FULL-POS IN THE CYCLE 46T1 OF ARPEGE/IFS.

YESSAD K. (METEO-FRANCE/CNRM/GMAP/ALGO)

December 19, 2018

Abstract:

This documentation describes the software FULL-POS doing post-processing on different kind of vertical levels. In particular, post-processable variables and organigramme are given. Some aspects of horizontal and vertical interpolators (which may be used in some other applications) are also described.

Résumé:

Cette documentation décrit le logiciel de diagnostics FULL-POS, qui fait du post-traitement sur différents types de niveaux verticaux. On y décrit en particulier la liste des variables post-traitables et l'organigramme. Certains aspects des interpolateurs horizontaux et verticaux (susceptibles d'être utilisés dans d'autres applications) y sont également décrits.

Contents

1 Introduction about post-processing.

Post-processing can be done on pressure levels, height levels, potential vorticity levels, potential temperature levels, temperature levels, flight levels or η-levels. Each variable is defined by a sequence of letters [X] which appears in the name of some variables of **YOMAFN**. For example X=T for upper air temperature.

The extensive list of post-processable variables is no longer given in this documentation. Reader can look at routines such YOMAFN, SUAFN1, SUAFN2 or SUAFN3 to know this list. There are five groups of post-processable variables:

- 3D dynamical fields (DYN3D): includes some GMV or GFL variables.
- 2D dynamical fields (DYN2D).
- Surface physical fields (PHYSOL).
- Surface cumulated fluxes (CFU).
- Surface instantaneous fluxes (XFU).

Remarks:

- Some of them are diagnosed at ECMWF via some PHYSOL quantities, and at METEO-FRANCE via some XFU quantities: this is the case of several "VDIAG" surface variables (example: the high, medium and low cloud cover).
- The same quantity can be sometimes found in the topics DYN2D, PHYSOL and XFU (example: the wind-components at 10 m).
- Some of them are used only at ECMWF (example: the boundary layer dissipation).
- Some of them are used only at METEO-FRANCE (example: the A, B, C climatological ozone profiles).
- Some of them have a different generic name at METEO-FRANCE and ECMWF. Example: surface snow albedo ([ALSN] in PHYSOL at METEO-FRANCE, [ASN] in PHYSOL at ECMWF).
- Some of the PHYSOL variables are variables of the surface data flow, but not always with the same generic name. For example for the surface snow albedo one uses the following generic names: [A] in the surface dataflow, [ALSN] in PHYSOL-pp at METEO-FRANCE, [ASN] in PHYSOL-pp at ECMWF.

Some denotations:

- L: number of layers of the model.
- $\left(\frac{dT}{dz}\right)^{\text{st}}$: standard atmosphere vertical gradient of the temperature in the troposphere (0.0065 K/m).
- R_{d}: dry air constant.
- g: gravity acceleration.

Configurations 1, 903, 904:

- Post-processing (off-line, in-line, change of resolution) can be done via configuration 1.
- For off-line post-processing, configuration 903 can be also used.
- For off-line post-processing dedicated to resolution change, configuration 904 can be also used.

2 Horizontal interpolations.

Horizontal interpolations can be bilinear interpolations or 12 points cubic interpolations.

2.1 Bilinear horizontal interpolations.

2.1.1 Horizontal interpolation grid and weights for bi-linear interpolations.

A 16 points horizontal grid is defined as it is shown in figure 2.1. The interpolation point O is between B_1, C_1, B_2 and C_2. Λ and Θ are the longitudes and latitudes on the computational sphere (departure geometry). The following weights are defined as follows:

- zonal weight number 1:
$$ZDLO1 = \frac{\Lambda_O - \Lambda_{B_1}}{\Lambda_{C_1} - \Lambda_{B_1}}$$
- zonal weight number 2:
$$ZDLO2 = \frac{\Lambda_O - \Lambda_{B_2}}{\Lambda_{C_2} - \Lambda_{B_2}}$$
- meridian weight:
$$ZDLAT = \frac{\Theta_O - \Theta_{B_1}}{\Theta_{B_2} - \Theta_{B_1}}$$

2.1.2 Bilinear interpolation.

For a quantity X, are computed successively:

- a linear interpolation on the longitude number 1:
$X_1 = X_{B_1} + ZDLO1(X_{C_1} - X_{B_1})$.
- a linear interpolation on the longitude number 2:
$X_2 = X_{B_2} + ZDLO2(X_{C_2} - X_{B_2})$.
- a meridian linear interpolation:
$X_{interpolated} = X_1 + ZDLAT(X_2 - X_1)$.

In the FULL-POS code the weights are pre-computed in routines **SUHOW2** and **SUHOWLSM**, so splitting between zonal and meridian interpolations is not visible in the interpolation routines.

2.2 12 points horizontal interpolations.

2.2.1 Horizontal interpolation grid and weights for 12 points cubic interpolations.

A 16 points horizontal grid is defined as it is shown in figure 2.2. The interpolation point O is between B_1, C_1, B_2 and C_2. The following weights are defined as follows:

- zonal weight number 0:
$$ZDLO0 = \frac{\Lambda_O - \Lambda_{B_0}}{\Lambda_{C_0} - \Lambda_{B_0}}$$
- zonal weight number 1:
$$ZDLO1 = \frac{\Lambda_O - \Lambda_{B_1}}{\Lambda_{C_1} - \Lambda_{B_1}}$$
- zonal weight number 2:
$$ZDLO2 = \frac{\Lambda_O - \Lambda_{B_2}}{\Lambda_{C_2} - \Lambda_{B_2}}$$
- zonal weight number 3:
$$ZDLO3 = \frac{\Lambda_O - \Lambda_{B_3}}{\Lambda_{C_3} - \Lambda_{B_3}}$$
- meridian weights:
$$ZCLA1 = \frac{(\Theta_O - \Theta_{B_0})(\Theta_O - \Theta_{B_2})(\Theta_O - \Theta_{B_3})}{(\Theta_{B_1} - \Theta_{B_0})(\Theta_{B_1} - \Theta_{B_2})(\Theta_{B_1} - \Theta_{B_3})}$$
$$ZCLA2 = \frac{(\Theta_O - \Theta_{B_0})(\Theta_O - \Theta_{B_1})(\Theta_O - \Theta_{B_3})}{(\Theta_{B_2} - \Theta_{B_0})(\Theta_{B_2} - \Theta_{B_1})(\Theta_{B_2} - \Theta_{B_3})}$$
$$ZCLA3 = \frac{(\Theta_O - \Theta_{B_0})(\Theta_O - \Theta_{B_1})(\Theta_O - \Theta_{B_2})}{(\Theta_{B_3} - \Theta_{B_0})(\Theta_{B_3} - \Theta_{B_1})(\Theta_{B_3} - \Theta_{B_2})}$$

2.2.2 Horizontal 12 points interpolation.

Let us define:

- $f_1(\alpha) = (\alpha+1)(\alpha-2)(\alpha-1)/2$
- $f_2(\alpha) = -(\alpha+1)(\alpha-2)\alpha/2$
- $f_3(\alpha) = \alpha(\alpha-1)(\alpha+1)/6$

For a quantity X, are computed successively:

- a linear interpolation on the longitude number 0:
  $X_0 = X_{B_0} + ZDLO0(X_{C_0} - X_{B_0})$.
- a cubic 4 points interpolation on the longitude number 1:
  $X_1 = X_{A_1} + f_1(ZDLO1)(X_{B_1} - X_{A_1}) + f_2(ZDLO1)(X_{C_1} - X_{A_1}) + f_3(ZDLO1)(X_{D_1} - X_{A_1})$.
- a cubic 4 points interpolation on the longitude number 2:
  $X_2 = X_{A_2} + f_1(ZDLO2)(X_{B_2} - X_{A_2}) + f_2(ZDLO2)(X_{C_2} - X_{A_2}) + f_3(ZDLO2)(X_{D_2} - X_{A_2})$.
- a linear interpolation on the longitude number 3:
  $X_3 = X_{B_3} + ZDLO3(X_{C_3} - X_{B_3})$.
- a meridian cubic 4 points interpolation:
  $X_{interpolated} = X_0 + ZCLA1(X_1 - X_0) + ZCLA2(X_2 - X_0) + ZCLA3(X_3 - X_0)$.

In the FULL-POS code the weights are pre-computed in routines **SUHOW2** and **SUHOWLSM**, so splitting between zonal and meridian interpolations is not visible in the interpolation routines.

2.2.3 Multilinear interpolations.

More diffusive multi-linear interpolations are also available. Combine the above 4 or 12 points interpolations with:

- bilinear interpolations using points (B_0, C_0, B_3, C_3).
- bilinear interpolations using points (A_1, D_1, A_2, D_2).
- bilinear interpolations using points (A_0, D_0, A_3, D_3).

2.3 Location of computations.

Once the coordinates of the interpolation points known:

- The "north-western" model grid point coordinates are computed in the routine **SUHOW1**. This is the model (departure geometry) grid point which is immediately at the north-west of the interpolation point.
- The weights not modified by the land-sea mask are computed in routine **SUHOW2**.
- The weights modified by the land-sea mask are computed in routine **SUHOWLSM**. This is equivalent to use weights not modified by the land-sea mask and to multiply the field to be interpolated by the land-sea mask (0 if sea, 1 if land).
- The horizontal bilinear interpolations are done by routine **FPINT4**.
- The horizontal 12 points interpolations are done by routine **FPINT12**.
- Horizontal interpolations need intermediate quantities computed in routine **FPSCAW**.

Additional modifications can be done after 12 points interpolations: they are done by routine **FPHOR12**, for example add a monotonicity condition. More details about these additional modifications and the post-processed fields concerned by these modifications are described in the part describing each routine.

2.4 Plane geometry (LAM models).

All previous formulae for weight computation can be used for an irregular latitude spacing and a different number of points on each longitude. The LAM grid has a horizontal regular spacing, so the previous formulae can be simplified. **SUEHOW1**, **SUEHOW2** and **SUEHOWLSM** are called instead of **SUHOW1**, **SUHOW2** and **SUHOWLSM**.

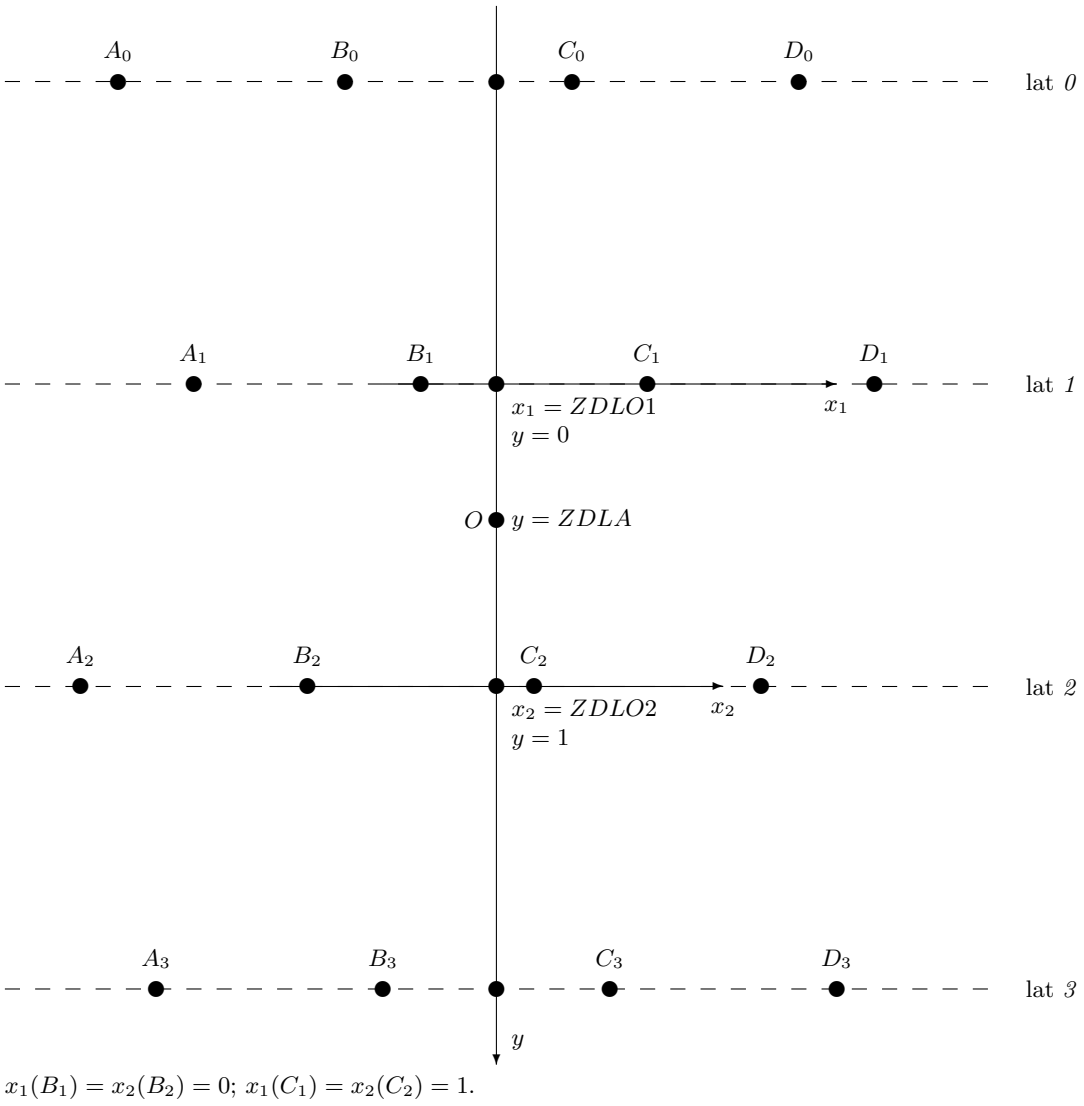
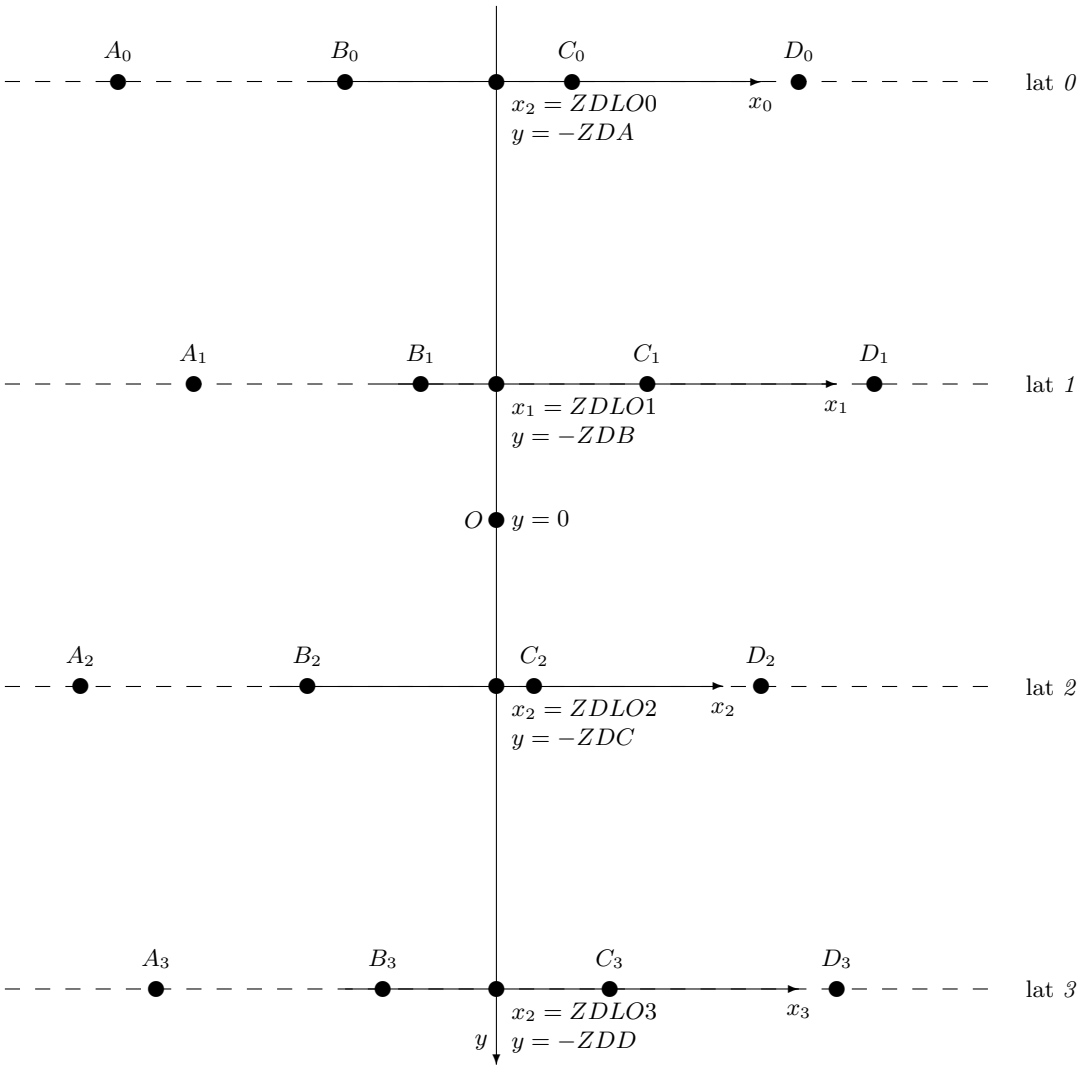


Figure 2.1: Interpolation horizontal grid for bilinear interpolations.



$x_0(B_0) = x_1(B_1) = x_2(B_2) = x_3(B_3) = 0$; $x_0(C_0) = x_1(C_1) = x_2(C_2) = x_3(C_3) = 1$.

Figure 2.2: Interpolation horizontal grid for 12 points interpolations.

3 Vertical interpolations and extrapolations.

3.1 General considerations.

For 3D variables to be vertically interpolated, vertical interpolations are generally linear interpolations between the layers where are defined model variables. The treatment of the extrapolations above the upper layer, the extrapolations below the lower layer or the surface depend on the variable considered. In particular cases some variables can be diagnosed using the vertically interpolated value of some other variables.

The "**OLD**" versions of routines when available are used at ECMWF (key **LOLDPP**=.T.). METEO-FRANCE uses the option **LOLDPP**=.F. .

3.2 More details for 3D dynamical variables.

3.2.1 Wind components, wind velocity.

∗ Way of interpolating if LOLDPP=.T.:

- Linear interpolation between the layer 2 and the lower layer.
- The coordinate used for linear interpolation is the logarithm of the pressure.
- Quadratic interpolation between the layer 1 and the layer 2 using the values of the layers 1, 2 and 3.
- Quadratic interpolation between the top and the layer 1 using the values of the top, layers 1 and 2; the value of the top is obtained by a linear extrapolation from the values of the layers 1 and 2.
- The coordinate used for quadratic interpolation is the logarithm of the pressure.
- Extrapolation below the middle of the lower layer and below the surface assumes that the quantity is constant.

∗ **Way of interpolating if LOLDPP=.F.:** The same as for **LOLDPP**=.T. but quadratic interpolations are replaced by linear interpolations.

3.2.2 Temperature.

Applies to temperature if the vertical coordinate of post-processing is not the potential vorticity, otherwise see routine **PP2DINT**.

∗ Way of interpolating if LOLDPP=.F. (routine PPT):

- Quadratic interpolation between the middles of the upper and lower layers.
- Quadratic interpolation between the top and the middle of the upper layer: the top value of the temperature is assumed to be equal to the value of the middle of the upper layer; due to the fact that the interpolation is a quadratic one, that does not mean that the temperature is constant in this atmosphere depth.
- The coordinate used for quadratic interpolation is the logarithm of pressure. For more details about the quadratic interpolation used, which is a quadratic analytic expression of the logarithm of pressure, and the reason of using a quadratic interpolation, see (Undén, 1995).
- A surface temperature T_{surf} is computed as follows:

$$T_{\text{surf}} = T_{Le} + \left(\frac{dT}{dz}\right)^{\text{st}} \frac{R_{\text{d}}}{g} \left(\frac{\Pi_{\text{s}}}{\Pi_{Le}} - 1\right) T_{Le} \tag{1}$$

  Le is the level number defined by variable **NLEXTRAP**, matching with height **HEXTRAP**.
- Extrapolation below the middle of the lower layer and the surface is a linear interpolation between T_L and T_{surf}.
- Extrapolation under the surface is done according a more complicated algorithm:

$$T_{\text{extrapolated}} = T_{\text{surf}} \left(1 + y + \frac{y^2}{2} + \frac{y^3}{6}\right) \tag{2}$$

  where:

$$y = \Gamma \frac{R_{\text{d}}}{g} \log\left(\frac{\Pi_{\text{s}}}{\Pi_L}\right) \tag{3}$$

Expression of Γ: $\Gamma = \left(\frac{dT}{dz}\right)^{\rm st}$ if $\Phi_{\rm s}/g < 2000$ m; if $\Phi_{\rm s}/g \geq 2000$ m expression of Γ is more complicated:

$$\Gamma = \frac{g}{\Phi_{\rm s}} \max(T_0^{'} - T_{\rm surf}, 0) \tag{4}$$

Expression of $T_0^{'}$:

- if $\Phi_{\rm s}/g > 2500$ m:

$$T_0^{'} = \min\left(T_{\rm surf} + \left(\frac{dT}{dz}\right)^{\rm st} \frac{\Phi_{\rm s}}{g}, 298{\rm K}\right) \tag{5}$$

- if $\Phi_{\rm s}/g \leq 2500$ m and $\Phi_{\rm s}/g \geq 2000$ m: $T_0^{'}$ is computed by a linear interpolation (coordinate of interpolation is $\Phi_{\rm s}$) between the two values $\min\left(T_{\rm surf} + \left(\frac{dT}{dz}\right)^{\rm st} \frac{\Phi_{\rm s}}{g}, 298{\rm K}\right)$ and $T_{\rm surf} + \left(\frac{dT}{dz}\right)^{\rm st} \frac{\Phi_{\rm s}}{g}$.

∗ Way of interpolating if LOLDPP=.T. (routine PPT_OLD):

- Linear interpolation (between the upper and the lower layer).
- The coordinate used for linear interpolation is the pressure.
- Extrapolation above the middle of the upper layer assumes that the quantity is constant.
- Extrapolation below the middle of the lower layer and the surface is a linear interpolation between T_L and $T_{\rm surf}$ like in **PPT**.
- Extrapolation under the surface is done according the same algorithm as in **PPT** (code of part 1.4 is different in **PPT** and in **PPT_OLD** but actually does the same calculations).

3.2.3 Geopotential height.

Applies to geopotential height (and Montgomery geopotential) if the vertical coordinate of post-processing is not the potential vorticity, otherwise see routine **PP2DINT**.

∗ Way of interpolating if LOLDPP=.T.:

- The variable interpolated is a geopotential height departure from a reference defined by a standard atmosphere without any orography. After the interpolation an increment is added, sum of the surface orography and the "standard" geopotential height depth between the pressure level of interpolation and the actual surface. This method avoids to introduce interpolations for the standard component of the geopotential height which can be computed analytically (in routine **PPSTA**).
- Linear interpolation between the layer 2 and the surface.
- The coordinate used for linear interpolation is the logarithm of the pressure.
- Quadratic interpolation between the layer 1 and the layer 2 using the values of the layers 1, 2 and 3.
- Quadratic interpolation between the top and the layer 1 using the values of the top, layers 1 and 2.
- The coordinate used for quadratic interpolation is the logarithm of the pressure.
- Extrapolation below surface uses the surface temperature $T_{\rm surf}$ of equation (1).

$$gz_{\rm extrapolated} = \Phi_{\rm s} - R_{\rm d} T_{\rm surf} \log\left(\frac{\Pi_{\rm s}}{\Pi_L}\right)\left(1 + \frac{y}{2} + \frac{y^2}{6}\right) \tag{6}$$

  where y is defined by formula (3) with $\Gamma = \left(\frac{dT}{dz}\right)^{\rm st}$ in all cases.
- For more details about this algorithm, see (Andersson and Courtier, 1992) which is still valid for the cycle 46t1.

∗ Way of interpolating if LOLDPP=.F.:

- The variable interpolated is a geopotential height departure from a reference defined by a standard atmosphere without any orography. After the interpolation an increment is added, sum of the surface orography and the "standard" geopotential height depth between the pressure level of interpolation and the actual surface. This method avoids to introduce interpolations for the standard component of the geopotential height which can be computed analytically (in routine **PPSTA**).
- Quadratic interpolation between the middles of the upper and lower layers.
- Quadratic interpolation between the top and the middle of the upper layer.

- The coordinate used for quadratic interpolation is the logarithm of pressure. The quadratic interpolation is not exactly the same as for **LOLDPP**=.T., it is a quadratic analytic expression of the logarithm of pressure of the same type as the one used to post-process the temperature for **LOLDPP**=.F. . For more details about the quadratic interpolation used, and the reason of using a quadratic interpolation, see (Undén, 1995).
- Linear interpolation between the lower layer and the surface, as for **LOLDPP**=.T. .
- Extrapolation below surface uses the same algorithm as for **LOLDPP**=.T. .

3.2.4 Variables interpolated using routine PP2DINT.

∗ List of variables:

- Geopotential height gz if vertical coordinate is potential vorticity.
- Temperature T if vertical coordinate is potential vorticity.
- Relative vorticity ζ.
- Divergence D.
- Potential temperature Θ if vertical coordinate is not potential temperature.
- Virtual potential temperature Θ_{v} if vertical coordinate is not potential temperature.
- Velocity potential χ.
- Stream function ψ.
- Equivalent potential temperature Θ_{e}.
- Absolute vorticity $\zeta + f$.
- Stretching deformation STD.
- Shearing deformation SHD.
- Potential vorticity PV.
- True vertical velocity w for non hydrostatic model (defined at half levels).

∗ Way of interpolating:

- Linear interpolation (between the upper and the lower layer for quantities defined on the middle of layers, between the layer 1 and the surface for quantities defined at half levels).
- The coordinate used for linear interpolation is the pressure.
- Extrapolation above the middle of the upper layer assumes that the quantity is constant.
- Extrapolation below the middle of the lower layer and below the surface assumes that the quantity is constant.

3.2.5 Post-processable GFL variables (like moisture) and pressure departure variable: variables using routine PPQ.

∗ Way of interpolating (routine PPQ):

- Linear interpolation (between the upper and the lower layer).
- The coordinate used for linear interpolation is the pressure.
- Extrapolation above the middle of the upper layer assumes that the quantity is constant.
- Extrapolation below the middle of the lower layer and below the surface assumes that the quantity is constant.

This method is also applied to relative humidity.

3.2.6 Pressure coordinate vertical velocity ω (routine PPVVEL).

∗ Way of interpolating (routine PPVVEL):

- Linear interpolation (between the upper and the lower layer).
- The coordinate used for linear interpolation is the pressure.
- Extrapolation above the middle of the upper layer is a linear interpolation between a zero value at the top and the value of the upper layer.
- Extrapolation between the middle of the lower layer and the surface assumes that the quantity is constant.
- Extrapolation below the surface assumes that the quantity is zero.

3.2.7 Moist (irreversible) pseudo-adiabatic potential temperature $\Theta'_{\rm w}$ (routine PPTHPW).

Routine **PPTHPW** is a diagnostic one. It takes as input the vertically post-processed pressure, temperature, moisture, liquid water and ice and computes $\Theta'_{\rm w}$ at the post-processing levels using a diagnostic (and rather complicated) algorithm.

3.2.8 Vertical divergence variable (currently d).

One uses the vertically post-processed vertical velocity w (done by **PP2DINT**) then diagnoses the post-processed vertical divergence (for both **NVDVAR**=3 or 4).

3.3 2D dynamical variables which need extrapolations.

3.3.1 Mean sea level pressure $\Pi_{\rm MSL}$ (routine PPPMER).

If $\mid \Phi_{\rm s} \mid$ is lower than 0.001 J/kg the mean sea level pressure is set to the surface pressure. In the other cases one uses the following algorithm:

- One computes the surface temperature $T_{\rm surf}$ of equation (1) and the " mean sea level " temperature $T_0 = T_{\rm surf} + \left(\frac{dT}{dz}\right)^{\rm st} \frac{\Phi_{\rm s}}{g}$.
- To avoid extrapolation of too low pressures over high and warm surfaces the following modifications are done:
  - if $T_0 > 290.5$ K and $T_{\rm surf} \leq 290.5$ K, Γ is defined by:
$$\Gamma = (290.5 - T_{\rm surf}) \frac{g}{\Phi_{\rm s}} \tag{7}$$
  - if $T_0 > 290.5$ K and $T_{\rm surf} > 290.5$ K, Γ is set to 0, $T_{\rm surf}$ is modified and set to 0.5*(290.5 K + old value of $T_{\rm surf}$).
- To avoid extrapolation of too high pressures over cold surfaces the following modifications are done when $T_{\rm surf} < 255$ K: Γ is set to $\left(\frac{dT}{dz}\right)^{\rm st}$ and $T_{\rm surf}$ is modified and set to 0.5*(255 K + old value of $T_{\rm surf}$).
- In the other cases Γ is set to $\left(\frac{dT}{dz}\right)^{\rm st}$.
- Mean sea level pressure is computed as follows:
$$\Pi_{\rm MSL} = \Pi_{\rm s} \exp\left[\frac{\Phi_{\rm s}}{R_{\rm d} T_{\rm surf}} \left(1 - \frac{x}{2} + \frac{x^2}{3}\right)\right] \tag{8}$$
where:
$$x = \frac{\Gamma \Phi_{\rm s}}{g T_{\rm surf}} \tag{9}$$

4 Post-processing files.

- Departure file (**CFNISH**= ICMSH[code]INIT) is read on logical unit number **NINISH**=81.
- Departure geometry climatology is read on logical unit **NULCL1**=10 (file Const.Clim).
- Arrival geometry climatology is read on logical unit **NULFP01**=54 (file Const.Clim.DDDDDD).
- For NFPOS=1 or 2 post-processing fields are written on logical unit number **NULFP01**=54 (file **CPFPN**=PF[code]DDDDDD+0000), where DDDDDD is the content of the character variable **CFPDOM** of **YOMFPC**.

5 Filtering in spectral space.

5.1 General considerations.

- two filters are available for global models: a low-pass filter (also called THX filter because it uses a tanh function) and a bell-shaped filter (also called Gaussian filter). Variable **LFPBED** allows to choose the filter (THX filter if **LFPBED**=T).
- derivatives in stretched global models: require to do a filter which is equivalent to filter on an equivalent unstretched geometry.
- filtering is done in routines **SPOS** (low-pass filter) and **SPOSGF** (Gaussian filter).
- namelist tunable variables are generally in **NAMFPF**.

∗ Low-pass filter in global models:

This function looks like a smoothed step function; for a given total wavenumber n the formula is:

$$f_{\mathrm{THX}}(n) = \frac{1 - \tanh(e^{-k}(n - n_0))}{2} \tag{10}$$

It means that this function equals roughly 1 if n is less than n_0, and 0 if it is bigger than n_0.

- tunable parameter n_0 is stored in variable **NFMAX**.
- tunable parameter k is stored in variable **RFPBED**.

∗ Bell-shaped filter in global models:

For a given total wavenumber n the formula is:

$$f_{\mathrm{bsARP}}(n) = e^{-4\frac{k}{2}(n/\min(n_0, N_{\mathrm{s}}))^2} \tag{11}$$

where N_{s} is the model truncation (**NSMAX**) and k a tunable variable stored in variable **RFPBED**.

∗ Bell-shaped filter in LAM models:

For a given pair of wavenumbers (n, m), the formula is:

$$f_{\mathrm{bsAL}}(n, m) = e^{-6\frac{k}{2}((n/\min(n_0, N_{\mathrm{s}}))^2 + (m/\min(n_0, N_{\mathrm{ms}}))^2)} \tag{12}$$

where N_{s} is the model meridian truncation (**NSMAX**), N_{ms} is the model zonal truncation (**NMSMAX**), and k a tunable variable stored in variable **RFPBED**.

5.2 Filtering in the code.

∗ Fields which can be filtered in computational space:

Filtering is done in routine **SPOS** or **SPOSGF**: multiplication of array **PSPBFP** (containing post-processed data to be filtered) by array **RFPFIL** (in **SPOS**) containing the filtering operator. In **SPOSGF** the filtering operator is hard-coded (no storage in **RFPFIL**).

∗ Fields which must be filtered in an equivalent not-stretched space:

This case occurs in case of stretched global geometry, to filter some "derivative fields". It is available for low-pass filter. This filter is active if:

- Variable **NFMAX** is smaller than the "equivalent unstretched sphere" truncation N_{c} (in practical N_{c} is between $1.1 * c * N_{\mathrm{s}}$ and $1.2 * c * N_{\mathrm{s}}$, where c is the stretching factor).

Filtering is done as follows:

- array **PSPBFP** is multiplied by array **RFPMAT** containing the filtering operator.
- **RFPMAT** contains $\mathcal{C} * f * \mathcal{D}$ and is computed in routine **CPFPFILTER**. f denotes the filter; $\mathcal{D}$ and $\mathcal{C}$ are dilatation and contraction matrices (they can be computed by the formerly numbered configuration 911, now in "utilities", see the corresponding documentation (IDRD); they can also be computed internally in **RDFPFILTER**).
- Remarks about this matricial operator:
  - It is possible to store this matricial operator on a file by setting **LFPWRFIL**=.T. in **NAMFPF**.
  - This file can be read by setting **LFPRDFIL**=.T. in **NAMFPF**.

6 Organigramme.

Call-tree is very complex and some parts are not described extensively.

6.1 FULL-POS via configuration 1.

6.1.1 Setup routines and call tree above STEPO_FPOS.

∗ General architecture under CNT0 (only FULL-POS features are mentioned):

```
CNT0 ->
* IFS_INIT -> for example SUCT0, SUCT1, SUCST, SUPPVI
* SU0YOMA -> for example SUGEOMETRY -> (call tree not detailed)
* SU0YOMB ->
  - SUPHY -> SUMTS
  - SUCFU -> SUFPCFU
  - SUXFU -> SUFPXFU
  - SUBFPOS ->
    * General configuration of post-processor:
      - SUFPC -> (call tree not detailed)
      - SUFPCNT
    * Horizontal geometries aspects:
      - UPDTRANS -> (call tree not detailed)
      - SUFPMODELGEO
      - SUFPGEOMETRY -> (call tree not detailed)
      - SUFPSC2_DEP
      - SUFPSC2
      - SUFPD -> (call tree not detailed)
      - SUFPG
      - SUFPUSERGEO -> (call tree not detailed)
      - SUFPEZO
      - SUFPWIDE -> SLCSET
      - SUFPWFPBUF -> SU(E)HOW1, SU(E)HOW2, SU(E)HOX1, FPSCAW, FPSCAX
    * Associated vertical eta coordinate:
      - SUFPV
      - SUFPVERT
    * IO handlings:
      - SUFPIOS
      - SUFPIOH -> (call tree not detailed)
    * Fields descriptors:
      - SUAFN -> SUAFN1, SUAFN2 and SUAFN3 (call tree not detailed)
    * Spectral filters:
      - SUFPF
      - SUFPFILTERS -> (E)FPFILTER, RDFPFILTER, CPFPFILTER, WRFPFILTER, etc.
  - IO_SERV_SUIOSCTMPL -> (call tree not detailed)
  - FP_SERV_SUIOSCTMPL -> (call tree not detailed)
* CNT1 -> (see below call tree under CNT1)
```

∗ General architecture under CNT1 (only FULL-POS features are mentioned):

```
CNT1 ->
* SU1YOM ->
  - SUINIF ->
    * SUSPEC -> (call tree not detailed)
    * SUGRIDF, SUGRIDO and SUGRIDU -> (call tree not detailed)
    * SUGRCFU and SUGRXFU -> (call tree not detailed)
* CNT2 -> CNT3 ->
  - CNT4 ->
    * SUINIF_FP (fullpos IO server, lower call tree not detailed)
    * FULLPOS_DRV ->
      - SU4FPOS -> PPREQ, SUFPC, SUFPFIELDS, SUFPPHY and SUFPDYN (lower call tree not detailed)
      - SUPPDATE
      - SUFPDATA ->
        * SUFPRFPBUF_CLIM (lower call tree not detailed)
        * SUFPSUW -> CPCLIMI, SU(E)HOW1, SU(E)HOWLSM, SUFPCIP (lower call tree not detailed)
      - ALLFPOS -> (see below call tree under ALLFPOS)
      - DEALLOC_FPFIELDS
    * SIGPOST
    * FP_SERV_SYNC (fullpos IO server, lower call tree not detailed)
```

6.1.2 ALLFPOS, VARFPOS, GRIDFPOS, DYNFPOS.

∗ General architecture under ALLFPOS:

```
ALLFPOS ->
* SUFPOFNAME
* SUFPTR2
* FPMODCFU, FPMODXFU, FPMODPREC -> (call tree not detailed)
* MAXGPFV
* GRIDFPOS -> (see below call tree under GRIDFPOS)
* SUFPOROG -> (call tree not detailed)
* FP2SX1, FP2SX2 -> (call tree not detailed)
* FPGPNORM, FPSPNORMS -> (call tree not detailed)
* SUVPOS
* CPVPOSPR
* STEPO_FPOS -> (see below call tree under STEPO_FPOS)
* SUVFPOSL
* ESPFP -> (call tree not detailed)
* INI2WRFP
* WRSFP and WRHFP (ARPEGE files), WRMLFP and WRPLFP (GRIB files) -> (call tree not detailed)
```

∗ General architecture under VARFPOS:

```
VARFPOS ->
* SUFPTR2
* FPMODCFU, FPMODXFU, FPMODPREC -> (call tree not detailed)
* GRIDFPOS -> (see below call tree under GRIDFPOS)
* SUFPOROG -> (call tree not detailed)
* FPGPNORM -> (call tree not detailed)
* SUVPOS
* CPVPOSPR
* STEPO_FPOS -> (see below call tree under STEPO_FPOS)
* SUVFPOSL
```

∗ General architecture under GRIDFPOS:

```
GRIDFPOS ->
* HPOSSFX
* SCAN2M_HPOS -> HPOS, HPOS_CFU, HPOS_XFU
* FPTENSOR
* FPHALO
* processor communication routines SLCOMM and (E)SLEXTPOL
* FPOSHORPHY -> (see below call tree under FPOSHORPHY)
* FPBOYD
* FPTRDTOA (transposition)
* FPOSHORLAGPHY -> (see below call tree under FPOSHORLAGPHY)
* SUEFPBIP, FPEZO2H
```

∗ General architecture under DYNFPOS:

```
DYNFPOS ->
* VPOS_PREP
* VPOS -> (see below call tree under VPOS)
* HPOS_DYN -> (call tree not detailed)
* FPHALO
* processor communication routines SLCOMM and (E)SLEXTPOL
* FPOSHOR -> (see below call tree under FPOSHOR)
* FPBOYD
* FPTRDTOA (transposition)
* FPOSHORLAG -> (see below call tree under FPOSHORLAG)
* FPEZO2H
```

6.1.3 General architecture under STEPO_FPOS.

Lower routines appearing in this call-tree may call routines not detailed there.

```
STEPO_FPOS ->
* Vertical interpolations:
  - SCAN2M_VPOS ->
    * VPOS_PREP
    * (E)SPEREE
    * VPOS -> (see below call tree under VPOS)
    * EBIPOS
* Model direct transforms:
  - TRANSDIR_FP
  - FPTSA_DIR
* Spectral calculations (filters):
  - SPOSGF
  - SPOS
* Model inverse transforms:
  - FPTSA_INV
  - TRANSINV_FP
* Horizontal interpolations:
  - DYNFPOS -> (see call tree under DYNFPOS)
  - SUFPILMOD
  - CPVPOSPR
  - FPSATURCAP
  - FPSPECFITG
* Lagged vertical interpolations:
  - ENDVPOS -> (see below call tree under ENDVPOS)
* Output direct transforms:
  - TRANSDIR_FP
  - FPTSA_DIR
* Spectral calculations (filters) in output space:
  - SPOSGF
  - SPOS
* Output inverse transforms:
  - FPTSA_INV
  - TRANSINV_FP
```

6.1.4 Vertical interpolator.

* General architecture under VPOS and POS:

```
VPOS ->
  - GPHPRE
  - CTSTAR
  - POS ->
    * pos.F90/SUPTRPPGFL_POS
    * POS_PREPGFL
    * Several adiabatic GP... and GNH... routines (in part 2.1)
    * Additional adiabatic GP... routines, and PPCVIRT (in part 2.2)
    * CTSTAR
    * FPPS -> PPPMER
    * A series of routines computing intermediate quantities and weights for vertical interpolations.
      - PPLTP
      - PPLTETA
      - PPLTEMP
      - PPLETA -> several adiabatic GP... routines
      - PPINIT
      - PPFLEV
      - PPSTA
    * A series of vertical interpolation routines.
      - PPUV -> PPINTP and PPITPQ
      - PPT -> PPT_OLD, PPSTA and PPINTP
      - PPQ -> PPINTP
      - PPGEOP -> PPSTA, GPGEO, PPITPQ and PPINTP.
      - PP2DINT
      - PPVVEL -> PPINTP
      - PPPMER
      - PPLTP
      - PPTHPW
      - POAERO -> TJQUD, TJCUBI and TJQUAA
      - PPLTEMP
      - FPPS -> PPPMER
      - PPWETPOINT
    * Storage of post-processed fields.
  - PHYMFPOS ->
    * adiabatic GP... and GNH... routines.
    * FPACHMT -> ACSOLW and ACHMT (call tree not detailed)
    * FPCICA -> FPCINCAPE
    * phymfpos.F90/STORE_DATA
    * MTS_PHYS (call tree not detailed)
```

* General architecture under ENDVPOS:

```
ENDVPOS -> ENDPOS ->
  * endpos.F90/SUPTRPPGFL_ENDPOS
  * ENDPOS_PREPGFL
  * Several adiabatic GP... routines (mostly in part 1.3).
  * CTSTAR
  * FPPS -> PPPMER
  * PPLETA -> several adiabatic GP... routines
  * APACHE -> (see below call tree under APACHE)
  * FPACHMT -> ACSOLW and ACHMT (call tree not detailed)
  * FPCICA -> FPCINCAPE
  * PPLTEMP
  * PPWETPOINT
  * PPTHPW
  * PPCVIRT
  * POAERO -> TJQUD, TJCUBI and TJQUAA
  * endpos.F90/STORE_DATA
```

∗ General architecture under APACHE:

```
APACHE ->
* Several adiabatic GP... routines.
* PPINIT
* FPVIEW
* Several adiabatic GP... routines.
* PPFLEV
* PPSTA
* PPUV -> PPINTP and PPITPQ
* PPT -> PPT_OLD, PPSTA and PPINTP
* PPQ -> PPINTP
* PP2DINT
* PPGEOP -> PPSTA, GPGEO, PPITPQ and PPINTP.
```

6.1.5 Horizontal interpolator.

∗ Organigramme of FPOSHOR and FPOSHORPHY:

```
FPOSHOR ->
* FPINTDYN -> FPINT12 and FPINT4.
* FPSAMPL -> FPINT12 and FPINT4.

FPOSHORPHY ->
* FPCLIPHY, FPNILPHY (to compute LLCLI, LLNIL)
* FPINTPHY -> FPINT12, FPINT4, FPINT4X, FPAVG, FPNEAR.
* FPSAMPL -> FPINT12 and FPINT4.
```

∗ Organigramme of FPOSHORLAG and FPOSHORLAGPHY:

```
FPOSHORLAG ->
* FPCORDYN -> FPHOR12.
* FPGEO

FPOSHORLAGPHY ->
* FPCLIPHY -> CVLANISO.
* FPNILPHY -> CVLANISO.
* FPCORPHY -> (call tree not detailed)
* FPGEOPHY
```

6.2 FULL-POS via configuration 903.

∗ General architecture under CNT0 (only FULL-POS features are mentioned):

```
CNT0 ->
* IFS_INIT -> (see above)
* CPREP3 -> (see below call tree under CPREP3)
```

∗ General architecture under CPREP3:

```
CPREP3 ->
* SUGEOMETRY -> (call tree not detailed)
* MODEL_CREATE, FIELDS_CREATE, VARIABLES_CREATE -> (call tree not detailed)
* SUFPOBJ
* SUBFPOS -> (see above)
* IO_SERV_SUIOSCTMPL -> (call tree not detailed)
* GET_ENVIRONMENT_VARIABLE
* SIGMASTER
* CNT3_WAIT -> (call tree not detailed)
* SUOFNAME
* FILEDATE -> (call tree not detailed)
* SUCST
* SUFPINIF -> (call tree similar to SUINIF one)
* SP2GPMCUF
* FPSERVER ->
  - SU4FPOS -> (see above)
  - SUFPDATA -> (see above)
  - ALLFPOS -> (see above call tree under ALLFPOS)
  - FPWRNCF
* GRIBIOFLUSH, if .NOT.LARPEGEF -> (call tree not detailed)
* SIGPOST
* IO_SERV_SYNC -> (call tree not detailed)
* LOGDIS
* GEOMETRY_DELETE, MODEL_DELETE, FIELDS_DELETE, VARIABLES_DELETE
```

6.3 FULL-POS via configuration 904.

∗ General architecture under CNT0 (only FULL-POS features are mentioned):

```
CNT0 ->
* IFS_INIT -> (see above)
* CPREP4 -> (see below call tree under CPREP4)
```

∗ General architecture under CPREP4:

```
CPREP4 ->
* GET_ENVIRONMENT_VARIABLE
* GEOMETRY_SETUP -> SUGEOMETRY, etc..
* MODEL_CREATE, FIELDS_CREATE, VARIABLES_CREATE -> (call tree not detailed)
* SUBFPOS -> (see above)
* READ_FIELDS -> SUINIF, TRANSINVH, etc.
* MODEL_STEP -> STEPO_OOPS, etc.
* SUFPDATA -> (see above)
* FIELDS_FP_CHANGE_RESOL ->
  - FPCHRESOL ->
    * VARFPOS -> (see call-tree above)
    * other routines (for ex. spectral transforms, norm printings)
  - other routines
* FIELDS_ZEROS, FIELDS_COPY, FIELDS_SUB, FIELDS_ADD -> (call tree not detailed)
* SUQLIMSAT -> (call tree not detailed)
* GPNORM_GMV, GPNORM_GFL, GPNORM3, GPNORM2 -> (call tree not detailed)
* GEOMETRY_DELETE, MODEL_DELETE, FIELDS_DELETE, VARIABLES_DELETE
```

6.4 Action and brief description of each routine.

6.4.1 Setup routines and call tree above STEPO_FPOS.

∗ General architecture under CNT0:

- **CNT0**: controls integration job at level 0.
- **SU0YOMA**: 0-level interface routine for set-up: first part.
- **SU0YOMB**: 0-level interface routine for set-up: second part.
- **SUCT0**: routine to initialise level 0 control module.
- **SUCT1**: sets-up **YOMCT1**.
- **SUCST**: routine to initialise universal and astronomical constants.
- **SUGEOMETRY**: routine to initialise model geometry.
- **SUPPVI**: initialise variables used in the vertical interpolator.
- **SUPHY**: header setup routine for the different packages of physics.
- **SUMTS**: setup routine for TOVS radiative transfer code.
- **SUCFU**: initialises the control of cumulated fluxes.
- **SUFPCFU**: initialises cumulated fluxes keys for FULL-POS.
- **SUXFU**: initialises the control of instantaneous fluxes.
- **SUFPXFU**: initialises instantaneous fluxes switches for FULL-POS.
- **SUBFPOS**: general set-up routine for FULL-POS called under **SU0YOMB**.

∗ General architecture under SUBFPOS:

- **SUFPC**: initialises FULL-POS post-processing scientific and technical options; in particular initialise some dimensioning quantities. Reads namelist **NAMFPC**, initialises **YOMFPC**.
- **SUFPCNT**: initialises FULL-POS post-processing control options.
- **UPDTRANS**: to get the LAM/global aspect of a given resolution.
- **SUFPMODELGEO** and **SUFPUSERGEO**: initialise the geometry of a post-processed grid for FULL-POS from a model geometry (**YOMFPUSERGEO** quantities).
- **SUFPGEOMETRY**: to initialize the post-processing horizontal distribution in the departure and the arrival geometries.
- **SUFPSC2** and **SUFPSC2_DEP**: reads namelist **NAMFPSC2** and **NAMFPSC2_DEP**. When **NFPDISTRIB**>=1 only **NAMFPSC2_DEP** is read.
- **SUFPD**: initialise FULL-POS horizontal subdomains. Computes total number of output points. Reads namelist **NAMFPD**, initialises **YOMFPD**.
- **SUFPG**: initialise output geometry (some **YOMFPG** quantities).
- **SUFPEZO**: for LAM models, set-up of variables related to post-processing on E-zone.
- **SUFPWIDE**: initialises the control variables depending on the size of the halo (horizontal interpolations): interpolation buffers quantities, some distributed memory quantities.
- **SLCSET**: set-up for quantities involved in the halo necessary for horizontal interpolations.
- **SUFPWFPBUF**: computes the weights **WSTD04** and **WSTD12** for interpolations.
- **SUHOW1** (**SUEHOW1** for LAM models): initialises the coordinates of the nearest model grid point of the FULL-POS interpolation points.
- **SUHOW2** (**SUEHOW2** for LAM models): computes weights without land-sea mask and surface temperature for horizontal interpolations used in FULL-POS.
- **SUHOX1** (**SUEHOX1** for LAM models): cf. **SUHOW1** in a more optimised way.
- **SUHOWLSM** (**SUEHOWLSM** for LAM models): cf. **SU(E)HOW2** but weights take account of land-sea mask or surface temperature.
- **FPSCAW**: for horizontal interpolator, computes interpolation grid and some indexes to be used with interpolation buffer.
- **FPSCAX**: cf. **FPSCAW**, in a more optimised way.

- **SUFPV**: set-up output vertical geometry variables.
- **SUFPVERT**: set-up output vertical eta coordinate.
- **SUFPIOH**: initialise post-processing I/O parameters (ex: **YOMFPOP**).
- **SUFPIOS**: set-up for using work files on post-processing buffers. To set-up **YOMFPIOS**. Opens work files.
- **SUAFN**: initialises field descriptors. Reads namelist **NAMAFN**, initialises **YOMAFN**.
- **SUAFN1** and **SUAFN2**: initialises field descriptors: part of calculations done before (in SUAFN1) and after (in SUAFN2) the namelist reading.
- **SUAFN3**: prints field descriptors.
- **SUFPF** and **SUFPFILTERS**: initialises the profile of the spectral post-processing filter. **SUFPF** reads namelist **NAMFPF** and initialises **YOMFPF**.
- **CPFPFILTER**: filtering for stretched models:
  - reads or computes dilatation/contraction matrices.
  - computes filtering matrices; stores them in **RFPMAT**.
- **FPFILTER**: fills **RFPFIL** (filtering operator).
- **RDFPFILTER**: filtering for stretched models; reads filtering matrices and fills **RFPMAT** (filtering operator).
- **WRFPFILTER**: filtering for stretched models; write **RFPMAT** on a file.

∗ General architecture under CNT1:

- **CNT1**: controls integration job at level 1.
- **CNT2**: controls integration job at level 2.
- **CNT3**: controls integration job at level 3.
- **CNT4**: controls integration job at level 4.
- **SU1YOM**: 1-level interface routine for set-up.
- **SUINIF**: interface routine for reading the departure files.
- **SUSPEC**: interface for reading the spectral fields of the departure files (ARPEGE files).
- **SUGRIDF**: interface for reading the surface grid point fields of the departure files (ARPEGE files).
- **SUGRIDO**: interface for reading the ocean mixed layer model grid point fields of the departure files.
- **SUGRIDU**: interface for reading the upper air grid point fields of the departure files.
- **SUGRCFU**: reads the cumulated fluxes on ARPEGE files.
- **SUGRXFU**: reads the instantaneous fluxes on ARPEGE files.
- **SUINIF_FP**: receive fields (fullpos IO server).
- **FULLPOS_DRV**: FULL-POS driver.
- **SU4FPOS**: 4-level interface routine for set-up of FULL-POS features. Initialises blocks **YOM4FPOS**.
- **PPREQ**: search for a suitable post-processing namelist file and reads it.
- **SUFPFIELDS**: initialise post-processed fields list for FULL-POS.
- **SUFPPHY**: initialise the requests of post-processing at a given time-step. Reads namelist **NAMFPPHY**.
- **SUFPDYN**: initialise dynamical requests of post-processing at a given time-step. Reads namelists **NAMFPDY...**
- **SUPPDATE**: set-up post-processing date.
- **SUFPDATA**: initialize auxiliary data needed for surface post-processing (climatology, surface-dependent interpolations weights).
- **SUFPRFPBUF_CLIM**: initialise buffer **RFPBUF** of **YOMRFPB**, which contains the output climatology and geometry.
- **SUFPSUW**: computes weights for horizontal interpolations of surface fields.

- **CPCLIMI**: interpolate climatology fields.
- **SUFPCIP**: setup localization of created isolated points (lake or island) in the horizontal interpolation step.
- **DEALLOC_FPFIELDS**: deallocation of the working arrays used exclusively by FULL-POS.
- **SIGPOST**: post events to signal completion of IOs.
- **FP_SERV_SYNC**: to send a syncho signal to the FP server (fullpos IO server).

∗ **General architecture under CPREP3:**

- **CPREP3**: controls the post-processing job at highest level for configuration 903.
- **SUFPOBJ**: set-up FULLPOS objects.
- **SUOFNAME**: set-up an output filename.
- **FILEDATE**: extracts the date of a file.
- **SUFPINIF**: interface routine for reading the departure files (specific version of **SUINIF** for conf 903).
- **SP2GPMCUF**: convert spectral monitoring of coupling update frequencies (MCUF) into gridpoint field.
- **FPSERVER**: runs a post-processing job.
- **GRIBIOFLUSH**: flush GRIB IOs at the end of a step.
- **IO_SERV_SYNC**: send a synchronous signal to the IO server.
- **LOGDIS**: displays the current state of the model execution.
- **FPWRNCF**: to write FULL-POS status.

∗ **General architecture under CPREP4:**

- **CPREP4**: controls the post-processing job at highest level for configuration 904.
- **GEOMETRY_SETUP**: set-up of geometry.
- **READ_FIELDS**: setup of fields.
- **MODEL_STEP** + **STEPO_OOPS**: manages one timestep (OOPS version).
- **FIELDS_FP_CHANGE_RESOL** + **FPCHRESOL**: to change the resolution of a FIELDS object.
- **SUQLIMSAT**: applies a correction to specific humidity.
- **VARFPOS**: performs a complete step of post-processing in configuration 904.

6.4.2 ALLFPOS, GRIDFPOS, DYNFPOS.

- **ALLFPOS**: performs a complete step of post-processing.
- **GRIDFPOS**: interface routine managing post-processing of grid point fields. Performs horizontal post-processing on physical fields and fluxes and writes post-processed fields on a file.
- **DYNFPOS**: interface routine managing vertical then horizontal interpolations for post-processing on dynamical 3D and 2D fields, calls several sequences of **STEPO_FPOS**.
- **SUFPOFNAME**: set-up variables needed before opening files for post-processing purpose.
- **SUFPTR2**: computes the number of fields **NFPVT0** and the associated pointers for the auxiliary grid-point quantities.
- **FPMODCFU** (resp. **FPMODXFU**, **FPMODPREC**): preliminary modification of cumulated fluxes (resp. instantaneous fluxes, precipitations) for post-processing: prepares input fields before horizontal interpolations.
- **MAXGPFV**: computes maximum value of a grid-point field.
- **SUFPOROG**: set-up output orography.
- **FP2SX1** and **FP2SX2**: FULL-POS interfaces for SURFEX.
- **CPVPOSPR**: computes field pointers for vertical post-processing.
- **SUVFPOSL**: initialises **YOMFP4L** which contains the working variables and arrays to write out the dynamical post-processed fields.
- **ESPFP**: LAM model routine writing vertical post-processed data on a FA file, equivalent of **WRSFP**.

- **WRSFP**: writes out the vertically post-processed dynamical fields to ARPEGE/ALADIN file. Spectral fields are written out as spectral coefficients (if spectral output is wanted), grid-point fields are written out as grid-point fields. The horizontal grid is the model one.
- **WRHFP**: writes out the horizontally post-processed fields, sorts out the output points for each domain, then writes records on files. Only grid-point data are written.
- **WRMLFP**: GRIB routine writing data on model layers.
- **WRPLFP**: GRIB routine writing data on pressure layers, potential vorticity layers, potential temperature layers.
- **SCAN2M_HPOS**: post-processing in grid-point space: control routine for horizontal interpolations.
- **HPOS**: interface routine managing non lagged part of horizontal interpolations. Fills buffer for quantities to be interpolated (array **PFPBUF1**).
- **HPOS_DYN**: fills part of array **PFPBUF1**.
- **HPOS_CFU** and **HPOS_XFU**: fills part of array **PFPBUF1** (CFU, XFU).
- **FPTENSOR**: to compute the fields parity array and scale fields with map factor, depending on the order of derivative of the fields.
- **FPHALO**: to fill the distributed halo from core data, prior to horizontal interpolations.
- **SLCOMM** does processor communication to fill the halo required for horizontal interpolations.
- **SLEXTPOL** adds data of extra-polar latitudes in the halo when necessary.
- **FPTRATOD**: transposition routine between arrival geometry DM-distribution and departure geometry DM-distribution.
- **FPTRDTOA**: does the inverse operation of routine **FPTRATOD**.
- **SUEFPBIP**: to set up the list of fields to be biperiodicised.
- **FPEZO2H**: post-processing of extension zone in LAM models.

6.4.3 General architecture under STEPO_FPOS.

- **STEPO_FPOS**: controls FULL-POS job at lowest level.
- **SCAN2M_VPOS**: post-processing in grid-point space: control routine for unlagged vertical interpolations.
- **VPOS_PREP**: computes input quantities for **VPOS**.
- **EBIPOS**: bi-periodicization of vertically post-processed fields in LAM models.
- **TRANSDIR_FP**: direct spectral transforms for post-processed quantities.
- **TRANSINV_FP**: inverse spectral transforms for post-processed quantities.
- **FPTSA_DIR**: memory transfer in spectral space between arrays coming from spectral transforms and arrays entering spectral calculations.
- **FPTSA_INV**: does the inverse operation of routine **FPTSA_DIR**.
- **SPOS**: filters some spectral quantities in spectral space; in particular does filtering requiring array **RFPMAT**.
- **SPOSGF**: filters some spectral quantities in spectral space by a Gaussian filter.
- **SUFPILMOD**: set-up the list of primitive variables for the post-processing.
- **FPSPECFITG**: spectral fit of some model fields, for ECMWF congigurations.

6.4.4 General architecture under VPOS and ENDVPOS (vertical interpolator).

- **VPOS**: interface routine managing all vertical interpolations which can be done before horizontal interpolations. In particular **VPOS** does one call to **POS**.
- **POS**: inner interface for vertical post-processing.
- **POS_PREPGFL**: GFL preliminary calculations for **POS**.
- **GPHPRE**: computes half-level and full-level pressures and intermediate quantities on η-levels.

- **CTSTAR**: computes the standard surface temperature and the surface temperature to be used for extrapolations of temperature and geopotential height.
- **PPCVIRT**: transforms a temperature variable into its virtual counterpart.
- **FPPS**: computes the pressures of a set of surfaces, given their geopotential heights.
- **PPPMER**: computes mean sea level pressure.
- **PPLTP**: computes the pressures of post-processing on potential vorticity levels.
- **PPLTETA**: computes the pressures of post-processing on Θ-levels.
- **PPLTEMP**: computes the geopotential for a given temperature in a vertical profile.
- **PPLETA**: computes the pressures of post-processing on η-levels.
- **PPINIT**: computes intermediate quantities connected with full levels and half levels pressures, prior to vertical interpolations on pressure or height levels.
- **PPFLEV**: finds full or half levels under specified pressures.
- **PPSTA**: integrates standard atmosphere for geopotential height.
- **PPUV**: vertical interpolations on pressure levels for wind components.
- **PPT**: vertical interpolations on pressure levels for temperature.
- **PPT_OLD**: old version of routine **PPT** used at ECMWF.
- **PPQ**: vertical interpolations on pressure levels for humidity and most of post-processable GFL variables.
- **PPGEOP**: vertical interpolations on pressure levels for geopotential height.
- **PP2DINT**: vertical linear interpolation (on pressure levels or height levels).
- **PPVVEL**: vertical interpolations on pressure levels or height levels for variables linked to vertical velocity.
- **PPTHPW**: computation of the moist irreversible adiabatic potential temperature, vertical interpolation on pressure or height levels.
- **POAERO**: post-processing of jet and tropopause heights.
- **TJQUD**, **TJCUBI** and **TJQUAA**: post-processing of jet and ICAO tropopause.
- **PPLTEMP**: computes the pressure value for a given temperature in a vertical profile.
- **PPWETPOINT**: computation of the wetpoint temperature.
- **PPINTP**: vertical linear interpolation on pressure levels.
- **PPITPQ**: vertical quadratic 3 points interpolation.
- **PHYMFPOS**: physico-dynamic post-processing interface for METEO-FRANCE physics.
- **FPACHMT**: FULL-POS interface to ACHMT, to compute pressure, humidity, temperature and wind.
- **ACSOLW**: computation of some surface coefficients and variables.
- **ACHMT**: calculation of some surface characteristics; interpolation of some variables (U, V, T, q) at some chosen heights.
- **FPCICA**: interface to **FPCINCAPE** (see below).
- **FPCINCAPE**: routine computing CAPE (convective available potential energy) and CIN (convective inhibition).
- **MTS_PHYS**: physics for TOVS radiative transfer code.
- **ENDVPOS**: interface routine managing all post-processing computations which can be done only after horizontal interpolations.
- **ENDPOS**: finish post-processing calculations for output height levels or η-levels.
- **ENDPOS_PREPGFL**: GFL preliminary calculations for **ENDPOS**.
- **APACHE**: interface routine for some vertical interpolations, called for example if post-processing on height or η-levels.
- **FPVIEW**: compute weights for vertical interpolations combining 2 profiles.

6.4.5 General architecture under FPOSHOR and FPOSHORPHY (horizontal interpolator).

- **FPOSHOR** and **FPOSHORPHY**: interface routine managing lagged part of horizontal interpolations.
- **FPOSHORLAG** and **FPOSHORLAGPHY**: interface routine managing lagged part of horizontal interpolations: second part.
- **FPCLIPHY**: writes the output climatology in the post-processing buffers and initialises a logical climatological mask.
- **FPNILPHY**: computes the physical fields which cannot be directly interpolated but need auxiliary computation or need to be set-up by a straight value.
- **FPINTDYN**: interface for horizontal interpolations, for fields coming from the dynamics.
- **FPINTPHY**: interface for horizontal interpolations, for fields coming from the physics.
- **FPSAMPL**: horizontal sampling, when interpolation points are on the input grid.
- **FPINT4**: 4 points horizontal interpolations; also used for some multi-linear interpolations.
- **FPINT4X**: 4 points horizontal interpolations, with possibility to take account of missing values (less than 4 points may be used in this case).
- **FPINT12**: 12 points horizontal interpolations; also used for some multi-linear interpolations.
- **FPAVG**: FULL-POS interpolation based on average over the FULL-POS halo.
- **FPNEAR**: interpolate field for FULL-POS using the nearest point in the FULL-POS halo.
- **FPCORDYN** (resp. **FPCORPHY**): corrects dynamical (resp. physical) fields after the horizontal interpolations which may create overshoots, or to recover the requested fields.
- **FPHOR12**: corrections after 12-points horizontal interpolations, for example to prevent overshoots.
- **FPGEO** and **FPGEOPHY**: transforms the post-processed fields into the output geometry.
- **CVLANISO**: makes a transformation for surface orography in order to have the horizontal gradient (departure data are standard deviation, anisotropy coefficient and direction of the principal axis).

6.4.6 Printings and norms.

- **FPSPNORMS**: compute and print norms of spectral post-processed fields.
- **FPGPNORM**: compute and print norms of grid-point post-processed fields.
- **GPNORM_GMV**, **GPNORM_GFL**, **GPNORM3**, **GPNORM2**: compute and print grid-point norms, respectively for GFL, GMV, 3D and 2D fields.

6.4.7 Actions on objects.

- **MODEL_CREATE**, **FIELDS_CREATE**, **VARIABLES_CREATE**: creation of objects, for example for object MODEL, FIELDS.
- **GEOMETRY_DELETE**, **FIELDS_DELETE**, **MODEL_DELETE**, **VARIABLES_DELETE**: destruction of objects, for example for object GEOMETRY, MODEL, FIELDS.
- **FIELDS_ZEROS**, **FIELDS_COPY**, **FIELDS_SUB**, **FIELDS_ADD**: operations on object FIELDS, for exemple set to 0, copy, addition, substraction.

7 Sequences of calls of post-processing.

7.1 Vertical coordinates of post-processing.

Post-processing can be done on pressure levels, height levels, potential vorticity levels, potential temperature levels, temperature levels, flight levels or η-levels. For post-processing on pressure levels a vertical coordinate linked to pressure (pressure or logarithm of pressure, according to the post-processed variable) is used to compute weights for vertical interpolations. For post-processing on height levels a vertical coordinate linked to geopotential height is used to compute weights for vertical interpolations. For post-processing on potential vorticity levels, potential temperature levels or η-levels an intermediate state through a pressure-type coordinate is needed to compute vertical weights.

7.2 Sequences of calls: general considerations.

Post-processing for each coordinate is done by different sequences under routine **STEPO_FPOS**. Vertical interpolations and horizontal interpolations are done by different sequences under **STEPO_FPOS**. **STEPO_FPOS** has its own sequence **CLCONF** (now completely different from the model one), defined by nine letters (or zeros) [L1][L2][L3][L4][L5][L6][L7][L8][L9]

- L1 controls vertical post-processing.
- L2 controls model direct spectral transforms.
- L3 controls spectral filters in model space.
- L4 controls model inverse spectral transforms.
- L5 controls the grid-point horizontal interpolations for post-processing.
- L6 controls lagged vertical post-processing.
- L7 controls output direct spectral transforms.
- L8 controls spectral filters in output spectral space.
- L9 controls output inverse spectral transforms.

No extensive inventory will be given in this documentation for letters used in the different elements of **CLCONF**, but we can say for example that:

- '0' means that no action is done.
- 'P' or 'F' is generally used for elements controlling spectral transforms when they must be done.

For more details reader can refer to technical document (IDFPOS2).

7.3 Different choices for horizontal output.

* Groups of post-processed fields and abbreviations:

- DYN3D: 3D dynamical fields.
- DYN2D: 2D dynamical fields.
- DYN2DGP: 2D dynamical fields always written in grid-point.
- MSLP: mean sea level pressure.
- PHIS: surface orography.
- PS: surface pressure.
- PHYSOL: surface physical fields.
- CFU: cumulated fluxes.
- XFU: instantaneous fluxes.

Each group may have a specific treatment; CFU and XFU are treated like PHYSOL.

∗ **Choice of horizontal geometry and representation (spectral vs grid-point) for output:** Output data can be written on the following output grids and with the following representations:

- Spectral representation, Gaussian grid.
- Grid-point representation, Gaussian grid.
- Spectral representation, LELAM grid.
- Grid-point representation, LELAM grid.
- Grid-point representation, LALON grid.

It is done according to variables **CFPFMT** in **NAMFPC** and **NFPOS** in **NAMCT0**.

- (NFPOS, CFPFMT) = (2,'MODEL'): output horizontal geometry is identical to model one, and output fields have the same representation as model ones.
- (NFPOS, CFPFMT) = (1,'GAUSS'): for global model input geometry only. Output horizontal geometry is a Gaussian grid one, which can be different (or not) from the model grid. Output fields are written in grid-point representation.
- (NFPOS, CFPFMT) = (2,'GAUSS'): for global model input geometry only. Output horizontal geometry is a Gaussian grid one, which can be different (or not) from the model grid. Output fields have the same representation as model ones.
- (NFPOS, CFPFMT) = (1,'LELAM'): for both global model and LAM input geometry. Output horizontal geometry is a LELAM grid one, which can be different (or not) from the model grid. Output fields are written in grid-point representation.
- (NFPOS, CFPFMT) = (2,'LELAM'): for both global model and LAM input geometry. Output horizontal geometry is a LELAM grid one, which can be different (or not) from the model grid. Output fields have the same representation as model ones.
- (NFPOS, CFPFMT) = (1,'LALON'): for both global model and LAM input geometry. Output horizontal geometry is a LALON grid one. Output fields are written in grid-point representation.

8 Some distributed memory features.

8.1 Calculations packets.

∗ **Message passing division into processors:** The total number of processors is **NPROC**. There are two levels of parallelisation. Distribution on these two levels is different in the different spaces of calculations.

∗ **Grid-point computations:** **NPROC**=**NPRGPNS**∗**NPRGPEW**. The total number of processors involved in the A-level parallelisation is **NPRGPNS**. The total number of processors involved in the B-level parallelisation is **NPRGPEW**.

One 2D model field has **NGPTOTG** points divided into **NPRGPNS***NPRGPEW sets of **NGPTOT** points treated by each processor. **NGPTOT** may be processor-dependent (with very small variations): the maximum value of **NGPTOT** is **NGPTOTMX**.

For one given processor, the **NGPTOT** points are divided into packets of length **NPROMA** (the useful number of values in each packet is lower or equal than **NPROMA**). **NPROMA** is identical for all processors. There are **NGPBLKS** blocks of **NPROMA** packets:

$$\mathbf{NGPBLKS} = int[(\mathbf{NGPTOT} + \mathbf{NPROMA} - 1)/\mathbf{NPROMA}]$$

A **NPROMA**-packet does not always contain a set of complete latitudes. This subdivision into **NPROMA**-packet only concerns not lagged computations.

There are **NFPRGPG** interpolation points, and there are two ways to distribute them among processors:

- A "departure horizontal geometry" distribution, used in horizontal interpolations.
- A "arrival horizontal geometry" distribution, used in calculations done after horizontal interpolations.

In the departure (resp. arrival) horizontal geometry distribution, each processor has **NFPRGPL_DEP** (resp. **NFPRGPL**) points to process. **NFPRGPL_DEP** may be significantly processor dependent. **NFPRGPL** is slightly processor dependent (with very small variations). The maximum value of **NFPRGPL_DEP** (resp. **NFPRGPL**) is **NFPRGPLX_DEP** (resp. **NFPRGPLX**). For one given processor, the **NFPRGPL_DEP** (resp. **NFPRGPL**) points are divided into **NFPBLOCS_DEP** (resp. **NFPBLOCS**) packets of length **NFPROMA_DEP** (resp. **NFPROMA**). **NFPROMA_DEP** and **NFPROMA** are identical for all processors.

If **NFPDISTRIB**=0, the arrival horizontal geometry distribution is set equal to the departure horizontal geometry distribution. For example **NFPRGPL**=**NFPRGPL_DEP**, **NFPROMA**=**NFPROMA_DEP**. Such configuration is always used when the arrival horizontal geometry is identical to the departure horizontal geometry.

If **NFPDISTRIB**=1 or 2, the arrival horizontal geometry distribution is set different from the departure horizontal geometry distribution. The variation of **NFPRGPL** between each processor is minimized to ensure a good load balance. If there is one domain (**NFPDOM**=1), data of the **NFPRGPG**-arrays are dispatched first in the processor number 1, then in the processor number 2, etc... For example if **NFPRGPG**=64 and **NPROC**=2, **NFPRGPL** will be equal to 32 for both processors, **NFPRGPG**-arrays indices number 1 to 32 (resp. 33 to 64) will be treated by processor 1 (resp. processor 2). If there are several domains, such distribution is done first for the first domain, then for the second domain, etc.., so each processor has a subset of points of all domains. **NFPDISTRIB**=1 or 2 is recommended if the arrival horizontal geometry is significantly different from the departure horizontal geometry (cases where the load balance can be not good in the departure horizontal geometry distribution); **NFPDISTRIB**>=1 is compulsory when the arrival horizontal geometry is different from the departure one and when some SURFEX surface fields are post-processed.

The way of dispatching the points among the processors in the departure horizontal geometry distribution is the same for **NFPDISTRIB**=1 or 2 and **NFPDISTRIB**=0 (see part 8.2).

NFPDISTRIB=2 sets gridpoint distributions according to the spectral transforms package. **NFPDISTRIB**=2 is needed for **NFPOS**=2 and **CFPFMT**/='MODEL'.

More details will be given later for the data transmission for horizontal interpolations. For global models, all these variables take account of the reduced Gaussian grid. All the vertical levels and the variables corresponding to a same grid-point are treated by the same processor. There are necessary transpositions (reorganisation of data) between grid point computations and Fourier transforms because Fourier transforms need complete latitudes.

∗ **Fourier transforms:** **NPROC=NPRTRNS∗NPRTRV**. The total number of processors involved in the A-level parallelisation is **NPRTRNS**. The total number of processors involved in the B-level parallelisation is **NPRTRV**. Fourier transforms are done latitude by latitude for each **NPROMA**-packet. A processor treats a subset of latitudes, one latitude at the time. A processor can treat only a subset of fields to be post-processed (if **NPRTRV**>1). Data reorganisation and transpositions are necessary in the Fourier space between the zonal wave structure necessary for Legendre transforms and the latitudinal structure necessary for Fourier transforms.

The B-level parallelization is a circular one in spectral space and spectral transforms (Fourier and Legendre); the list of fields to be interpolated is considered as a list of 2D fields and these 2D fields are spread among the **NPRTRV** processors (fields numbers 1 to **NPRTRV** on processors 1 to **NPRTRV**, then fields numbers **NPRTRV**+1 to 2∗**NPRTRV** on processors 1 to **NPRTRV**, etc). When it is necessary some fields are kept together (for example U and V on a same layer) so the distribution is not completely circular.

∗ **Legendre transforms (North-South Fourier for LAM models):** **NPROC=NPRTRW∗NPRTRV**. The total number of processors involved in the A-level parallelisation is **NPRTRW**. The total number of processors involved in the B-level parallelisation is **NPRTRV**. Legendre transforms are done zonal wave number by zonal wave number. A processor treats a subset of zonal wave numbers, one zonal wave number at the time.

∗ **Filtering in spectral space:** **NPROC=NPRTRW∗NPRTRV**. The total number of processors involved in the A-level parallelisation is **NPRTRW**. The total number of processors involved in the B-level parallelisation is **NPRTRV**. Filtering in spectral space is done zonal wave number by zonal wave number. A processor treats a subset of zonal wave numbers, all this work being done in one call of **SPOS** and **SPOSGF**.

8.2 Transmission of data necessary for FULL-POS horizontal interpolations (for example from HPOS_DYN to FPOSHOR): interpolation buffers.

Description is done for the couple of routines (**HPOS_DYN**, **FPOSHOR**) involved in post-processing of "dynamics" fields. First one associates to each interpolation point, which is generally not a model grid-point, an "associated" model grid-point which currently matches to the following rule:

- the "associated" model grid-point is on the longitude immediately west to the interpolation point.
- the "associated" model grid-point is on the latitude immediately north to the interpolation point if the interpolation point is not between the Gaussian North pole and the first Gaussian latitude; in the contrary one takes the first Gaussian latitude.

This associated model grid-point is always between the latitudes 1 and **NDGLG** (it is never a pole or an extra-polar latitude point). In the departure horizontal geometry distribution, the processor which treats the FULL-POS interpolation point is the processor which treats this associated model grid-point.

Interpolations use data of points which are not necessary on the same latitude and longitude as the interpolation point. Thus interpolation routines need to have access to a limited number of surrounding latitudes and longitudes which are not necessary treated by the current processor. Interpolation points are collected so that a call to **FPOSHOR** for horizontal interpolations treats a set of points, the "associated" model grid-points set of which is a **NGPTOT** set which is treated by one processor. The number of surrounding latitudes and longitudes rows necessary for interpolations but which do not belong to the current processor is precomputed in the subroutine **SUFPWIDE** (variable **YFPSTRUCT%NSLWIDE**). This is a sort of "halo" belonging to some other processors. Due to the "halo" there is still need to split calculations into not lagged ones (**HPOS_DYN**) and lagged ones (**FPOSHOR**). Quantities to be interpolated are computed in the non-lagged part and interpolations are performed in the lagged part. In **HPOS_DYN**, only data of the current processor (without any extra-longitudinal data nor polar and extra-polar data) are computed. For all the **NPROMA**-packages treated by the current processor these data are stored in the arrays **PFPBUF1** in **HPOS_DYN**. Then a memory transfer is immediately done after each call to **HPOS_DYN** in a bigger interpolation buffer where some place is let for extra-longitudes and the halo. Then some communication routines are called to constitute the halo. **SLCOMM** does processor communication to constitute the halo (receives and sends data from some other processors). **SLEXTPOL** adds data of extra-polar latitudes in the halo when necessary. When all the dataset is ready for interpolation, the lagged part **FPOSHOR** is called which do horizontal interpolations for all the **NFPRGPL_DEP** data to be interpolated for the current processor.

8.3 Case LEQ_REGIONS=T.

This case is relevant only when **NPRGPEW**>1 (B-level parallelisation at least in the grid-point calculations), and use a global model with a reduced Gaussian grid. This is an optimised version of the **LEQ_REGIONS**=F case which is well designed for reduced Gaussian grid and it improves the load balance in this case. A comprehensive description can be found in (Mozdzynski, 2006). To sum-up, we can say that:

- the A-level grid-point distribution splits the Earth into **N_REGIONS_NS** bands. **N_REGIONS_NS** can be slightly different from **NPRGPNS**.
- for each band *jroca*, the B-level grid-point distribution splits the band into **N_REGIONS**(*jroca*) zones: the minimum value of **N_REGIONS** is at the poles of the computational sphere (equal to 1 in the examples provided by Mozdzynski); the maximum value of **N_REGIONS** is at the equator of the computational sphere and this maximum is equal to **N_REGIONS_EW**. The meridian variations of **N_REGIONS** are highly correlated to those of **NLOENG**.
- In the examples provided by Mozdzynski, **NPRGPNS**=**NPRGPEW**=**NPRTRW**=**NPRTRV** and we notice that **N_REGIONS_NS** is slightly below **NPRGPNS**, and that **N_REGIONS_EW** is slightly below 2∗**NPRGPEW**.

When **LEQ_REGIONS**=F, variables **N_REGIONS_NS**, **N_REGIONS** and **N_REGIONS_EW** are still used but in this case:

- **N_REGIONS_NS**=**NPRGPNS**.
- **N_REGIONS**=**NPRGPEW** everywhere.
- **N_REGIONS_EW**=**NPRGPEW**.

9 Module and namelist variables to be known.

These modules are auto-documented so description of each variable is provided in the code source. We can recall here the most important variables to know for each module:

9.1 Module FULLPOS.

This module defines two main types: **TFPOS** and **TFPDATA**, with attributes which are also derivated types. In **CNT0**, **TFPOS**-typed variable **YLFPOS** is declared. In **CNT4**, **CPREP3** and some other routines, **TFPDATA**-typed variable **YLFPDATA** is declared.

Sub-types of **TFPDATA**:

- Type **TFPSUW**: used in **YOMWFPB**. Contains some indexes and weights for horizontal interpolations.
- Type **FPOSBUF**: used in **TYPE_FPOSBUF**.

Sub-types of **TFPOS**:

- Type **TFPCNT**: used in **YOMFPCNT** (controler).
- Type **TFPGEOMETRY**: used in **YOMFPGEOMETRY** (horizontal geometry).
- Type **TVAB**: used in **YOMVERT** (vertical geometry).
- Type **TFPFILTERS**: used in **YOMFPFILTERS** (spectral filters).
- Type **SL_STRUCT**: used in **EINT_MOD** (externalisable part of interpolator: horizontal halo management).
- Type **TFPWSTD**: used in **YOMWFPB**. Contains some indexes and weights for horizontal interpolations.
- Type **TFPIOH**: used in **YOMFPOP**. Contains post-processing file-handling variables.
- Type **TFPAFN**: used in **YOMAFN**. Contains fields descriptors. Some of these variables are in namelist **NAMAFN**.
- Type **TNAMFPSCI**: used in **YOMFPC** (scientific parameters).
- Type **TNAMFPINT**: used in **YOMFPC** (horizontal interpolations).
- Types **TNAMFPL**, **TNAMFPOBJ**: used in **YOMFPC**.

9.2 Other modules which define types.

- Type **TRQFP**: used in **YOMFP4L**. Contains information relative to lagged variables needed to write out post-processed fields.
- Type **TFPUSERGEO**: used in **TYPE_FPUSERGEO** (geometry definition).
- Type **TFPGEO**: used in **YOMFPGEO** (target grid or interpolation grid parameters).
- Type **TFPGIND**: used in **YOMFPGIND**. Contains variables defining some indexes.
- Type **TNAMFPIOS**: used in **YOMFPIOS**. Contains control variables for FULL-POS file read/write. Some of these variables are in namelist **NAMFPIOS**.
- Type **TFAOPH**: used in **TYPE_FAOPH**.
- Type **TFPOFN**: used in **TYPE_FPOFN**.
- Type **FULLPOS_TYPE**: used in **FULLPOS_MIX**.
- Type **FPDSPHY**: used in **TYPE_FPDSPHYS**.
- Type **TFPFIELDS**: used in **TYPE_FPFIELDS**.
- Type **TYPE_FPRQDYN**: used in **TYPE_FPRQDYNS**. Related to DYN3D and DYN2D post-processable fields.
- Type **TYPE_FPRQPHY**: used in **TYPE_FPRQPHYS**. Related to PHYSOL, CFU and XFU post-processable fields.

9.3 Other modules.

- **PARFPOS**: contains basic dimensions for FULL-POS post-processing.
- **YOM4FPOS**: contains variables relative to FULL-POS working arrays (level 4).
- **EXTFPSELECT_MOD**: contains encapsulated routines doing memory transfers from/to buffers.
- **FPGPNORM_MOD**: contains encapsulated routines doing grid-point norms calculations.
- **YOMFPD**: contains variables concerning the boundaries and the horizontal dimensions of each output subdomain. Some of these variables are in namelist **NAMFPD**.
- **YOMFPF**: contains variables for FULL-POS filtering. Some of these variables are in namelist **NAMFPF**.
- **YOMFPG**: contains variables defining the characteristics of the (transformed) output geometry. Some of these variables are in namelist **NAMFPG**.
- **YOMFPOBJ**: contains variables defining fullpos objects. Some of these variables are in namelist **NAMFPOBJ**.
- **YOMFP_SERV_DINF** and **YOMFP_SERV**: for fullpos IO server.
- **EINT_MOD**: externalisable part of interpolators, and halo management (including distributed memory aspects to communicate the halo among the different processors). In particular, variable **YFPSTRUCT** is used in FULL-POS horizontal interpolator.
- **YOMCAPE**: contains variables to control CAPE computation in FULL-POS. Some of these variables are in namelist **NAMCAPE**.
- **YOMARG** (0-level control, former command line) and **YOMCT0** (0-level control):
  - NCONF (in **NAMARG**).
  - NFPOS.
  - LECPOS.
  - CFPNCF, CFDIRLST, CNPPATH.
  - NFRPOS, NPOSTS.

  Some of these variables are in namelist **NAMCT0**.
- **YOMCT1** (1-level control): in particular N1POS. Some of these variables are in namelist **NAMCT1**.
- **YOMDIM**: contains dimensioning variables. In particular NDLON, LOPTPROMA, NPROMA, NGPBLKS, NDGSAFPH, NDGENFPH. Some of these variables are in namelist **NAMDIM**.
- **YOMFPC**: contains scientific and technical variables for post-processing. Some of these variables are in namelist **NAMFPC**.
- **YOMIOS**: contains packing factors and pathnames for some files. Some of these variables are in namelist **NAMIOS**.
- **YOMMP0** and **YOMMP**: distributed memory environment, see documentation (IDDM) for more details.
- **YOMOPH0**: contains file-handling variables. Some of these variables are in namelist **NAMOPH**.
- **YOMPPVI**: contains variables for vertical interpolator used for example in FULL-POS. Some of these variables are in namelist **NAMPPVI**.
- **YOMSTA**: contains constants for standard atmosphere. In particular HEXTRAP.
- Modules for geometry:
  - **TYPE_GEOMETRY** (geometry derived types).
  - **YOMCSGEOM** (computational space grid-point geometry).
  - **YOMGSGEOM** (geographic space grid-point geometry).
  - **YOMOROG** (orography).
  - **YOMGEM** (other variables for horizontal geometry): all variables. Some of these variables are in namelist **NAMGEM**.
  - **YOMVERT** (vertical geometry).
  - **YOMVV1** (namelist variables for vertical geometry). Variables are in namelist **NAMVV1**.
  - **YEMGEO** (LAM model geometry): all variables. Some of these variables are in namelist **NEMGEO**.
- Some modules in directory "oops", in particuliar:
  - **FIELDS_FP_MOD**: fullpos driver for OOPS.

Additional namelists elements: these namelists are **NAMFPPHY** (for physical fields and fluxes), **NAMFPDY2** (for 2D dynamical fields), **NAMFPDYP** (fields post-processed on pressure levels), **NAMFPDYH** (fields post-processed on height levels), **NAMFPDYV** (fields post-processed on potential vorticity levels), **NAMFPDYI** (fields post-processed on isentropic levels), **NAMFPDYT** (fields post-processed on temperature levels), **NAMFPDYF** (fields post-processed on flight levels), and **NAMFPDYS** (fields post-processed on hybrid levels). These namelists can be used to make an accurate list of post-processed fields.

10 References.

- (TDECDYN) 2017: IFS technical documentation (CY43R3). Part III: dynamics and numerical procedures. Available at "https://software.ecmwf.int/wiki/display/IFS/Official+IFS+Documentation".
- (TDECTEC) 2017: IFS technical documentation (CY43R3). Part VI: technical and computational procedures. Available at "https://software.ecmwf.int/wiki/display/IFS/Official+IFS+Documentation".
- Andersson, E., and Ph. Courtier, 1992: Post-processing changes for IFS cycle 9. (Research Department memorandum, internal note).
- (IDFPOS2) El Khatib R., 2012: FULLPOS-2: design, specifications study and development (version 6). Internal note, 15pp.
- Estrade J.-F., 1997: Développement ARPEGE/ALADIN. Mémoire distribuée/ mémoire partagée (internal note in French).
- Janoušek, M., 2001: New port of the new geographical routines to ALADIN/ARPEGE (internal note).
- (IDEQR) Mozdzynski, G., 2006: A new partitioning approach for IFS. Internal note, 6pp.
- (NTA30) Rochas, M., et Ph. Courtier, 1992: La méthode spectrale en météorologie. Note de travail ARPEGE numéro **30**, 58pp.
- Undén, P., 1995: IFS post-processing changes proposed by METEO-FRANCE and an improved formulation (internal note).
- (IDBAS) Yessad, K., 2018: Basics about ARPEGE/IFS, ALADIN and AROME in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDEUL) Yessad, K., 2018: Integration of the model equations, and Eulerian dynamics, in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDTS) Yessad, K., 2018: Spectral transforms in the cycle 46t1 of ARPEGE/IFS (internal note).
- (IDRD) Yessad, K., 2018: Sphere to sphere transforms in spectral space in the cycle 46t1 of ARPEGE/IFS: configuration 911. (internal note).
- (IDDM) Yessad, K., 2018: Distributed memory features in the cycle 46t1 of ARPEGE/IFS (internal note).