OOPS observation cleaning

Scope: the Fortran code of the working IFS

Alan Geer, Peter Lean, Deborah Salmond, Thibaut Montmerle, Christophe Payan

Overview

- Principles:
  - Keep IFS and OOPS observation code as similar as possible (ideally identical)
  - Isolate the observation world from the model world
  - Make it easier to work with existing obs and to add new ones
  - All changes will be bit-reproducible and existing science capabilities will be preserved
- Organisation of work:
  - VarBC, direct code, test harness, radiance operators: ECMWF
  - TL and AD, MF-specific aspects like APACHE: Meteo France

Main developments

- GOM_PLUS is a new module/derived type for model quantities in observation space which:
  - encapsulates all model-related code currently called from the observation operator (e.g. gphpre)
  - makes it easy to access model fields in the code and pass from one subroutine to another.
  - will be the input to the OOPS observation operator (not the GOM directly)
  - one per observation set
- Hop/tl/ad, hretr – major cleaning, with most observation operator code being moved down into subroutines.
- Test harness to call observation operator and generate FG departures and TL/AD testing without calling any model code or initialising any model modules.

Rules

- Good coding (not always followed in the current IFS)
  - Subroutines (the smaller the better)
  - No code duplication
  - Structured coding
  - Clearly defined inputs and outputs (avoid input through modules)
- No accessing model variables from observation operator
  - All necessary information should come through gom_plus
  - Gom_plus is INTENT(IN) in TL and direct code, clearly indicating that model data is an input to an observation operator
- No accessing the ODB from the model or data assimilation
  - ODB is internal to the observation code
- No accessing DA state information (e.g. lifstraj in yomct0)
  - We still need a few switches like LSCREEN, to be passed in by argument

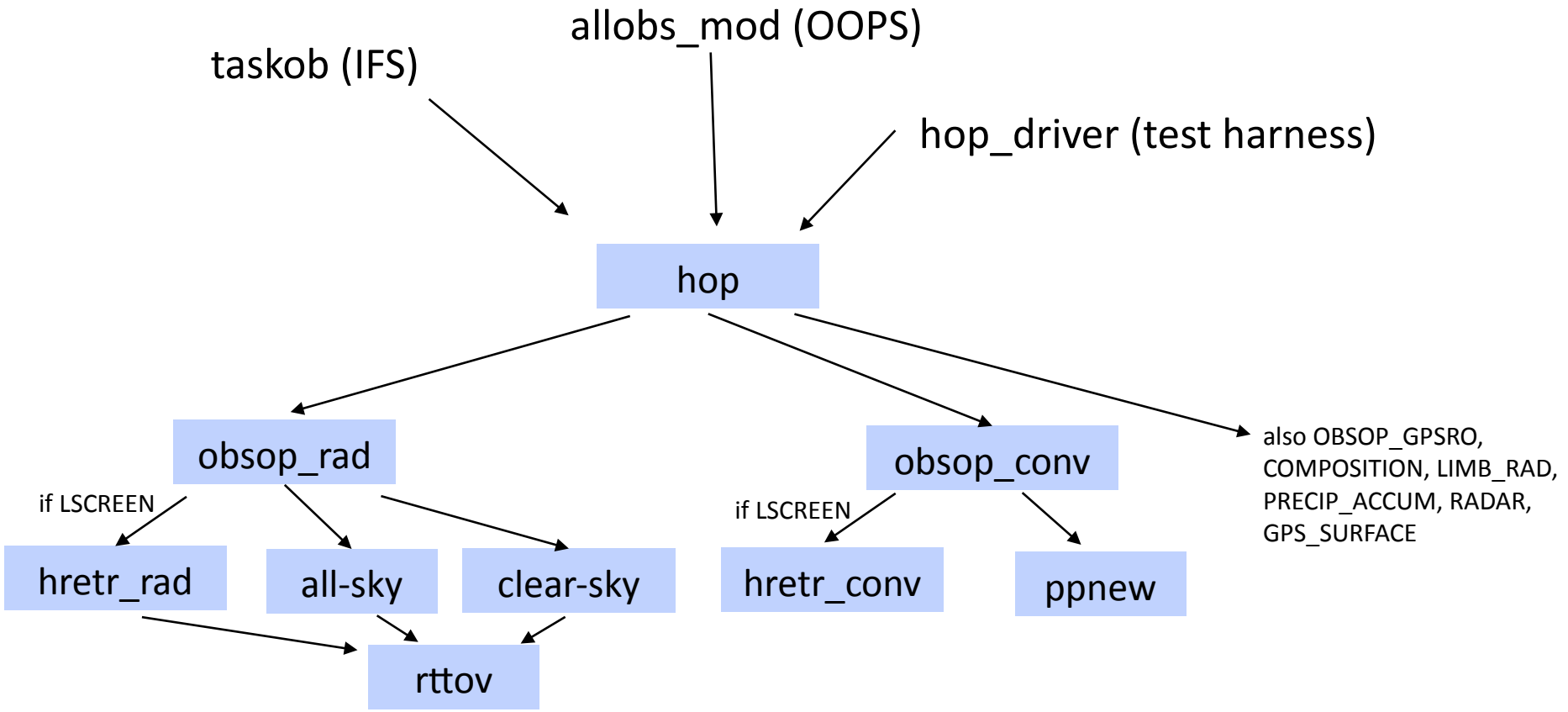
Low-level cleaning

- Access to global module data removed
- kdlen and ilen dual-dimension removed (unwanted optimisation for ancient supercomputer architectures)
- RTTOV level code removed
- 0:NFLEVG false vertical dimension removed
- Preint* routines, long argument lists for model fields (e.g. PTF, PQF, PO3F, PXPP…), and gems_profs have been removed and replaced by gom_plus
- NVNUMB(X) loop and ZXPP removed
  - Operators are now selected by varno (e.g. varno%bend_angle triggers GPSRO operator)
- VarBC code is encapsulated in fortran “object”
- Observation sets are encapsulated in fortran object

Cleaning

- Hop is now a “pure” observation operator
  - hretr (screening and one-off processing) is now called under the relevant obsop, under the switch LSCREEN=true (e.g. obsop_rad calls hretr_rad)
  - departure_jo takes non-obsop things (e.g. departure computation, FSO) and separates them from hop
  - conventional operator dependence on globals (GOM, model geometry) has been removed with help of “ppnew.F90” which replaces ppobsa, ppobsas etc.

New structure



Single routine covers direct, TL and AD cases

Direct/TL/AD polymorphism with F95

Optional gom_plus arguments. If PRESENT they indicate TL or AD behaviour

SUBROUTINE HOP(YDGP5, YDVARBC, YDSET, PHOFX, PBIAS, YDGP_TL, YDGP_AD)

CALL OBSOP_CONV(ROBHDR, ROBODY, SATHDR, SATBODY, ROBSU, YDGP5,
VARNOS_TO_PROCESS, LSCREEN, LLFINAL_TRAJ, YDSET, PHOFX, YDVARBC,
YLVARBC_EXTRA_PRED, YDGP_TL=YDGP_TL,YDGP_AD=YDGP_AD)

CALL PPNEW(YDGP5, ZVPOBS, ICOUNT, ZXPP, CDPAR=CLV, YDGP_TL=YDGP_TL,
YDGP_AD=YDGP_AD)

Direct (usually always)

IF (LLTL)

IF(LLAD)

Separate TL and AD routines are still sometimes necessary at the very lowest levels

Summary at 42r3

- Once in 20 year cleaning of the observation operator
  - About 15,000 poorly structured lines of code replaced by 9,000 refactored lines (excluding extensive low-level operators like RTTOV).
  - 2 person-years effort from ECMWF and Meteo-France
- Final tidying
  - Some remaining access to forbidden global variables?
  - Some minor bugs are possible in difficult-to-test areas of code (e.g. composition, BG error randomisation)
- Support for OOPS C++ layer
  - We now have a clean observation operator suited to the IFS and that will be operationally used in IFS and MF CY43
  - There may still be some refactoring needed to make it support the way the OOPS C++ works, e.g. accumulation of GOMs.
    - But this would have been impossible before cleaning!