations may currently comply with the following assumptions:

- monotonous.
- equal to 0 at the top.
- equal to 1 at the bottom.
- not horizontally dependent.

These constraints allow to use a cheap algorithm to retrieve model level just above interpolation point, by using the intermediate array NVAUTF.

Do we want to have more flexibility in the future (use a vertical coordinate not complying with the above assumptions)?

4.4 KWIS or not KWIS?

Keeping KWIS combined to some other keys, or completely abandoning KWIS, still requires some discussion and will depend on developments expected in the next 10 years.
The advantage of having blocks under KWIS is to separate the different uses of LASCAW:

- trajectory research in a 3D model with semi-Lagrangian advection.
- treatment of equations RHS in a 3D model with semi-Lagrangian advection.
- algorithms requiring only horizontal interpolations (choice between bilinear or bicubic interpolations).

Shortcoming to use KWIS is some code repetition; it limits the number of combinations we want to have.

4.5 Data ordering in input array PB1.

LASCAW is currently too much linked to the way we order data in input array PB1, and to the current uses of halos. People wanting to use LASCAW in a simple environment are discouraged to use LASCAW because they must deal with SLCSET, SLRSET, SLCOMM and so on. Features currently hard coded in LASCAW, allowing to compute arrays such KL0 and KLH0, must probably be moved outside of LASCAW.

5 Conclusion.

There are currently a lot of interpolators in the ARPEGE/IFS code and we have tried to list them; we have tried to define what could be an externalised all-purpose interpolator in the future.
Tasks must be spread among years 2012-2015; they need to know what new possibilities must be available in the next 10 years. Externalisation must be completed if possible before 2015. CY41 would be a good target for that.