PROPOSAL OF CLEANINGS IN ARPEGE/IFS IN 2013-2014.

YESSAD Karim.

June 6, 2013

Version 9b. Basis of study: CY40

1 Introduction and purpose.

This paper is an updated (and shortened) version (basis CY40) of a paper written in 2007, and we don't recall all the introduction of the previous versions. To sum-up we briefly re-call the purpose of this paper and of the actions listed inside.

- There is now slightly more than 3000000 code lines.
- There is a necessity to do regularly cleaning actions in the code, in particuliar for the following topics: proper naming of routines, prune obsolete code and obsolete variables, better use of F90 features to improve the code concision, improving comments, improving the library architecture, externalisations, reduce useless code duplication, respect to the coding standards.

This version focusses on actions which should be done during the next two years (targets: cycles 41, 42 and 43). Three levels of priority to do these cleanings have been defined:

- (P0): the high level priority tasks which should be done for CY41.
- (P1): the medium level priority tasks which should be completed for CY42.
- (P2): the low level priority tasks which can be completed after CY43.

This "V9" version of "cleaning memorandum" takes account of CY40. Some actions described in this document can also be seen as preliminary cleaning actions for the future OOPS project.

2 Cleaning of obsolete features.

2.1 Cleaning of obsolete options.

The following options can be good candidates for pruning.

- (P1) Remove old sponge (SUPONG, YOMPONG).
- (P1) Remove option LHV for vertical interpolations (Hermite cubic ones).
- (P1) Remove option NPROFILEHD=4 in horizontal diffusion (obsolete version of NPROFILEHD=3).
- (P1) Remove options LRETCFOU/LWRTCFOU (transmission coefficients for radiation stored in Fourier space).
- (P1) Remove the following keys and involved code (under .F. or .T.?): LOBSTL, LSIMOB, LOBSREF, L131TL.
- (P1 or P2) In the calculation of the displacement in the SL scheme, remove the useless option LELTRA=T.
- (P1 or P2) Remove first-order uncentering VESL>0 in the SL scheme.
- (P1 or P2) Remove case (LSETTLS,LPC_NESC)=(F,F) in the SL scheme.
- (P1 or P2) NH model: remove case (LRDBBC,LGWADV)=(F,F) which generates chimneys above slopes.

Keeping or pruning the following options is also questionable in the future.

- (P2): Some values of NLANTYPE seem to be not used in configuration 601 and the corresponding code is not regularly validated. What is used in the MF PEARP (ensemble prevision at METEO-FRANCE) is NLANTYPE=1 (the NLANTYPE=6 code may be called in a configuration 601 with NLANTYPE=1). NLANTYPE=5 is maybe useful also. Options which seem useless are NLANTYPE=2,3,4,7,8,9,10,11.
- (P2): remove the minimizer N1CG1 (no longer used) and all the pieces of code using it.
- (P2): configuration 923: remove parts 2 to 10 (when they will be replaced by something else to make surface climatologies).
- (P2): CANARI: remove features doing upper-air OI assimilation (only surface assimilation is still used).
- (P2): Conf 801: remove options NJROPT=2 or 3.

2.2 Cleaning of obsolete (unused) variables.

Some variables have been used in the past; they are now not used any longer, but their declaration and set-up has not been completely removed. Most of them have now been removed in the ARP/IFS project; occurrences may still occur in some other projects.

Actions and priority:

- (P2): remove useless variables in the other projects.

2.3 Cleaning of obsolete commented pieces of code.

There are obsolete commented pieces of code spread in the code (old pieces of code which have become obsolete). These pieces of code must be removed especially in routines containing a lot of that.

Actions and priority:

- (P1+P2): remove obsolete commented pieces of code.

2.4 Removal of obsolete routines.

Actions and priority:

- **(P0): remove obsolete routines listed in appendix H.**

3 Improvement of commenting.

A lot of routines may be insufficiently commented, or may contain false comments. Concerning the last points, there have been code evolutions where the update of the corresponding comments (which were originally correct) has not been done properly, so the comments have become partially wrong during the time. It is not straightforward to search for all the ill-commented routines, and such work must be regularly done by the responsible people. In particular we can focus in two sets of comments:

- Comments in modules: remaining uncommented modules are listed in Appendix A.
- Comments in routine header to say briefly what the routine does: list routines where they are missing and add missing comments in English. A partial list is provided in Appendix B.
- Comments for dummy arguments: list routines where they are missing, and add missing comments in English. The comprehensive list of relevant routines is not provided here and is actually difficult to obtain (spread in all directories: probably a lot of routines; some of them have been found in ECMWF radiation routines or CANARI routines).
- Comments in French: translate them in English where no English comment is currently provided. French comments without any English translation are found mainly in the following routines:
  - arp/canari/ca...F90 routines (CANARI).
  - most of arp/dia DDH routines.
  - most of arp/function/qa...h routines (CANARI).
  - most of arp/phys_dmn routines.
  - most of xrd/ddh routines.
  - most of xrd/fa and xrd/lfi routines.
  - some of xla/internal/minim routines.

  Some other routines (not listed) may contain untranslated French comments.

  To sum-up, French comments are mainly localised in routines which are not used at ECMWF at all (DDH, LFI and LFA software, MF physics, CANARI OI assimilation) with some isolated exceptions for DDH, but they may be used by some of our HIRLAM or ALADIN partners: some actions to do English translation may be required if expressed by some of our HIRLAM or ALADIN partners.
- Isolated other language comments may appear (German for example).

Actions and priority:

- **(P0): add missing comments (or correct false comments) in declaration modules.**
- **(P0): add missing comments for routine meaning (header).**
- (P2): translate French comments into English in the DDH package and more generally in all routines with keyword "DIA".
- (P2): add missing comments for dummy arguments.
- (P2): add missing comments or update/correct comments in other routines.
- (P2): translate French comments into English in other parts of the code.

4 Improvement of the library architecture (projects and directories inside each project).

4.1 Right place of routines.

Cleaning has been done for CY35. For the following cycles, checking must been done for new routines to ensure they are properly named and placed in projects/directories. Expected routine displacement is listed in Appendix E.

∗ **Dynamical-physical interface routines:** The current status is that the dynamical-physical interface routines are not easy to recognise in the code (no specific prefix in their names; they are spread among several directories like adiab, phys_dmn, phys_ec or phys_radi). People working on physics or dynamics may need to work on these routines and to know what they are allowed or not allowed to do in these routines (for example is it allowed to call GPHPRE or not?). See documentation (IDKEYW) to know routines doing physics-dynamics interface.

The minimal proposal we can do is to move routines which are not in one of the "phys_.." directory in the appropriate "phys_.." directory (for example move CPSPE and POSTPHY in phys_ec), and to maintain a documentation where the list of dynamical-physical interface routines is provided.

∗ **Project UTI:** Arborescence in this project must be reconsidered, and still needs thinking.

∗ **Projects ODB, SAT, AEO:** It would be desirable to dispatch routines in directories "external", "include", "module", "programs", "internal" according to their content and the way how they are called. This topic still need discussions. A better separation between these projects and the other ones can take part of the "OOPS" reorganisation project.

∗ **Other topics (a proposal of movings has been given in Appendix E):**

- Place of routine xrd/eclite/uv2sd.F (which gives the argument and modulus of a vector) is questionable: project XRD or XLA?
- Moving in arp/fullpos is questionable also for pos.F90 and suppdate.F90.
- Routine arp/var/suvifce.F90 looks like a hat routine for pp_obs routines (vertical interpolator) and is used in applications which are not only variational ones: moving in arp/pp_obs is questionable.
- Routines like ald/utility/elalo2xy.F90, arp/transform/tracare.F90, arp/transform/trareca.F90, could be moved in xrd/utilities.

∗ **Priorities for these actions:**

- **(P0) Actions described in appendix E1.**
- **(P0) A subset of actions described in appendix E2: SUECHK, YEMCHK.**
- (P2) Appropriate storage of dynamical-physical interface routines.
- (P2): Reorganisation of project UTI according to the solution which will be retained.
- (P2): Reorganisation of projects AEO, ODB and SAT according to the solution which will be retained.
- (P1+P2) Other actions described in appendix E.

4.2 Externalisation.

∗ **Possible externalizations of tasks currently present in projects ARP/IFS and ALD:** What I call externalization is to put a set of routines in a specific project, so that this project can work as an independent one (that means that it can be in theory used by a caller software which is not necessary ARPEGE/IFS). One calls a header and this header calls internal routines of this project which can mostly be seen as a blackbox.

Here is the sum-up of some proposals coming from different people:

- Externalize the interpolators, and the halo calculation necessary for horizontal interpolators.
- Externalize parallel environment routines (currently in arp/parallel) in xrd/ifsaux.
- Externalize the observation operator.
- Externalize the vertical interpolator (used in FULL-POS for example).
- Externalize FULL-POS, or some parts of FULL-POS.
- Externalize configurations 9xx like 923, 931, 932 (but some of them may become obsolete in some years).
- Externalize the OASIS coupler.
- Externalize physics (with probably separate libraries for radiation, surface processes and other upper-air processes).

Note that the "OOPS" reorganisation project will probably require some externalisations.

Some remarks about these proposals:

- Observation operator, vertical interpolator of FULL-POS (also used in the observation operator), horizontal interpolator of FULL-POS: these tasks are not completely independent.
- Physics: the following sets are already externalised:

  - ECMWF surface scheme (project SUR).
  - surface scheme used in AROME (projects SURFEX and MSE).
  - upper-air microphysics used in AROME (project MPA).

The following sets remain to be externalised:

  - all radiation schemes (project RAD to create).
  - all surface schemes which are yet in ARP/IFS (project SUF to create).
  - all other upper-air physics routines which are yet in ARP/IFS (project UAP to create).

Priorities for externalizations:

- (P2) Externalisation of interpolators and halo calculation.
- (P2) Externalisation of conf 923, 931, 932 in UTI.
- (P2) Other externalisation tasks.

4.3 Removal of forbidden dependencies between projects.

Documentation (IDARC) gives allowed and forbidden dependencies between projects. Forbidden dependencies must be progessively removed.

Priorities for removal of forbidden dependencies:

- **(P0) Remove forbidden dependencies in uti/combi.**
- **(P0) Remove forbidden dependencies in xla.**
- (P2) Do an inventory of forbidden dependencies existing in the code
- (P2) Remove forbidden dependencies in the other projects, in particuliar in project ODB.

5 Misnamed routines and decks.

Some work has been done in CY35. About the following cycles, naming of new routines entering these cycles must be checked.
New proposals of decks renaming have been listed in Appendix E.

∗ Priorities for renaming:

- (P1+P2) for all renamings.

6 Misplaced variables.

Some variables are currently mis-placed (not in the right module or in the right namelist); there are sometimes inconsistencies between place in namelist, place in module, and set-up.

This work must now be considered in the OOPS frame; create mixed modules containing: variable declaration, set-up, allocation, deallocation, namelist reading. See documentation (IDOOPS) for more details about the possible reorganisation of variables.

7 Consistency between module and namelist naming.

Remark: this topic may be reconsidered in the OOPS frame (reorganisation of variables and set-up).

Provisionally it is desirable to have a better consistency between modules and namelist names (variable in namelist NAM[XXX] to be in module YOM[XXX] (or YEM[XXX], QA[XXX], [XXX]_MOD)).

∗ Priorities for ensuring name consistency between namelist and module:

- **(P0) Complete work for geometry object.**
- (P1)+(P2) Some other actions, done in the frame of the OOPS project (variables and set-up encapsulated in modules, and some new feature replacing namelists).

8 Duplicata.

∗ **Duplicata of names.** In this topic we can find:

- D1/ two variables in two different modules of the same project, having the same name (but not necessary the same meaning).
- D2/ two variables in two different projects, having the same name (but not necessary the same meaning).
- D3/ a variable has the same name as a subroutine, in the same project.
- D4/ a variable has the same name as a subroutine, but in different projects.
- D5/ a variable has the same name as a subroutine of the scientific library or an intrinsic routine.
- D6/ two routines have the same name, but in different projects (example: MINIMA).
- D7/ two different variables of the same project (in different modules) have exactly the same meaning (ex: ROMEGA in yomcst.F90 and OMEGA in yhlconst.F90).

Allowed and forbidden duplicata:

- D1 should be avoided. It does not generate any problem of compilation if both versions of the variable are not used in the same routine, but if a code development leads to use both version in a same routine, that will oblige to change the name of one of them.
- D2 is allowed (but misleading for the source navigator which does not properly distinguish the different versions).
- D3 should be avoided.
- D4 is not recommended, but it is difficult to completely avoid it: such occurrences are present in the code.
- D5 should be avoided, but there probably remains such occurrences in the code.
- D6 should be avoided if possible (there are occurrences of such duplicata in the code currently): it is desirable to ensure convergence between the two versions and to use only one of these versions in the future. If not possible the "duplicata" must be renamed (names keeping a common root).
- D7 should be avoided because that causes useless complexification of the code.

These types of duplicata have not been checked in a comprehensive way (this task is very difficult to do without an adequate automatic procedure which remains to be written). We can notice some existing duplicata:

- D1/ A lot of occurrences have been found, in particuliar in the ECMWF radiation.
- About D6, we must notice that there are no duplicata left (contrary to some years ago) between ARP/IFS and ALD. Some renamings have been done in CY35T2, but all occurrences have not been completely removed (two versions of MINIMA, in ARP/IFS and SCT).
- About D7, this case is encountered for the constants used in the HIRLAM physics: some variables of module yhlconst.F90 (generally not DOCTOR compliant) have their counterpart in yomcst.F90 (for example PI in yhlconst.F90 and RPI in yomcst.F90).

9 Insufficient use of the shortness possibilities for code writing.

No comprehensive list of pieces of code which can be significantly shortened will be given in this paper, because this is a huge task impossible to do. But we can explore some solutions and give example of pieces of code which can be shortened.

9.1 Features which do not use enough the F90 possibilities, or which generate too long pieces of code, and what to do?

∗ **Loops involving arrays: array-syntax or use of DO loops.** About this topic, a compromise must be found between the shortness of the code and its optimisation properties. The conclusions about the optimisation properties lead to use array syntax only for simple operations (like initializing or copying whole arrays) and to use DO loops otherwise (cf. recommandation CCPT(10) of documentation (IDCRT)).

∗ **Reducing the number of dummy arguments, and better use of the derivated type variables.** There are still routines with an outstanding huge number of dummy arguments, and an effort has to be done to match the CTRL(10) recommendation of documentation (IDCRT): avoid if possible to have more than 9 dummy arguments; prohibit routines with more than 50 dummy arguments. Appendix I provides a list of routines having more than 50 dummy arguments.

The use of derivated type variables can be a way to reduce the number of dummy arguments, and the data flow rewritings done these last years (implementation of the GMV and GFL data flow) has allowed to reduce the number of dummy arguments in routines called in the dynamics. This way must go further, and some other sets of variables might be transformed into new derivated type variables or global buffers. New proposals of derivated types have been listed in appendix G.

Some routines have useless dummy arguments (in particuliar dummy arguments labelled with the statement "Argument NOT used"): desirable to be removed. Appendix I provides a list of routines with occurrences of useless dummy arguments labelled with "Argument NOT used".

∗ **Call tree complexity.** Putting some pieces of code in a separate subroutine can reduce the size of the code in some cases (repetition of the same piece of code). There is currently a rather good level of call tree complexity in the code (at least for ARP/IFS and ALD) but it remains some very (too) long routines. Routines having more than 2000 lines (comments included) should be avoided.

There are also pieces of code where an excessive level of complexity can be noticed in the call tree, and that leads to lengthen the amount of code. For example routines just calling another one should be avoided (examples: EIGSOL which just calls RG). A list of routines generating a too high level of complexity in the call tree is provided in appendix C.

∗ **Reducing code duplication.** Additionally to the "D6-type" code duplication we spoke in a previous part, there are routines doing nearly the same thing. We should recall that one rule of the project ARPEGE is that, for a given application, there is only one routine doing it.

∗ **Removing comments which do not provide any relevant information.**

- There may be too detailed comments (sometimes not properly updated) in some routines, which can be shortened.
- Headers of empty paragraphs must be removed (this is the case in several "OBS_PREPROC" routines).
- In part referencing the list of modifications brought to the routines, it is desirable to keep no more than 10 years information. Information about modifications done before 2001 has been removed in CY37T1.

∗ **Excessive splitting of variables declarations among modules.** There are too many modules declaring variables and some of them only declare one or two variables. In appendix F we make a list of modules which can be gathered with other ones.

9.2 Examples of pieces of code which can be shortened or where duplication could be reduced.

∗ **Reduction of the number of dummy arguments in the grid-point calculations under GP_MODEL.** This is the main part of the code where there remains a lot of routines with a very long list of dummy arguments (and this part of the code is frequently modified). Significant improvements have been done in the adiabatic part of the code. Some work remains to be done in the physics-dynamics interface and in the physics.

∗ **Reduction of the number of dummy arguments elsewhere.**

- SUFPDYN and UPDVPOS: variables of module YOM4FPOS could be gathered in a new structure.
- SUVFPOS: variables of module YOMFPDIM could be gathered in a new structure.

∗ **Sets of routines doing roughly the same thing.** Some work has been done in CY37 to merge routines doing roughly the same thing. But some work remains to do.

- First I have noticed several routines doing the inversion of a set of tridiagonal linear systems, without pre-computing a LU decomposition in the set-up (tridia.F90, sgtsl.F, which are now in the new project XLA, and mse/internals/tridiag_ground_1d.mnh). Additionally to that, there are also inversions by (time-dependent) tridiagonal linear systems in some other parts of the code, for example in arp/phys_ec/cubidiag.F90 (which has TL and AD codes).

  Duplication of code must be reduced.
- In directory "arp/pp_obs":
  - PPRH does the same thing as GPRH+PPQ: same remark valid for TL and AD codes.

∗ **Rationalisation in the physics packages.** Even if some convergence has been tried in order to reduce the number of physical parameterizations (radiation for example), we notice that there are currently six sets of physical parameterizations:

- METEO-FRANCE physical parameterizations.
- AROME physical parameterizations.
- ALARO physical parameterizations.
- ARPEGE/CLIMAT physical parameterizations.
- ECMWF physical parameterizations.
- HIRLAM physical parameterizations (significant convergence with METEO-FRANCE physics has been done in CY38T1).

Some packages of simplified parameterizations must be added to this list.

The convergence between physical parameterizations should go further and the number of routines doing physical parameterizations should more decrease, and this question must remain steered by people working on physical parameterizations.

∗ **Rationalisation in the horizontal interpolators.** There are several horizontal interpolators we can sum-up as follows:

- The interpolator used in the semi-Lagrangian scheme (LAI... routines), also used in the observation horizontal interpolator under SLINT.
- The FULL-POS horizontal interpolator (FPINT... routines).
- The horizontal interpolator used in some C9xx configurations (arp/c9xx/INTER... and ald/c9xx/EINTER... routines).
- The horizontal interpolator used in the ECMWF physics to pass from refined to coarse grid (or the reverse operation) when using coarse resolution physics, in particuliar in the radiation scheme.
- The horizontal interpolator used to go from the high resolution trajectory towards the low resolution increments in the ECMWF 4D-VAR (routines arp/utility/GRID_...).
- The horizontal interpolator (bilinear interpolations) in SUHIFCE (under INTERP_OBS) used in the observation pre-processing at ECMWF.

Easily externalisable interpolators (and halo management) must be externalised; other interpolators must be rewritten in order to use externalised routines.

∗ **Priorities to reduce duplicata and to shorten the size of the code:**

- **(P0) Finish to replace calls to GPPRE or GPPREH+GPXYB+GPPREF by GPHPRE (see appendix K).**
- **(P0) Gather modules containing few variables: at least the following actions must be done:**
  - **Content of module YOMGPSK must become local variables in PREGPFPOS.**
  - **Content of PARMCUF must go in YOMMCUF.**
  - **Content of YOMWM must go in YOMPPC.**
- **(P0) Reduce calltree complexity: actions listed in appendix C1.**
- (P1) SUFPDYN and UPDVPOS: gather variables of module YOM4FPOS in a new structure.
- (P1) SUVFPOS: gather variables of module YOMFPDIM in a new structure.
- (P1) Reduce calltree complexity: actions listed in appendix C2.
- (P1+P2) Reduce the number of dummy arguments in the dynamics (remaining actions).
- (P0+P1+P2) Eliminate useless dummy arguments.
- (P0+P1+P2) Reduce the number of dummy arguments in the physics (remaining actions, in particuliar output fluxes).
- (P1+P2) Rationalisation in the physics packages.
- (P1+P2) Gather modules containing few variables: remaining actions described in appendix F.

10 Ensure consistency of variables names.

We now use several hundreads of meteorological quantities and it is desirable to:

- use the same root for naming the same variable throughout the code (for example use root OROG for surface orography, LSM for land-sea mask).
- not use the same root for two different quantities (avoid to use "R" for moist air constant in some parts of the code and for prognostic rain in some other parts of the code).

Documentation (IDRVN) gives recommended roots. It would be useful to use these recommended roots at least in global variables and attributes of global variables:

- GRIB codes (YOM_GRIB_CODES).
- Post-processing identifiers in module YOMAFN.
- ARPEGE file descriptors in module YOMFA.
- GFL, GMV, GMVS and surface fields attributes (modules YOM_YGFL, TYPE_GMVS, SURFACE_FIELDS_MIX, YOMSP, YOMSPJB, YOMSP_PTRS).
- GOMS attributes in module GOM_MOD.
- Code under CPG (in particuliar CPG_GP).
- Standard atmosphere variables in module YOMSTA.
- Some attributes in xla/module/spectral_fields_data.F90 (for SPJB).
- Some DDH identifiers in YOMMDDH.
- Some pointers in PTRGFU and PTRXFU.
- Some NVAR_... variables in YOMCOSJO.
- Some S... variables in YOMCLI.
- Miscellaneous variables (in YOMVODCST).

* Priorities to ensure consistency of variables names.

- (P2) Change names in order to use standard roots in global variables and attributes of global variables.
- (P2) Change names in local variables or dummy argument variables to use standard roots (example for land-sea mask: root ITM or SLM to be changed into LSM).

11 Ensure consistency between the AD and TL codes and their direct code counterpart.

For some pieces of code, the AD and TL code still matches with old versions of the direct code. The comprehensive list of mis-matchings is not provided.

12 Make the code norm-compliant.

See (IDCRTN) for details about these norms.

Most of the norm violations (in projects ARP/IFS and ALD only) are listed in a compilation by "gmkpack".

From now on it is desirable to remove the easiest norm violations during pre-merge for main cycles, in particuliar:

- remove declarations of unused variables (modules, local variables).
- rename non-DOCTOR compliant variables.
- remove tabulations.
- replace occurrences of obsolescent relational operators (like .EQ., .GE.) by modern ones.
- replace occurrences of END DO, END IF, END WHERE, ELSE IF by ENDDO, ENDIF, ENDWHERE, ELSEIF.
- correct missing & for continuation lines.
- break lines with more than 132 characters.
- replace PRINT * statements by WRITE(NULOUT,*) ones.
- correct arguments passed to DR_HOOK when easy to do.
- use CHARACTER(LEN=..) for declaring character variables.
- add INTENT statement when missing for dummy arguments (excepted for some pointers).
- label optional dummy arguments in callers, if not labelled.

∗ **Priorities to reduce norms violations.**

- **(P0) Keep the number of violations under 2400 (new standards) in CY41.**
- (P0+P1+P2) Reduce the number of GOTO and FORMAT statements.

13 References:

- (IDCRT) El Khatib, R., 2003: Coding standards for ARPEGE/IFS/ALADIN. Internal note, 42pp.
- (IDCRTN) Rivière, 0., 2011: Coding standards for ARPEGE/IFS/ALADIN systems. Internal note, 9pp.
- (IDARC) Yessad, K., 2013: Library architecture and history of the technical aspects in ARPEGE/IFS, ALADIN and AROME in the cycle 40 of ARPEGE/IFS. Internal note.
- (IDFPOS) Yessad, K., 2013: FULL-POS in the cycle 40 of ARPEGE/IFS (internal note).
- (IDKEYW) Yessad, K., 2013: Keywords in the cycle 40 of ARPEGE/IFS. Internal note.
- (IDOOPS) Yessad, K., 2013: OOPS: action cleaning (version v7). Internal note.
- (IDRVN) Yessad, K., 2013: Recommended variable naming in ARPEGE/IFS. Internal note.
- Yessad, K., 2012: Status of different interpolators coded in ARPEGE/IFS: towards an unification and externalisation (version v7). Internal note.

Appendix A: list of insufficiently commented declaration modules.

The following declaration modules in project “ARP/IFS” are insufficiently commented, sometimes not commented at all:

```
* grg_photolysis_mix.F90
* gridpoint_buffers_mix.F90
* iostream_mix.F90
* qacost.F90
* spectral_columns_mix.F90
* yoeaermap.F90
* yoeaersrc.F90
* yoeneur.F90
* yomfger.F90
* yomgetmini.F90
* yomglobs.F90
* yomjg.F90
* yomjq.F90
* yomlcz.F90
* yommts.F90
* yompaddh.F90
* yomrain_lb.F90
* yom_ssmi.F90
```

Appendix B: list of routines without any general comment describing action.

The following projects have been examinated: ARP/IFS, ALD, BIP, XRD/IFSAUX, XLA/ALGOR, TFL, TAL.

∗ Project ARP/IFS: ECMWF physics (principally radiation).

- arp/phys_ec/aer_phy1.F90 (insufficient commenting)
- arp/phys_ec/cuadjtq.F90 (insufficient commenting)
- arp/phys_ec/gwprofil.F90 (insufficient commenting: Stress profile?)
- arp/phys_ec/gwsetup.F90 (insufficient commenting)
- arp/phys_ec/gwsetupad.F90 (insufficient commenting)
- arp/phys_ec/gwsetuptl.F90 (insufficient commenting)
- insufficient commenting is also noticed in a significant amount of arp/phys_radi routines, for example in some rrtm.. and srtm.. routines.

∗ Project ARP/IFS: other topics.

```
* arp/canari: canari_sx_ics.F90.
* arp/dia: wrdistio.F90, wrgridall.F90, wrspeca_compress.F90, wrspeca_compress_mt.F90.
* arp/obs_preproc: extrap.F90, extraptl.F90, extrapad.F90.
* arp/phys_dmn: mts_phys.F90.
* arp/pp_obs: heapsort.F90.
* arp/utility: findminmaxg.F90.
* arp/var: add_modbias_ad.F90, add_modbias_tl.F90.
* arp/var: chavar....F90 (insufficient commenting)
* arp/var: convddr.F90, evjcdfi.F90, get_jbvcoord_coeffs.F90, supavarc.F90, suprepjcdfi.F90.
```

∗ Project XRD/IFSAUX.

- cma/oldcma_get_address.F (maybe useless?)
- cma/oldcma_set_address.F (maybe useless?)
- most of “eclite” files.
- grib_io/grib_set.F
- grib_mf/gsbyte_mf.F
- most of “misc” files.
- parallel/cmpl_binding.F90
- parallel/coml_binding.F90
- programs/datefa.F
- programs/gribdiff.F
- programs/gribtrace.F
- most of “support” files.
- most of “utilities” files.

∗ Project XLA/ALGOR.

- most of “internal/lanczos” routines.
- internal/linalg/cdiv.F
- internal/linalg/mxva.F
- internal/linalg/sgemmx.F (insufficient commenting)
- internal/minim/ctcab_1dv.F
- internal/minim/ctonb_1dv.F
- internal/minim/dpseuclid.F90
- internal/minim/ecube_1dv.F
- internal/minim/euclid_1dv.F
- internal/minim/euclid.F
- internal/minim/m1qn3a....F routines (insufficient commenting)

∗ **Project TFL.**

- module/abort_trans_mod.F90
- module/fspgl_int_mod.F90
- module/set_resol_mod.F90
- module/setup_dims_mod.F90
- module/setup_geom_mod.F90
- module/spnormc_mod.F90
- module/spnorm_ctl_mod.F90
- module/spnormd_mod.F90
- module/sufft_mod.F90
- programs/aatestprog.F90
- programs/test_adjoint.F90

∗ **Project TAL.**

- module/cpl_int_mod.F90
- module/eset_resol_mod.F90
- module/esetup_dims_mod.F90
- module/esetup_geom_mod.F90
- module/espnormc_mod.F90
- module/espnorm_ctl_mod.F90
- module/espnormd_mod.F90
- module/suefft_mod.F90
- programs/aatestprog.F90
- programs/test_adjoint.F90

Appendix C: excessive complexity in call tree.

Some routines just call another one (with one or two additional instructions), and they can be in-lined then removed. More generally, we point out excessive (and useless) complexity in the call tree which makes the code more difficult to read.

C1: project ARP/IFS: ECMWF observation pre-processing (directory obs_preproc).

Call tree of ECMWF observation pre-processing can be simplified.

- AIREPBE, DRIBUBE, PAOBBE, RAD1CBE, REO3BE, SATOBBE just call FILFBDE: directly call FILFBDE in callers (i.e AIREPIN, DRIBUIN, PAOBIN, GEOSRIN, RAD1CIN, REO3SIN, SATAMIN, SATOBIN).
- PILOTBE, SCATBE, SYNOPBE, TEMPBE just call FILFBDE + one additional memory transfer: inline these routines in callers (i.e. AWPRFIN, DWLIN, EWPRFIN, PILOTIN, NSCATIN, LNDSYIN, METARIN, PGPSIN, SHIPIN, TEMPIN).

C2: project ARP/IFS: other possible actions.

- arp/dia/spnormbm.F90 just calls SPNORMB+ESPNORMB (+ some calls to GSTATS): in-line the content of SPNORMBM in its callers.
- The content of SUMETRIC and SUEMETRIC can be in-lined in SPNORM.
- arp/phys_dmn/ecr1d.F90 just calls LFAECRR: call LFAECRR instead of ECR1D.
- arp/phys_dmn/ecr2df.F90 just calls LFAECRR: call LFAECRR instead of ECR2DF.
- arp/phys_dmn/ecr2dv.F90 just calls LFAECRR: call LFAECRR instead of ECR2DV.
- arp/phys_dmn/ecrpnebh.F90 just calls LFAECRR: call LFAECRR instead of ECRPNEBH.
- arp/dia/wrmlpp.F90 just calls WRMLPPA or WRMLPPG: the content of WRMLPP can be in-lined in its callers.
- The content of RDRINC (and maybe also WRRINC) can be in-lined in IOPACK.
- The content of RD801 can be in-lined in IOPACK.
- The content of SUCMAD1P can be in-lined in SUCMAD1.

Appendix D: inconsistencies between namelist naming and module naming.

Some of these renamings can be done during the OOPS reorganisation of setup and variables.

D1: List of inconsistencies.

∗ NAC.., NAD.., NAI.. and NAL.. namelists (CANARI).

- NACOBS: some variables are in QAPABO, not in QACOBS.

∗ NAE.. namelists (ECMWF physics).

- NAEAER: variables are spread among YOEAERSRC, YOEAERATM, YOEDBUG, YOEAERMAP.
- NAEPHY: some variables are spread among YOEWCOU, YOMCOAPHY, not in YOEPHY.
- NAERAD: some variables are spread in YOMPRAD, YOERDI, YOE_UVRAD.

∗ NAM.. namelists.

- NAM_CANAPE: variables are in QAREF.
- NAMCT0 variables are spread among several modules.
- NAMDDH: logical variables are in YOMLDDH; integer variables are in YOMMDDH.
- NAMDYN: some variables are not in YOMDYN (they may be for example in YOMMASS if mass corrector, YOMTRAJ for high resolution trajectory, YOMSEP, etc...)
- NAMFPSC2 and NAMFPSC2_DEP variables are in YOMFPSC2.
- NAMGFL: variables are in YOM_YGFL and not in YOMGFL.
- NAMJBCODES: variables are in YOMJG.
- NAMJG: some variables are not in YOMJG (they are in YOMWAVELET or YOMCOSJB).
- NAMNPROF: variables are in YOMECTAB.
- NAMPAR0: variables are spread among YOMGSTATS, YOMCT0, YOMMP0.
- NAMPAR1: variables are spread among YOMIO and YOMMP0.
- NAMRCF: variables are spread among YOMRES and YOMMASS.
- NAMSCEN: RI0 is in YOMCST, not in YOMSCEN.
- NAMTRAJP: variables are spread among YOPHNC, YOMIOP, YOEPHLI, YOMNCL.
- NAMVAR: some variables are spread among YOMTRAJ, YOMJQ, YOMVCGL.

∗ NEM.. namelists (LAM models).

- NEMVAR: variables are in YEMVARGP.

∗ Other inconsistencies.

- nammatchup.h (ODB) contains a namelist called NAM_MATCHUP.
- namsort.h (ODB) contains a namelist called NAM_SORT.
- namstdin.h (ODB) contains a namelist called NAM_STDIN.
- namwt.h (ODB) contains a namelist called NAM_WT.

D2: A first set of actions to be done.

- **rename YEMVARGP into YEMVAR and SUEVARGP into SUEVAR (or rename NEMVAR into NEMVARGP).**
- **rename NAM_CANAPE into NACREF, QAREF into QACREF (CANARI).**
- **rename NAMGFL into NAM_YGFL.**

and also:

- **ODB namelist: rename deck nammatchup.h into nam_matchup.h.**
- **ODB namelist: rename deck namsort.h into nam_sort.h.**
- **ODB namelist: rename deck namstdin.h into nam_stdin.h.**
- **ODB namelist: rename deck namwt.h into nam_wt.h.**

D3: A second set of actions to be done.

- YOMTRAJ variables of NAMVAR must be moved in a new namelist element NAMTRAJ.
- Some other actions not detailed for the time being (some of them can be done during the OOPS setup reorganisation).

Appendix E: Decks to be moved or renamed.

This appendix does not take account of the namelist or module renamings listed in Appendix D to ensure name consistency between modules and namelists. It ignores the still open question of appendix of modules (appendix "mix" or "mod" will probably become useless since all modules may possibly become mixed modules in the future, with type definition or encapsulated routines).

E1: Routines to be moved without renaming into existing directories.

Routine	Current dir	New proposal of dir
ecoptra.F90	ald/c9xx	ald/var
elislap.F90	ald/c9xx	ald/adiab
sufpmove.F90	ald/fullpos	arp/fullpos
sueframe.F90	arp/setup	ald/setup
coptra.F90	arp/c9xx	arp/var
lislap.F90	arp/c9xx	arp/adiab
cpnudg.F90	arp/dia	arp/climate
pos.F90	arp/pp_obs	arp/fullpos
pos_prepgfl.F90	arp/pp_obs	arp/fullpos
troplev.F90	arp/var	arp/pp_obs
eigenmd.F90	arp/var	arp/utility
rd801.F90	arp/var	arp/setup
gp_sstaqua.F90	arp/phys_ec	arp/setup
iopack.F90	arp/utility	arp/control
matrixin.F90	arp/utility	xla/external/linalg
orthnorm.F90	arp/var	xla/external/linalg
control_vectors_comm_mod.F90	arp/module	xrd/module
indexfind_mod.F90	arp/module	xrd/module
tracare.F90	arp/transform	xrd/utilities
trareca.F90	arp/transform	xrd/utilities
xutrii.F90	arp/utility	xrd/utilities

There are also radiation routines which are still in directories arp/phys_dmn or arp/phys_ec; they must probably be moved in directory arp/phys_radi.

E2: Routines to be renamed without moving.

Current name	New proposal of name
ald/adiab/gpspng.F90	ald/adiab/gp_spng.F90 (g.p. calc., not GPS)
ald/var/erdlsgrad.F90	ald/var/erdlsgr.F90 (gradient, not adjoint)
ald/var/ewrlsgrad.F90	ald/var/ewrlsgr.F90 (gradient, not adjoint)
ald/var/suejbdat96.F90	ald/var/suejbdat.F90
arp/adiab/gpstress.F90	arp/adiab/gp_stress.F90 (g.p. calc., not GPS)
arp/dia/suechk.F90	arp/dia/suchk.F90
arp/module/yemchk.F90	arp/module/yomchk.F90
arp/module/yomcumfs.F90	arp/module/yoecumfs.F90 (ECMWF physics)
arp/module/yomfpop.F90	arp/module/yomfpoph.F90
arp/obs_preproc/flspeedbad.F90	arp/obs_preproc/flspeedwrong.F90 (not AD)
arp/pp_obs/ppinitza.F90	arp/pp_obs/ppinitzad.F90 (adjoint code)
arp/pp_obs/ppobsaza.F90	arp/pp_obs/ppobsazad.F90 (adjoint code)
arp/setup/sujbvolc.F90	arp/setup/suvolcano.F90
arp/var/surad.F90	arp/var/susats.F90 (not AD)
sct/qretrieve/minima.F	sct/qretrieve/minima_sct.F

E3: Routines to be renamed and moved.

Current directory/name	New proposal of directory/name
arp/climate/cormass3a.F90	ald/climate/ecormass3a.F90
arp/climate/cormass3b.F90	ald/climate/ecormass3b.F90
arp/setup/sugpqlim.F90	arp/adiab/gpqlim.F90
arp/var/suspqlim_part1.F90	arp/adiab/spqlim_part1.F90
arp/var/suspqlim_part2.F90	arp/adiab/spqlim_part2.F90
ald/setup/sueqlimsat.F90	ald/adiab/espqlim.F90
arp/var/qneglim.F90	arp/adiab/gp_qneglim.F90
arp/var/qneglimtl.F90	arp/adiab/gp_qneglimtl.F90
arp/adiab/larche_hlp.F90	xrd/utilities/???.F90

Appendix F: Modules which can be gathered with other ones.

F1: Modules containing variables which control some optimisation in the dynamics or the physics.

This work can be done for CY41, but it is not independent from the interpolator externalisation.

These modules contain a small amount of variables. Some of them are used to do optimisations in the interpolators (mostly adjoint code of interpolators) and may have to be externalised in the future new library INT or to have two versions between ARP/IFS and INT.

Variables remaining in ARP/IFS must be gathered into one module (OPTIM_MOD), with one encapsulated namelist (NAMOPTIM).

List of modules and variables is:

- YOMJFH: only contains N_VMASS.
- YOMPLDSW: only contains LOPT_SCALAR and LOPT_RS6K.
- YOMSEP: only contains LFINDVSEP, NVSEPC, NVSEPL.
- YOMSLREP: only contains LVECADIN, LSLADREP, NGPTOTAD, NADCORE, NADMAP, RSASIGN.

More details about work to be done, and potential interferences with the new project INT:

- First define LOPT_SCALAR and LOPT_RS6K (cf. SUMPINI).
- Define LVECADIN, according to LOPT_SCALAR and LSLAG (cf. SUCT0).
- Define LSLADREP, according to LOPT_SCALAR and LSLAG (cf. SUCT0, SUVAR).
- Define NGPTOTAD and NADCORE (cf. SUSLAD1, SUSLAD3).
- Define N_VMASS according to LOPT_RS6K (cf. SUJFH).
- Define LFINDVSEP according to NCONF and LSLAG (cf. CNT4).
- Define NVSEPC and NVSEPL according to NCONF, LSLAG and LFINDVSEP.
- Variables NADMAP and RSASIGN belong to the externalisable part of interpolators and must go in eint_mod.F90 (must go in the future library INT).

Additional remarks:

- LVECADIN and LSLADREP are currently used in some adjoint routines of interpolations, but they may become meaningful in some non externalisable parts too.
- NGPTOTAD and NADCORE are used in the adjoint code of the semi-Lagrangian scheme, but in non-externalisable parts only.
- LFINDVSEP, NVSEPC and NVSEPL: NVSEPC and NVSEPL are used to do optimisations in some adjoint routines of interpolations, but calculation of NVSEPC and NVSEPL is spread among several routines in some pieces of code which are not externalisable. It is better to keep calculation of these variables in ARP/IFS, and to pass NVSEPC or NVSEPL to adjoint routines of interpolations via dummy argument interface.

F2: Modules containing buffers involved in FULL-POS.

These modules contain a small amount of variables and they can be gathered into one module (YOMFPBUF).

- YOMDFPB: only contains HFPBUF and HFPBUF_DEP.
- YOMPFPB: only contains GFPBUF and GFPBUF_DEP.
- YOMRFPB: only contains RFPBUF.

Additionally:

- YOMGPSK: only contains GPSCFBUF, GPSXFBUF, only used in PREGPFPOS. These variables must be replaced by local variables in PREGPFPOS.
- YOMOMPDIST: the question is open about GPP.

F3: Modules for applications used at METEO-FRANCE.

- PARMCUF: only contains JPMFNR and JPMFOR. Can go in YOMMCUF.
- YOMGFUB: only contains GFUBUF. Can go in YOMCFU.
- YOMXFUB: only contains XFUBUF. Can go in YOMXFU.
- YOMWM: only contains FTHSOR. Can go in YOMPPC (used for ISP; goes with LMOVPH). Allocation in SUDYN is not at the right place (must probably go with allocation of GFUBUF and XFUBUF in part 3.4 of SUSC2B).
- YOMMPEXTRA (three variables). Can go in YOMMP (in this case the content of SUMPEXTRA must go in SUMP).
- YOMFPCT0: only contains NFPCT0 and FPINCR.
  - NFPCT0 can go in YOMCT0, with LFPART2 and NFPOS.
  - FPINCR can go in the same module as NFPINCR, i.e. YOMFPC.

F4: Various modules: more difficult tasks.

- YOMGPPCB: only contains GPPCBUF. Content of this buffer must be spread among GMV and GFL.
- Trajectory for NH model: YOMIOPNH (contains LTRAJNH and NG3NH95) and YOMTNH (NLENNH95B and TRAJNH). A more general thinking must be done about trajectory variables (in a OOPS frame) to know what to do (variables are currently spread among YOMTRAJ, YOMTRSL, YOMTNH, YOMSIMPHL, YOMIOP, YOMIOPNH, TRAJECTORY_MOD). Provisionally YOMIOPNH and YOMTNH can be gathered in a new module YOMTRNH.

Appendix G: Proposal of definition of new derivated type structures.

G1: Module variables and set-up.

About module variables computed in set-up routines, the priority must be given to sets of variables which appear together in subroutines calls, and generate long lists of dummy arguments.

∗ Global pointers:

- PTRSLB1 and PTRSLB2 pointers:
  - PTRSLB1 pointers (type TSLBUF1): the different pointers MSLB1[X] of PTRSLB1 can become attributes of a new variable YSLB1 declared with type TSLBUF1. NFLDSLB1 (currently in YOMSC2) can become an attribute of YSLB1. Additionally to that, new logical attributes LINT[X] can be added to YSLB1 to say if place must be allocated in SLBUF1 for variable [X] (equivalent to say that the interpolation of [X] is required): LINT[X] will be used everywhere in the code when requested (filling SLBUF1 in LATTEX/LATTES, doing interpolation in LARCINA/LARCINB).
  - PTRSLB2 pointers (type TSLBUF2): the different pointers MSLB2[X] of PTRSLB2 can become attributes of a new variable YSLB2 declared with type TSLBUF2. NFLDSLB2 (currently in YOMSC2) can become an attribute of YSLB2.
  - A new module slbuf_mod.F90 can be created to define types TSLBUF1 and TSLBUF2 and declare variables YSLB1 and YSLB2; additionally this module will contain the set-up routine SUSLB which is currently in the separate deck suslb.F90 (and maybe also SUSLB2 for the shallow-water model, but we can also keep the old dataflow for it provisionally).
- The question to extend this idea to the other horizontal interpolators (FULL-POS, radiation, coarse physics) must be studied.

G2: Local variables, intermediate calculations.

∗ Dimension variables.

- Input dimensions of most of GP.. routines (type TDIMGP): new type definition with the following attributes: IPROMA, IST, IEND, IFLEVG.

∗ Output of GP, GNH routines, or linear terms (for example computed in LANHSI): type definitions to be put in a new module named intdyn_mod.F90.

- Replace calls to GPPREH+GPXYB+GPPREF or GPPRE by GPHPRE.
- Output of GPRCP (type TRCP) in callers other than CPG, CPGTL, CPGAD: new type definition with the following attributes: CP, R, KAP. Enter gathered array in GPRCP, GPRCPTL and GPRCPAD.
- Output of GNHPRE, GNHGRPRE, GNHPREH (type TNHPRE): new type definition with the following attributes: at least NHPREF, NHPREH, RNHPPI, QCHAL, QCHAM.
- Output of GNHDLR and GNHGRDLR, for LRWSDLR=T (type TWSDLR): new type definition with the following attributes: WDLMUSF, WDLMUSH, WDLWF, attributes for the current content of RDLR.
- Output of GPGEO and GPGRGEO (type TGEO): new type definition with the following attributes: PHIH, PHIF, PHIHL, PHIHM, PHIFL, PHIFM.
- Output of LASSIE, LANHSI = linear terms (type TLINT): new type definition with the following attributes: SDIV, TOD, GAGTL, GAGTM, SPDS, LPD, DPD.
- Output of GPRT (type TRT): new type definition with the following attributes: RT, RTL, RTM.

∗ Output of physics.

- All the output fluxes of the physics, can be gathered in new derivated types structures.

Appendix H: Useless routines.

H1: ECMWF physics routines (to be conserved for the time being).

```
* arp/phys_ec/aer_drydep_ad.F90
* arp/phys_ec/aer_drydep_tl.F90
* arp/phys_ec/aer_sedim.F90
* arp/phys_ec/cloudaer.F90
```

H2: Other routines in ARP/IFS.

```
* arp/function/fcgeneralized_gamma.h
```

H3: Other routines in TAL, XRD/IFSAUX.

```
* tal/module/tpmald_tcdis.F90
* xrd/utilities/eggdir.F90
* xla/external/linalg/sgtsl.F
```

Appendix I: routines having too many dummy arguments, or useless dummy arguments labelled with “Argument NOT used”.

I1: Routines having useless dummy arguments labelled with “Argument NOT used”.

There are also useless dummy arguments, not referenced below, which are not labelled with “Argument NOT used”.

```
----- ALD:
* ald/c9xx       : eleci.F90
* ald/dia        : espnormb.F90

----- ARP: other routines
* arp/adiab      : fspglh.F90
* arp/c9xx       : inter1.F90, inter2.F90, inter6.F90, inter8.F90, inter10.F90, lislap.F90
* arp/control    : sim4d.F90
* arp/obs_preproc: ifsodbddr2s.F90
* arp/sinvect    : scaas.F90
* arp/var        : ctcab.F90, ctonb.F90, prosca.F90
```

I2: Routines having more than 50 dummy arguments.

```
----- at least 100 dummy arguments:
* APLPAR has 281
* POSTPHY has 233
* WRITEPHYSIO has 205
* MF_PHYS has 191
* WRITEMUSC has 183
* CPG_DIA has 157
* VDFOUTER has 152
* CPEDIA has 148
* VDFMAIN has 147
* WRITEPROFILE has 132
* INITAPLPAR has 130
* APLPARSTL has 129
* PROFILECHET has 128
* CALLPARTL has 128
* APLPARSAD has 127
* CALLPARAD has 127
* VDFMAINSAD has 113
* VDFMAINSTL has 111
* CPTEND_NEW has 106
* APL_AROME has 105
* ACHMTTL has 103
----- between 90 and 99 arguments:
* UPDATE_FIELDS has 96
* CPPHDDH has 94
* CUCALLN2TL has 90
----- between 80 and 89 arguments:
* CUCALLN2AD has 83
* CPG_GP has 82
* CPTENDSMTL has 81
* CPTENDSMAD has 81
* ARP_GROUND_PARAM has 80
* APLPARSADT has 80
----- between 70 and 79 arguments:
* ACHMTAD has 78
* ACDIFUS has 78
* CUBASEN2AD has 74
* VDFDIFHSAD has 74
* APLPARS has 73
* LARCIN2TL has 73
* LARCIN2AD has 72
* ACPCMT has 72
* VDFDIFHSTL has 72
* CUMASTRN2AD has 71
* CUMASTRN2TL has 71
* CPCHET has 70
----- between 60 and 69 arguments:
* VDFMAINS has 69
* CPCFU has 68
* ACHMT has 67
* onedvar_raintb has 67
* RADPAR has 65
* LAINOR2TL has 64
* LWBVAD has 63
* LAINOR2AD has 62
* CUCALLN has 63
* AER_PHY2 has 61
* CLOUDSC has 60
* AER_PHY3 has 60
----- between 51 and 59 arguments:
* PREINTSTL has 59
* AER_SRC has 59
* CUCALLN2 has 58
* CUBASEN2 has 58
* ACDIFSPAD has 58
* ACDIFSPTL has 58
* CPTEND has 58
* LWVAD has 57
* CUFLX2TL has 57
* CUDTDQN2AD has 57
* CUASCN2AD has 57
* CUFLX2AD has 57
* CUDTDQN2TL has 57
* CUASCN2TL has 57
* CPXFU has 57
* CPDYDDH has 57
* LARMES2AD has 56
* ACMTUD has 56
* CPPHDDHE has 55
* LASCAWTL has 55
* PREINTSAD has 55
* ACDIFV3 has 55
* CPG_GP_AD has 54
* LACDYNSHWTL has 54
* LARMES2TL has 54
* CPG_GP_TL has 54
* CPG has 54
* LASCAW has 54
* ACCVUD has 54
* RADLSWR has 54
* VDFDIFH has 54
* RADTR_ML_TL has 53
* RADINATL has 53
* RADLSWAD has 53
* RADLSWTL has 53
* RADTR_ML_AD has 52
* ACNTCLSTL has 52
* MF_PHYSAD has 52
* MF_PHYSTL has 52
* LWBV has 52
* ACCVIMPGY has 51
* RESTORE_SURFTSTP has 51
* RADLSW has 51
* WRPHTRAJTM has 51
* RDPHTRAJTM has 51
```

Appendix J: routines having at least 10 norm violations.

Not detailed for CY40, but most of norm violations “easy to remove” have been removed.

Appendix K: routines where activation of GPHPRE remains to be done.

Markers have been put how to replace GPPRE or GPPREH+GPXYB+GPPREF by new GPPRE in the following routines (ARP/IFS only):

```
* adiab/gpstress.F90          (ECMWF)
* chem/chem_massdia.F90       (ECMWF)
* climate/cormassdry.F90      (ECMWF)
* control/tracmf.F90          (ECMWF)
* fullpos/specfitg.F90        (ECMWF)
* obs_preproc/gefger.F90      (ECMWF)
* phys_ec/aer_clim.F90        (ECMWF)
* phys_ec/aer_climg.F90       (ECMWF)
* phys_ec/aer_climz.F90       (ECMWF)
* phys_ec/aer_clist.F90       (ECMWF)
* phys_ec/aer_stratcl.F90     (ECMWF)
* phys_ec/ec_phys_ad.F90      (ECMWF, partly done in CY39)
* phys_ec/ec_phys.F90         (ECMWF, partly done in CY39)
* phys_ec/ec_phys_tl.F90      (ECMWF, partly done in CY39)
* phys_ec/heldsuarez.F90      (ECMWF)
* phys_ec/radcfg.F90          (ECMWF)
* phys_ec/suphec.F90          (ECMWF)
* prism/couplo4_grg_input.F90 (ECMWF)
* prism/couplo4_grg_stats.F90 (ECMWF)
* setup/suspecg2.F90          (ECMWF)
* var/estsig.F90              (ECMWF)
* var/pregprh.F90             (ECMWF)
* var/suinfce.F90             (ECMWF)
```