# NEW PRESENTATION FOR ROUTINE POS.

YESSAD Karim.

March 24, 2009

Version 2.

# 1 Introduction and purpose.

Routine **POS** manages the vertical interpolations of the post-processing (now in FULL-POS). This routine has been written in the end of the 80's and has been used in the old post-processing software which existed before the implementation of FULL-POS. Its design is still strongly linked to the type of dynamics which existed in the early stages of ARPEGE/IFS, i.e. a thin layer hydrostatic formulation of the equations; this design was also well adapted for a limited amount of post-processable variables, but has reached some limits with the number of post-processable variables we have currently.
This is not too misleading for off-line uses of FULL-POS: in this case the forecast writes an historical file, and the external call of FULL-POS is usually done in a hydrostatic frame. Problems start to occur when using in-line FULL-POS in a model with NH equations or deep-layer equations. There is an increased use of in-line FULL-POS configurations, not only at ECMWF (where this is necessary because ECMWF has no specific historic files: outputs are always written via FULL-POS), but also now at METEO-FRANCE. The current METEO-FRANCE operational suite (cy35t1_op1) uses in-line configurations of FULL-POS, NH AROME included. For this reason having a properly coded **POS** becomes more and more stringent.

The purpose of this paper is to give a list of problems encountered with the current design of **POS**, and some proposals of rewritings to solve them. Rewriting will be extended too some other routines involved in the vertical interpolator of FULL-POS.

# 2 Shortcomings with the current design of POS.

The main shortcomings can be listed as follows:

- Dynamics which is in parts 1.2 and 1.3 is too much oriented "thin layer hydrostatic dynamics", and it is difficult to introduce NH dynamics and/or deep-layer formulations properly. About NH dynamics, what has been already done works for a subset of options, and is not very clean: some diagnostics are not consistent with the NH dynamics (geopotential height and its gradient for example). Deep-layer formulations have not been properly introduced in these pieces of code. I have also noticed that VFE vertical discretisation is not always easy to introduce properly, even if some efforts has been done during the last cycles.
- Parts doing dynamics (call of GP.. and GNH.. routines) and parts doing vertical interpolations, are not enough separated. There are still too many GP.. and GNH.. routines called under some interpolation routines, and these calls may not see the NH or deep-layer formulations effects. **APACHE** is a serious issue about this point.

  That can be a serious problem for some deep-layer NH formulations which require a variable change for the vertical coordinate.

  That can forbid implementation of alternate formulations of NH dynamics.

  Another consequence is that we do the same calculations several times, via successive calls to GP.. routines which exactly do the same thing. For example the calls to **GPRCP** done under **FPPS** (called three times in **POS**) compute exactly the same quantities as the call to **GPRCP** done in part 1.2 of **POS**. The same redundancy occurs for some of the GP... routines called under **APACHE**.
- Calculations done in parts 1.2 and 1.3 are switched on according to very tricky tests (using CDCONF and a large number of LL_. keys), maybe not always safe (for example test "LL_VD.OR.LL_VW.AND.(.NOT.LFPART2)" in part 1.2.6 is suspicious, I think this is rather "(LL_VD.OR.LL_VW).AND.(.NOT.LFPART2)"). The consequence is that there is a serious risk to use uninitialised intermediate quantities and to introduce bugs in these tests. New developments (NH alternate formulations, new post-processed variables) are difficult to implement in this frame.

And we can add the following remarks:

- GFL variables are treated individually: that generates long sequences of code (more than 600 lines only for the GFL treatment), and **POS** must be updated each time a new post-processable GFL is introduced. Introducing new GFL in **POS** and more generally in the vertical interpolator of FULL-POS becomes difficult, and I also notice that ill-formulated introduction of GFL variables in routine **APACHE** leads to have 5 calls to **APACHE** in **POS** (and also in **ENDPOS**), where the number of calls should be normally only one.

  Individual treatment of GFL variables is also present in some other FULL-POS routines (for example **PHYMFPOS**, **ENDPOS**).
- Although some cleaning has been done in the past, **POS** remains very long: around 3000 lines in CY35T2.
- There are too many dummy arguments (91 arguments in cycle CY35T2).
- Tests on LECMWF should be avoided in **POS** (the corresponding piece of code must be moved in a set-up routine) and more generally in the vertical interpolator of FULL-POS (where I have noticed several occurrences).

# 3 Proposals for recoding: POS.

In this section we focus on routine **POS**. About **APACHE** and some other routines involved in the vertical interpolator of FULL-POS, only collateral effects of the **POS** recoding will be mentioned.

## 3.1 Dynamics and compliancy with NH models and/or deep-layer formulations.

∗ **Main guidelines:** The main guidelines to rewrite parts 1.2 and 1.3 are the following ones:

- Have two distinct parts, the first one computing all the dynamical quantities which are also required in the model (i.e. found in **CPG_GP** for example), the second one computing pure diagnostic quantities (not required in the model, like the potential vorticity). Additionally to that, we plan to compute the first set of dynamical variables in all cases required by the dynamics, even if they are not-processed (no condition will be admitted other than LFPART2 and dynamical keys such as LNHDYN, NVDVAR, NPDVAR, LVERCOR, LRWSDLR, LVERTFE). That means for example that calculation of $\omega/\Pi$ will be always done. This first part will be recoded more consistently to what is done in **CPG_GP**: that will allow in the future an easier updating of this part of **POS** according to the modifications brought in **CPG_GP**.
- Gather all the dynamics in parts 1.2 and 1.3, avoid to recall GP.. routines in the part doing interpolations (for example under PP... routines). If this is not possible, make these additional calls to GP.. routines consistent with the type of dynamics used (NH vs hydrostatic, VFE vs FD, deep-layer vs thin-layer).

∗ **More details about rewriting the first class of post-processable variables:** The first class of post-processable variables, those which are also computed in the model integrations, includes the following list:

- $c_p$, $R$, $R/c_p$ (routine **GPRCP**).
- $RT$ and $\vec{\nabla}(RT)$ (routine **GPRT**).
- $\Pi$ at full and half levels, $\delta$, $\alpha$ and some other quantities linked with the hybrid coordinate (routines **GPPREH**, **GPXYB**, **GPPREF**).
- $\vec{\nabla}\delta$, $\vec{\nabla}\alpha$ (routine **GPGRXYB**).
- Pressure departure $p-\Pi$ and some other quantities linked with the pressure departure (routine **GNHPRE**).
- $\vec{\nabla}\Phi$ (routine **GPGRGEO**).
- $\dot{\eta}\frac{d\Pi}{d\eta}$ and $\frac{\omega}{\Pi}$ (routine **GPCTY**).
- Vertical divergence $d$ (routine **GPVERDIA** or some other routines).
- $w$ and $-G\Delta w$ (via routine **GPGW**).
- Geopotential height $\Phi$ (routine **GPGEO**).

Most of these calculations are in part 1.2 of **POS**, but calculation of $\Phi$ appears at the beginning of part 1.3.
We cannot replace this part by a call to **CPG_GP**, because **CPG_GP** computes quantities at two different instants and does computations which are useless in **POS**; furthermore **POS** must take account for the time being of the case LFPART2, not present in **CPG_GP** (this case is expected to disappear in the future when rewritting the 927 configuration but no one is currently available to do this work).
To be more consistent to what is done in **CPG_GP**, it would be recommended to compute these quantities in the following order, keeping the following conditions (works at least for model dataflow, to be checked if this is still valid for file dataflow too):

- $\Pi$ at full and half levels, $\delta$, $\alpha$ and some other quantities linked with the hybrid coordinate (routines **GPPREH**, **GPXYB**, **GPPREF**).
  Calculation of ZSPL and ZSPM must be always done.
- Calculation of ZRPRESF (under LVERTFE), then calculation of $\vec{\nabla}\delta$, $\vec{\nabla}\alpha$ (routine **GPGRXYB**).
  Call to **GPGRXYB** must be always done.
- $c_p$, $R$, $R/c_p$ (routine **GPRCP**).
  Call to **GPRCP** must be always done.
- $RT$ and $\vec{\nabla}(RT)$ (routine **GPRT**).
  Call to **GPRT** must be always done.
- LVERCOR=T calculations (at least $a/r$, $r/a$, $\vec{\nabla}(r/a)$).
  - LVERCOR=T: call to **GPVCTS**, **GPVCRS**, **GPGRVCRS** to compute ZVCTS0F, ZVCTS0FL, ZVCTS0FM (reference profile of temperature and its gradient), ZVCRSSA0F, ZVCRSSA0FL, ZVCRSSA0FM, ZVCRSSA0H ($r/a$, $\vec{\nabla}(r/a)$), ZVCASRS0F, ZVCASRS0H ($a/r$).
  - LVERCOR=F: ZVCTS0F=0., ZVCTS0FL=0., ZVCTS0FM=0., ZVCRSSA0F=1., ZVCRSSA0FL=0., ZVCRSSA0FM=0., ZVCRSSA0H=1., ZVCASRS0F=1., ZVCASRS0H=1. .

  ZVCRSSA0F, ZVCRSSA0FL, ZVCRSSA0FM, ZVCRSSA0H, ZVCASRS0F, ZVCASRS0H are required as input arguments of **GPCTY**; ZVCRSSA0F and ZVCASRS0F are required as input arguments of **GPPVO**.
  Calculations must be done according to the value of LVERCOR, but no other condition should appear.

- Pressure departure $p-\Pi$ and some other quantities linked with the pressure departure (routine **GNHPRE**).
  - LNHDYN and LFPART2: transfer ZSPD into ZPDT0; compute ZRRED0 (contains $R\frac{\Pi}{p}$).
  - LNHDYN and .NOT.LFPART2: call **GNHPRE** and **GNHGRPRE**; compute ZRRED0 (contains $R\frac{\Pi}{p}$).
  - .NOT.LNHDYN: ZPDT0=0., ZNHPPI=1., ZRNHPPI=1., ZNHPREF=ZPRESF, ZQCHAL=0., ZQCHAM=0. .

  Calculations must be done according to the values of LNHDYN and LFPART2, but no other condition should appear.
- LRWSDLR=T calculations: if LNHDYN.AND.LRWSDLR=T, compute $a/r$, $r/a$, $\vec{\nabla}(r/a)$, $Gr$ (geopotential height) and $\vec{\nabla}(Gr)$.
  Call to **GNHDLR** to fill the part of ZRDLR0 containing $r/a$ and $a/r$, ZGEOPH ($Gr$ at half levels), ZGEOPF ($Gr$ at full levels).

  Call to **GNHGRDLR** to fill the part of ZRDLR0 containing $\vec{\nabla}(r/a)$, (ZGPFL,ZGPFM), (ZGPHL,ZGPHM) (respectively $\vec{\nabla}(Gr)$ at full and half levels).
  ZRDLR0 is required as input argument of **GPCTY**, **GPXX**, **GPGW**. It would be useful as input argument of **GPPVO** when LRWSDLR will be implemented inside, but for the time being we assume that **GPPVO** is called out of the frame of LRWSDLR=T deep-layer NH equations calculations.
  No other condition than LNHDYN.AND.LRWSDLR=T should appear.
- $\dot{\eta}\frac{d\Pi}{d\eta}$ and $\frac{\omega}{\Pi}$ (routine **GPCTY**), then calculation of ZETADOT (according to the value of LVERTFE).
  Call to **GPCTY** and calculation of ZETADOT must be always done.
- Geopotential height $\Phi$ (routine **GPGEO**). Input data ZR0 must be replaced by ZRRED0 if LNHDYN=T.
  Calculations must be done according to the values of LNHDYN and LRWSDLR (**GPGEO** should not be called if LNHDYN.AND.LRWSDLR), but no other condition should appear.
- $\vec{\nabla}\Phi$ (routine **GPGRGEO**).
  Call to **GPGRGEO** must be done if (LNHDYN.AND.(.NOT.LRWSDLR)) or (.NOT.LNHDYN)). No other condition should appear.
- Half level horizontal wind, stored in (ZUH,ZVH): routines **GPHLWI**+**GPHLUV**.
  This calculation must be always done.
- Routine **GPUVS** with LLDER=.FALSE. .
  Call to **GPUVS** must be always done.
- Calculation of the $NHX$ term $(d_4 - d)$ if .NOT.LFPART2., stored in ZX0.
  - LSLAG.AND.LNHDYN.AND.(NVDVAR=4).AND.(ND4SYS=2): simple copy from PGMV (cf. what is done in **CPG_GP**).
  - otherwise: call to **GPXX**.

  No other condition should appear.
- Vertical divergence $d$ if .NOT.LFPART2.
  - LNHDYN and NVDVAR=3: ZDVER0=PSVD.
  - LNHDYN and NVDVAR=4: ZDVER0=PSVD-ZX0; remove some obsolete and false comments saying that this option is not ready.
  - .NOT.LNHDYN: compute $d_{hyd}$. Call to **GPVERDIA** then ZDVER0=(ZR0/RD)*ZVDT0. Or maybe simply ZDVER0(.,.)=-(ZR0(.,.)/RD)*(ZDIVT0(.,.)+ZX0(.,.) +(1.-ZKAP0(.,.))*ZVVEL(.,.)) to match what is done in **CPG_GP** (in this case routine **GPVERDIA** becomes useless).

  Calculations must be done if .NOT.LFPART2, according to the value of LNHDYN and NVDVAR, but no other condition should appear.
- $w$ (via routine **GPGW**) and $\Delta w$ at full levels.
  The first difficulty to update this part is that, for VFE (LVERTFE.AND.LVFE_GW=T), **GPGW** provides full level $Gw$, but the following parts of **POS** wait for half level $w$. The approximation currently done is to do computations always in FD and we will keep that for the time being.
  The second difficulty is that, for the time being, deep-layer NH equations are not taken into account in this part of **POS**.
  Recent developments on **GPGW**, combined with minor developments to be done (possibility to get the optional dummy argument PGDW also for FD vertical discretisation), would allow to obtain both half level $Gw$ and full-level $G\Delta w$, taking properly account of the NH deep-layer equations when relevant. That will allow to simplify this part of **POS**.
  Finally, the updated code of this part will look like:

– If .NOT.LFPART2: keeps LLVFE always to F.
Input data: ZDVER (contains $d$).
Call to **GPGW** with the right optional dummy arguments (including input LDGDWI=F, output PGDW): provides $Gw$ at half levels and $G\Delta w$ at full levels.
Divide by $G$ and fill ZWWT0 by $w$ at half levels; fill ZVDT0 by $-G\Delta w$ at full levels.

– If LFPART2: keeps LLVFE always to F.
Input data: PSVD (contains $-G\Delta w$).
Copy PSVD in ZVDT0.
Call to **GPGW** with the right optional dummy arguments (including input LDGDWI=T): provides $Gw$ at half levels.
Divide by $G$ and fill ZWWT0 by $w$ at half levels.

Additionally to that:

– correct some false comments: output ZVDT0 contains $-G\Delta w$.

No other condition than LFPART2, LVERTFE, LVFE_GW should appear. The upper air $w$ and $-G\Delta w$ will be always computed.

- At this level we may put the additional pressure variables conversions required for LRWSDLR=T (code which is currently at the end of part 1.2.2 of **POS**), and ignore for the time being the LRWSDLR=T effects for the calculation of purely diagnostic post-processable quantities (see second class below). In theory, conversions would be also required for $\dot{\eta}\frac{d\Pi}{d\eta}$ and $\frac{\omega}{\Pi}$ but I don't know really how to do them: they are currently ignored.

Note that the case LNH_GEOGW has been ignored for the time being.
These calculations must be always done, that means that the conditions appearing in parts 1.2.0a and 1.2.0b must be adapted.

### $*$ More details about rewriting the second class of post-processable variables:

The second class of post-processable variables, those which are only diagnosed, includes the following list:

- Potential temperature $\theta$ and potential vorticity $PV$ (routine **GPPVO**).
- Deformations.
- Equivalent potential temperature (routine **GPEPT**).
- Isobaric equivalent temperature (routine **GPIET**).
- Simulated reflectivity (routine **GPPRS0D**).
- Virtual temperature (routine **PPCVIRT**).

All these calculations are in part 1.3 of **POS**, and we can add the calculation of relative humidity $RH$ (routine **GPRH**) which is done later in the code (part 2.2.9).
No significant change has to be expected for these variables, excepted for the call to **GPRH** which must be in part 1.3 rather than in part 2.2.9. Conditions to call these calculations may remain unchanged.

### $*$ GP.. routines called under interpolations routines:

Here is the list of relevant routines:

- Routine **FPPS**: when called from **POS** the only GP.. routines called by **FPPS** are **GPRCP** and **GPGEO**.
- Routine **PPLETA**: calls **GPPREH**, **GPXYB**, **GPPREF**. (may generate inconsistencies for deep-layer NH model, no problem otherwise).
- Routines **PPGEOP** and **PPGEOP_OLD**: call **GPGEO**.
- Routine **PPRH**: calls **GPRH**.
- Routine **PPVVEL**: calls **GPCTY**.

More remarks and actions to be done about **FPPS**:

- The first remark we can do is that the calls to **GPRCP** and **GPGEO** done under the arborescence **POS** $->$ **FPPS** do exactly the same calculations as the calls of **GPRCP** and **GPGEO** done in parts 1.2 and 1.3 of **POS**.
- That means that the best thing is to move the calls to **GPGEO**, **GPRCP**, and also the calls to **GPPREH** and **GPXYB**, out of **FPPS** (must be done in the callers of **FPPS** before calling **FPPS**).
- The new version of **FPPS** will keep part 3 of the current **FPPS**, with the following dummy arguments:
input: KPROMA, KST, KND, KOPLEV, POROG, PGEOP, PT, PR (=air constant), PSP, PST, PRESH, PLNPR, PALPH, PGEOPH. All these quantities are available once done the call to **GPGEO** in part 1.3 of **POS**.
output: PSPPP

- **FPPS** is also called by **APACHE** and **ENDPOS**. That means that we will need to adapt **APACHE** and **ENDPOS** too (add calls to **GPGEO**, **GPRCP**, **GPPREH** and **GPXYB** before the call to **FPPS**). Further optimisation may be expected in **APACHE** and **ENDPOS** (potential repetition of calls to **GPGEO**, **GPRCP**, **GPPREH** and **GPXYB** which actually compute the same quantities).

More remarks and actions to be done about **PPLETA**:

- This routine is called in part 1.4.2 of **POS** only if post-processing is done at $\eta$-levels which are not model levels (FPVALH and FPVBH different from VAH and VBH). This call may generate inconsistencies for deep-layer NH model, but no problem otherwise.
- No action is planned for **PPLETA**, but use of configurations which call **PPLETA** under **POS** in a deep-layer NH model (LRWSDLR=T) is forbidden for the time being.

More remarks and actions to be done about **PPGEOP** and **PPGEOP_OLD**:

- The call to **GPGEO** which is done in **PPGEOP** is not the same one as the one done in part 1.3 of **POS**, because it computes a geopotential increment from a temperature increment (close to a departure from standard atmosphere temperature?). It is not possible to move this call to **GPGEO** in part 1.2 of **POS**, because it requires the calculation of ZSTTF which is done in part 1.4.5 of **POS**.
- For thin-layer NH models, the current call to **GPGEO** in **PPGEOP** is not correct because in this case the input dummy argument for air constant must be ZRRED0 and not ZR0 (PR0) to take account of the factor $\Pi/p$.
- For deep-layer NH models, the current call to **GPGEO** in **PPGEOP** must be in theory replaced by a call to **GNHDLR**, but in practical this is impossible to do it at this location or even just before the call to **PPGEOP**, because at the location where **PPGEOP** is called in **POS**, some data linked with apparent pressure coordinate are no longer known (contrary to part 1.2 of **POS**). The vertical interpolator only knows genuine hydrostatic pressure and not features linked with the deep-layer NH model.

  The only way to deal with that is to keep the thin-layer approximation to compute the increment of geopotential to be interpolated, assuming that we have quantities $\alpha$, $\delta$ consistent with the genuine hydrostatic pressure and not something linked with the apparent vertical pressure coordinate. The conversion which is expected at the end of part 1.2 of **POS** provides acceptable values for these quantities. This choice has probably limitations for idealised simulations like "small planet" ones.
- To sum-up, with the choices of dynamics we currently have, we can stick to adapt the input dataflow of **GPGEO** in order to provide ZRRED0 instead of ZR0 (PR0 in **PPGEOP**). This can be done keeping the call to **GPGEO** in **PPGEOP**.
- But I don't guarantee that future possible alternate formulations of NH dynamics (for example dynamics using the geopotential height as prognostic variable) will not raise new problems, difficult to solve with the current version of **PPGEOP** which assumes that the geopotential height is always diagnosed.
- We can notice that PR0 is used in other parts of **PPGEOP** and we must know if it must be replaced by ZRRED0 or not.
  - Factor RD/PR0 in the RHS of calculation of ZSTTF allows to convert a virtual reference temperature into a real reference temperature: no change is expected.
  - PR0 is used after the call to **GPGEO** to convert the temperature increment into $R$ times the temperature increment: this $R\delta T$ is required for interpolations or extrapolations below the lowest full level. It is not clear in my mind if NH effects linked to pressure departure must be taken into account in this part (and how) but this is an issue. The use of PR0 will be provisionally unchanged in this part.
- I now sum-up the actions to do on **PPGEOP** and **PPGEOP_OLD**:
  - An additional input dummy argument PRRED0 will be passed (containing $R$ if HYD, $R\frac{\Pi}{p}$ if NH), just after PR0. PR0 will be replaced by PRRED0 in the calls to **GPGEO**.
  - The calls to **GPGEO** remain in **PPGEOP** and **PPGEOP_OLD**.
  - The callers of **PPGEOP** must be adapted. In **AVAL** and **PPOBSA** the same quantity will be passed in dummy arguments PR0 and PRRED0 because the NH dataflow is not ready in these routines.
  - No action is required for the time being in the TL and AD codes because the TL and AD codes of the NH model are not yet implemented.

More remarks and actions to be done about **PPRH**:

- There is no difficulty to move the call to **GPRH** out of **PPRH**, and this is even desirable for the call to **PPRH** done in **POS**. Additionally we can notice that a call to **PPRH** is equivalent to a call to **GPRH**+**PPQ**.
- Call **GPRH** in part 1.3 of **POS**; use the output quantity containing relative humidity as input to the revised version of **PPRH**.

- **PPRH** is also called by **PPOBSA** and **AVAL**: in these routines call **GPRH** just before the call to the revised version of **PPRH**.
- The revised **PPRH** becomes identical to **PPQ**: use **PPQ** and remove deck pprh.F90.
- The same kind of work must be done on the TL and AD codes: **PPRHTL** (called by **PPOBSATL**) and **PPRHAD** (called by **PPOBSAAD**).

More remarks and actions to be done about **PPVVEL**:

- There is no difficulty to move the call to **GPCTY** out of **PPVVEL**, and this is even desirable. **GPCTY** does exactly the same calculations as the call to **GPCTY** done in part 1.2 of **POS**.
- Call **GPCTY** in part 1.2 of **POS**; use the output quantity containing $\dot{\eta}\frac{d\Pi}{d\eta}$ and $\frac{\omega}{\Pi}$ as input to the revised version of **PPVVEL**.
- The new version of **PPVVEL**, will have the following dummy arguments:
  input: KPROMA, KST, KPROF, KFLEV, KLEVP, KLOLEV, KLEVB, PRPRES, LDBELO, LDBLOW, PRXP, PRXPD, PRDELP, PEVEL, PVVEL.
  output: PVVPP.
  optional input: LDETADOT.
- **PPVVEL** is called only by **POS**.
- Additionally, remove the obsolete comments in part 2.2.10 (old call to **GPCTY**).

## 3.2 Global treatment of GFL and CUF post-processable quantities.

∗ **GFL:** It would be desirable to have a global treatment of post-processable GFL variables (introduction of a specific vertical interpolator GFL structure), as it is already done for the GFL variables in the dynamics. That will allow to shorten **POS** and also to have a quasi-automatic update of **POS** when adding new post-processable GFL variables. This code evolution would require at least the following features:

- Definition of two GFL structures, the first one for input data of **POS** (containing derivatives), the second one for keys LL_[X] and interpolated GFL.
- Replace input dummy PQ, PQL, PQM, PL, PI, PS, PR, PGR, PHL, PTKE, PEXT, PEXTL, PEXTM, PA, PO3, PUAL, PUOM, PDAL, PDOM, PUEN, PUNEBH, PAERO, PGHG, PTRAC, PGRG, PLRCH4, PCH4S, PERA40, by PGFL. This PGFL array must be flexible (this is not necessary GFL, that can be something dimensioned with NFPROMA). Order of storing data in PGFL must be defined in the caller of **POS**.
- Replace input dummy LDQ, LDQDER, ... , by a new variable LDGFL.
- Use a local array LL_GFL instead of the LL_[X].
- Use a local array ZGFLT0 instead of the Z[X]T0 (may contain derivatives).
- Ordering of GFL variables in PGFL, LDGFL, ZGFLT0, must be consistent, and defined (or known) in the caller of **POS**. To define it in **VPOS**, we can for example take the same ordering as in GFL.
- Ordering of GFL variables in LL_GFL, ZPPGFL, must be consistent: we can for example take the same ordering as in GFLT1, or define a specific order for post-processable GFL variables. This order will be defined in part 1.1.1 of **POS**.
- In part 1.1.1 fill LL_GFL, and also some other GFL conditions to do interpolations in part 2.1.1 and calls to **FILL_PP3** in part 3.1.
- In part 1.2.0a fill ZGFLT0.
- In part 1.2.8 fill ZPPGFL with zeros.
- Use ZGFLT0 where GFL variables are used (input to GP.., GNH.., PP.., **FPPS**, **APACHE**).
- Rewrite part 2.1.1 with a global treatment of GFL (one loop on JGFL, one call to **PPQ**, fill ZPPGFL).
- Rewrite part 3.1 with a global treatment of GFL (one loop on JGFL, one call to **FILL_PP3**)

Additional modifications are expected in **VPOS** (call to **POS**), **SCAN2M** (call to **VPOS**). Globalisation of GFL may be later extended to other FULLPOS routines.

∗ **CUF:** It would be desirable to have a global treatment of post-processable CUF variables. The maximum number of CUF is JPMFNR (in **PARMCUF**). This code evolution would require at least the following features:

- Replace input dummy PRMCUFGP1, PRMCUFGP2, PRMCUFGP3, PRMCUFGP4, PRMCUFGP5 by PRMCUFGP, dimensioned with JPMFNR.
- Replace TFP_CUF1 to TFP_CUF5 in **YOMAFN** (and all other routines where such variables are used) by TFP_CUF dimensioned with JPMFNR, or use TFP_DYNDS.
- Replace LL_CUF1 to LL_CUF5 by LL_CUF dimensioned with JPMFNR.
- Replace ZPPCUF1 to ZPPCUF5 by ZPPCUF dimensioned with JPMFNR.

- Part 1.1.2: CUF code becomes a loop on JCUF from 1 to JPMFNR filling LL_CUF from QFPTYPE%LL(TFP_CUF%ICOD).
- Part 1.2.8: CUF code becomes a loop on JCUF from 1 to JPMFNR filling ZPPCUF with zeros.
- Part 2.2.14: CUF code becomes a loop on JCUF from 1 to JPMFNR filling ZPPCUF(.,JCUF) from PRMCUFGP(.,JCUF).
- Part 3.2: CUF code becomes a loop on JCUF from 1 to JPMFNR calling FILL_PP2.

Additional modifications are expected in **VPOS** (call to **POS**), **SCAN2M** (call to **VPOS**). Globalisation of CUF may be later extended to other FULLPOS routines.

### 3.3 Reduction of the number of dummy arguments.

**POS** has 91 dummy arguments in CY35T2. Global treatment of GFL and CUF (see above) would allow a significant reduction of this number, but some other groups of dummy arguments may be shortened (for example pass PGMV and PGMVS as dummy arguments instead of the individual GMV and GMVS variables and their derivatives). It is desirable to have the dummy argument KPROMA instead of NPROMA because future uses of **POS** may use something different (for example NFPROMA, like in **ENDPOS**).
Additional modifications are expected in **VPOS** (call to **POS**), **SCAN2M** (call to **VPOS**).

### 3.4 Miscellaneous features.

- Tests on LECMWF should not appear in **POS**. The only location where it appears is to set-up a local variable ZRHMAX (equal to 1. or 2. according to LECMWF). I recommend instead to have a module variable VRHMAX in **YOMDYN**, set-up in **SUDYN** (for example in part 1.17), or several versions of this variable. Some PP.. routines called under **POS** also use LECMWF.

  Some occurrences of ZRHMAX are present elsewhere in the code, with the same meaning (maximum relative humidity), but not always the same value: example ZRHMAX=1 in **SURBOUND**, ZRHMAX=1.2 in **HRETR** and **PPOBSA**. This variable is often used as input variable of **GPRH**.
- Correct false comments (array ZVDT0 which contains $-G\Delta w$, array PSVD which contains the vertical divergence variable if LFPART2=F and $-G\Delta w$ if LFPART2=T).

## 4 Proposals for recoding: APACHE+AVAL.

The same kinds of recodings described for **POS** can be extended to **APACHE**+**AVAL**.

### 4.1 Global treatment of GFL post-processable quantities.

The POS GFL structure must be extended to **APACHE** and **AVAL**. After that, **APACHE** and **AVAL** will also treat $q_{GHG}$, $q_{TRAC}$, $q_{AERO}$, $q_{GRG}$, $q_{EXT}$ (this is currently not the case). This work will allow to replace 5 calls to **APACHE** in **POS** (and also in **ENDPOS**) by one.
**AVAL** will be in-lined in **APACHE**. If there is no objection for that, **APACHE** will be in-lined in its callers (**POS**, **ENDPOS** and **PPOBSAP**).
In parts 1, 2 and 3 of **AVAL**, GFL treatment must be done in separate subsections.

### 4.2 Gathering dynamics.

When possible, dynamics (call to GP.. and GNH.. routines) must be done in the caller of **APACHE**+**AVAL** and not in **APACHE**+**AVAL**. This is possible at least for the following calls:

- Calls to **GPPREH**+**GPXYB**+**GPPREF** done in part 1.3 of **APACHE** must be done in the caller. They are identical to those done in part 1.2 of **POS**.
- Call to **GPRCP** done in part 1.6 of **APACHE** must be done in the caller. It is identical to the one done in part 1.2 of **POS**.
- Dynamics done in **FPPS** (part 2.1 of **APACHE**) must go out of **FPPS**, and be done by the caller of **APACHE**; re-use for example calculations done in part 1.2 of **POS**.
- The call to **GPPRE** in part 3.1 is not identical to what is done in part 1.2 of **POS** (use of a different surface pressure) but this call can be moved just after the call to **FPPS**.
- There are also dynamics recalculation in **AVAL** which must be moved in the callers of **APACHE** (for example re-use calculations already done in part 1.2 of **POS**).

### 4.3 More remarks about AVAL.

**AVAL** has a structure similar to **POS**: it must be reorganised like in **POS** (memory transfers, then dynamics, then preparation of interpolations, then interpolations, with separated subsections for GFL). Use of LECMWF in **AVAL**, forbidden at this level of the code, must be replaced by something else: introduce a key LFPESCALE and replace .NOT.LECMWF by LFPESCALE.

Additionally, in-lining **APACHE+AVAL** in **POS** would allow to merge some parts:

- part 1 of **AVAL** and part 1.2 of **POS** (memory transfers and dynamics).
- part 2.1 of **AVAL** and part 1.4 of **POS** (preparation of interpolations).
- parts 2.2 to 2.13 of **AVAL** and parts 2.1 to 2.5 of **POS** (interpolations). All GFL treatment must be put in part 2.1 of **POS**.
- Additional interpolations under LFPESCALE: preparation of interpolations must go in part 1.4 of **POS**; interpolations must go in part 2.1 of **POS** for GFL, 2.2 to 2.5 of **POS** for other quantities.

## 5 Proposals for recoding: ENDPOS, PHYMFPOS, PPOBSAP and some other miscellaneous features.

### 5.1 ENDPOS.

**ENDPOS** has a structure similar to **POS** (but it is simpler) and we propose to rewrite them according to the rewritings done in **POS**:

- The order of calculations will be: memory transfers, then dynamics (consistent with NH and deep-layer formulations), then preparation of interpolations, then interpolations, then store data in PFP.
- If agreed, **APACHE+AVAL** will be in-lined in **ENDPOS**.
- The GFL structure will be introduced in **ENDPOS**, and GFL treatment will be done in separated sub-sections.
- Sections and subsections numbering will be done consistent between **POS** and **ENDPOS** to allow a future merge of these two routines.
- The number of dummy arguments will be reduced: pass PGMV and PGFL instead of the individual variables.

### 5.2 PHYMFPOS.

**PHYMFPOS** does not perform interpolations, but we retrieve structures similar to parts 1 and 3 of **POS**. Like in **POS**, it is desirable to have the following rewritings:

- Global GFL treatment.
- Part 1: memory transfers in part 1.1, with a separated sub-paragraph for GFL variables.
- Part 1: dynamics in part 1.2, including the dynamics currently done in **FPACHMT** which must go out of **FPACHMT**. Make this dynamics "NH and deep-layer formulations" consistent.
- The number of dummy arguments will be reduced: pass PGMV and PGFL instead of the individual variables.

### 5.3 PPOBSAP.

In-lining **APACHE+AVAL** in **PPOBSAP** will allow **PPOBSAP** to have a structure similar to **POS** and **ENDPOS** (but simpler).

- The order of calculations will be: memory transfers, then dynamics (consistent with NH and deep-layer formulations), then preparation of interpolations, then interpolations, then store data in PXPP and optionally in ROBODY.
- The GFL structure will be introduced in **PPOBSAP**, and GFL treatment will be done in separated sub-sections.
- Sections and subsections numbering will be done consistent between **POS** and **PPOBSAP**.
- The number of dummy arguments will be reduced: pass PGMV and PGFL instead of the individual variables.

### 5.4 FPACHMT.

Parts 1 to 3 (dynamics) will go out of **FPACHMT** (these calculations are generally already done in the callers); only part 4 will remain in **FPACHMT**.

# 6 Miscellaneous actions.

I recommend to merge **PPGEOP** with **PPGEOP_OLD**, **PPT** with **PPT_OLD**, **PPUV** with **PPUV_OLD**: both versions are not very different, and the key LOLDPP must act on the relevant differences only and not of the whole code. Use LECMWF under the OLD versions (forbidden at this level of the code) must be replaced by something else (specific key to be updated in **CNT0** and **CPREP4** just after LFPART2). TL and AD codes must be updated according to the direct code.

**BOB** (called by **AVAL**) is very close to **PPQ**; it must be replaced by **PPQ** (with a minor adaptation of **PPQ**).

# 7 Conclusion and perspectives.

## 7.1 Conclusion.

At a first glance, rewrite the dynamics in order to have a proper consistency with NH/deep-layer dynamics when relevant, and introduce a GFL structure, in the vertical interpolator, can be seen as two independent tasks. Indeed, the above mentioned work leads to recommend the following order for the different phases of the recoding:

- Introduce a GFL structure in **FPOS**, **ENDPOS**, **APACHE**+**AVAL**, and also in **PHYMFPOS** and **PPOBSAP**.
- Use this structure to have a global treatment of GFL.
- Put GFL treatment in separate sub-sections.
- Replace 5 calls to **APACHE** by one in **POS** and **ENDPOS** (treating all GFL).
- In-line **AVAL** into **APACHE**.
- If agreed, in-line **APACHE** into **POS**, **ENDPOS** and **PPOBSAP**.
- Global treatment of CUF in **POS**.
- Reduce the number of dummy arguments in **POS**, **ENDPOS**, **PPOBSAP** and **PHYMFPOS**. Replace NPROMA by KPROMA in **POS**.
- Make consistent the calculations ordering and paragraph numbering in the following routines: **POS**, **ENDPOS**, **PPOBSAP**, **PHYMFPOS**.
- Take out GP.. routines from **FPPS**, **PPRH**, **PPVVEL**, **FPACHMT**.
- Gather all the GP.. and GNH.. calls which can be gathered at the beginning of routines **FPPS**, **PPRH**, **PPVVEL**, **FPACHMT**.
- Make these calls to GP.. and GNH.. routines consistent with the NH or the deep-layer dynamics formulations; in particuliar in **POS** re-order the dynamics calculations as mentioned above. For calls to **GPGRP** it would be desirable to take account of the same list of hydrometeores everywhere in the vertical interpolator.
- Try to take properly account of NH dynamics in the GP.. routines which remain spread in the code (in particuliar **GPGEO**).
- Miscellaneous actions: merge PP... routines with their "old" versions when relevant, replace forbidden use of LECMWF by appropriate keys.

This is a huge task which must be a target for CY37 (CY36T1 on MF side).

An alternate choice would be to keep the introduction of a GFL structure at the end of the work, but that means in this case that we will have a very long version of **POS** between in-lining **APACHE**+**AVAL** and before introduction of a GFL structure.

## 7.2 Perspectives and other remarks.

Further actions can be studied: in **POS**, **ENDPOS** and **PPOBSAP**, gather interpolations variable by variable (interpolations coming from **POS**, interpolations coming from **AVAL** and specific LFPESCALE interpolations coming from **AVAL**); unification of **POS** and **ENDPOS**.

Rewriting conf 927 of FULL-POS in order to use modular spectral transforms and to remove case LFPART2=T case is another task which becomes stringent, but we have to find someone which has time to do that. A consequence of this action may be the call of **POS** with NFPROMA-dimensioned arrays.

Similar inconsistencies between dynamics and interpolators using GP.. routines may exist in the observation interpolator (example **PPGEOP** called by **PPOBSA**): for the time being consistency is ensured when the observation interpolator is called in a thin-layer hydrostatic model but I don't guarantee full consistency in NH models.

# Appendix A: Current use of GFL variables in POS, PHYMFPOS, APACHE+AVAL, ENDPOS.

| Variable | POS | PHYMFPOS | APACHE+AVAL | ENDPOS |
|---|---|---|---|---|
| q | yes | yes | yes | yes |
| q_l | yes | yes | yes | yes |
| q_i | yes | yes | yes | yes |
| q_a | yes | yes | yes | yes |
| q_O3 | yes | | yes | yes |
| q_UAL | yes | | | |
| q_UOM | yes | | | |
| q_DAL | yes | | | |
| q_DOM | yes | | | |
| q_UEN | yes | | | |
| q_UNEBH | yes | | | |
| q_s | yes | yes | yes | yes |
| q_r | yes | yes | yes | yes |
| q_g | yes | yes | yes | yes |
| q_h | yes | | yes | yes |
| q_TKE | yes | | yes | yes |
| q_LRCH4 | yes | | | |
| q_CH4S | yes | | | |
| q_GHG | yes | | missing | yes |
| q_TRAC | yes | | missing | |
| q_AERO | yes | | missing | yes |
| q_GRG | yes | | missing | yes |
| q_EXT | yes | | missing | yes |
| q_ERA40 | yes | | | |
| q_EZDIAG | yes | | | |