Arpège/IFS in OOPS Framework

- 3D-Var demonstration of IFS in OOPS
- OOPS Framework calls IFS Fortran
- State for IFS
- Observation Operator
- \( J_B \) modularisation
- Cleaning and reorganisation for Cycle 38
- New Scalability opportunities
- IFS Fortran plugs into OOPS

Yannick Trémolet, Mike Fisher, Anne Fouilloux, John Hague, Tomas Wilhelmsson, George Mozdzynski, Mats Hamrud, Karim Yessad, Jean-Noël Thépaut and Deborah Salmond
Warning ...
Work-in-Progress

Klein Matterhorn 3883 m.ü.m
3D-Var demonstration of IFS in OOPS

- 3D-Var will be the first milestone of the OOPS project to demonstrate how parts of the IFS can be extracted and plugged in to the OOPS control layer

- Model is not needed for 3D-Var – need State/Fieldset

- Work with One Observation type (AMSU-A) with vertical interpolation in RTTOV

- Most of the work in the Fortran code is to be included in (and improve) the Arpege/IFS cycles

- Minimise changes to low level IFS code
Proposed OOPS-Framework

C++ control layer

- State & Fieldset
- Observations & ODB
- Covariance Matrices
- Interpolation to Obs points
- Observation Equivalents
OOPS-IFS

Fortran Main
MPL_INIT
Set constants

OOPS control layer - replaces CNT0 etc
Definition of Algorithm

GMV, GFL, surface_fields

Observations & ODB

COBSALL

CVAR3IN

HOP
IFS Fieldset

Derived Types for IFS Fieldset:

```plaintext
type :: ifs_fields
    type(ifs_geometry_type) :: geom
    type(type_yomgmv) :: gmv
    type(type_yomgfl) :: gfl
    type(type_yom_ygfl) :: ygfl
    type(type_surface_fields) :: surf
end type ifs_fields

TYPE :: TYPE_YOMGFL
PRIVATE
REAL(KIND=JPRB),POINTER :: GFL (:,:,:,:) => NULL()
REAL(KIND=JPRB),POINTER :: GFLT1 (:,:,:,:) => NULL()
REAL(KIND=JPRB),POINTER :: GFLSLP (:,:,:,:) => NULL()
REAL(KIND=JPRB),POINTER :: GFL5 (:,:,:,:) => NULL()
REAL(KIND=JPRB),POINTER :: GFL_DEPART (:,:,:,:) => NULL()
REAL(KIND=JPRB),POINTER :: GFLPT (:,:,:,:) => NULL()
REAL(KIND=JPRB),POINTER :: GFLPC (:,:,:,:) => NULL()
END TYPE TYPE_YOMGFL

etc.
```
IFS Geometry object

List of IFS Modules with geometry information:
- yomcver: Variables for vertical finite elements
- yomdim: Dimensions of model working arrays
- yomgc: Grid point boundaries
- yomgem: NGPTOT, transformed sphere, vertical coords
- yomlap: Constants related to the Laplace space
- yomleg: Description of Legendre polynomials
- yommp: Distributed memory parallelization

Or from Karim’s OOPS Action Cleaning document:
- hdim_mod: Horizontal dimension variables
- vdim_mod: Vertical dimension variables
- Etc.
Multiple instances of data in IFS MODULES

MODULE YOMGFL
SAVE
REAL, POINTER :: GFL(:,:,:,:,:), NSCALAR, LFLAG
TYPE :: TYPE_YOMGFL
PRIVATE
REAL, POINTER :: GFL(:,:,:,:,:), NSCALAR, LFLAG
END TYPE TYPE_YOMGFL
END MODULE YOMGFL

CONTAINS

SUBROUTINE SET_YOMGFL(T)
  TYPE(TYPE_YOMGFL), INTENT(IN) :: T
  T%GFL => GFL
  NSCALAR = T%NSCALAR
  LFLAG = T%LFLAG
END SUBROUTINE SET_YOMGFL

SUBROUTINE SAVE_YOMGFL(T)
  TYPE(TYPE_YOMGFL), INTENT(OUT) :: T
  T%GFL => GFL
  NSCALAR = T%NSCALAR
  LFLAG = T%LFLAG
END SUBROUTINE SAVE_YOMGFL

* Saves much re-coding of IFS where global data is USE-ed from MODULES

Original

MODULE YOMGFL
SAVE
REAL, ALLOCATABLE :: GFL(:,:,:,:,:), NSCALAR, LFLAG
END MODULE YOMGFL

ALADIN / HIRLAM - 07-04-11
Encapsulate $H(x)$ in IFS

- **Horizontal Interpolation to OBSERVATION Locations**
  - Based on Karim’s reorganised version @ 37
  - COBSALL (calls COBS and COBSLAG)
  - What modules are input and where they are set?
  - Made simple driver to call COBSALL

- **New Derived Types for GOMS and GLOBS**
  - are passed through calling tree

- **Calculate OBSERVATION EQUIVALENTS**
  - HOP - (without HDEPART & HJO) → RTTOV
  - Split HOP into routines for different Obs Types
  - What modules are input and where they are set?
  - Made simple driver to call HOP
Cleaning and reorganising of HOP

- 94 subroutine calls directly from HOP and 231 IF tests

- In hop.F90 we have:

```fortran
CALL GETDB('HOP', NUPTRA, IRET, INFO, 0, ZINFO, 0, &
         & KSET, ITSLOT, -1, IOBSTYPE, ICDTP_TOVS, ISENSOR_GETDB)

IF (ICDTYP_TOVS == NSTRO3) then
  ...
elseif (ICDTYP_TOVS == NGTHRBR) then
  ...

- But in getdb we also have the same IF statements...

if (cdretr == 'HOP') then
  if (ICDTYP == NSTRO3) then
    llexecv(9) = .TRUE.     ! For get_soe_reo3.sql request
  elseif (ICDTYP == NGTHRBR) then
    llexecv(3) = .TRUE.    ! For sat_atovs.sql
  ...
```
Jb - CVAR3IN

- Modularisation (or encapsulation) of Jb
- Already fairly self-contained
- Involves ~170 subroutines including ~30 for LELAM
- Modules USE-ed
  1. Constants - same for all instances of Jb
  2. Owned by Jb - different for different instances of Jb
     - Will be moved to derived types
  3. Not owned by Jb

- CVAR3IN now compiles & links independently of IFS
- Also CVAR3INAD and Jb setup needed for 3D-Var
**Jb - Global variables**

- New derived type `TYPE_JB_STRUCT` containing derived types in `YOMJG` and `YOMWAVELET`

- All global variables will be removed and all references to variables in `YOMJG` and `YOMWAVELET` will be converted to references to elements of the derived types

- For example `LJBWAVELET` will become `JB_STRUCT%WJBCONF%LJBWAVELET`

- Fortran does not allow elements of a derived type to be specified by in a namelist - so namelist read must contain name of entire derived type 😞
Scalability

- OOPS will allow us to uncover more parallelism in 4D-Var
- Run with 1 execution instead of 7 will reduce start-up costs and I/O
- For weak constraint 4D-Var: Traj and Min steps for different sub-windows of time window can run in parallel as part of same execution - not possible in current IFS
Scalability of T1279 Forecast and 4D-Var

Speed-up

Operations 48 Nodes

User Threads on IBM Power6

10-day Forecast

4D-Var

Min_0 + Traj_0

Min_1 + Min_2

Traj_1 + Traj_2

ALADIN / HIRLAM - 07-04-11
Cleaning and reorganisation for Cycle 38

• Karim’s document V6e + pre-OOPS reorganisation
  • New derived types
    • SLCOMM and YOMGEM (Appendix J)
    • GOMS_MIX and YOMGLOBS
    • IFSFieldset
  • Remove YOMDIM from PP routines
  • Removal of unused variables in modules (Appendix G)
  • PARAMETER constants in YOMLUN
  • Split HOP
  • Move 600000 lines of data set-up (Appendix N)
    • IFS source reduced from 1600000 to 1000000 lines 😊
    • Cleaning of ODB & new ODB norms checker
• Tested in 37r3 - bit-reproducible with 37r2 for 4D-Var
Coding Norms for Cycle 38

- Coding Norm improvements
  - Removal of unused declarations - CCPT(04)
  - Modules USE with ONLY - NORM(09)
    - Remainder OK - eg. YOMDB and MPL_MODULE
  - 5565 in CY36 -> 4606 in CY37 -> 3482 in CY37R3

<table>
<thead>
<tr>
<th></th>
<th>CY36</th>
<th>CY37</th>
<th>CY37R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Useless declarations</td>
<td>1057</td>
<td>452</td>
<td>188</td>
</tr>
<tr>
<td>USE without ONLY</td>
<td>927</td>
<td>869</td>
<td>619</td>
</tr>
<tr>
<td>&gt; 50 arguments</td>
<td>1326</td>
<td>1280</td>
<td>1249</td>
</tr>
<tr>
<td>GOTO</td>
<td>706</td>
<td>673</td>
<td>639</td>
</tr>
</tbody>
</table>

- New coding norm document Olivier Rivière & Mike Fisher
Fortran Main
MPL_INIT
Set constants

C++ control layer - replaces CNT0 etc
Definition of Algorithm

Methods written in Fortran

GMV, GFL, surface_fields

Observations & ODB

COBSALL

CVAR3IN

HOP
IFS Fortran plugs into OOPS C++

- New directory ‘oops’
- Contains methods to be called by C++
- Methods will be written in Fortran and interface to IFS routines
- Modifications to ‘current IFS’ routines will be minimised
List of methods needed by OOPS

- **Geometry**
  ...
- **Model**
  ...
- **Fields**
  ...
- **Observation Operators**
  - radian_setup
  - radian_delete
  - rad_equiv  -> HOP
  - rad_equiv_tl  -> HOPTL
  - rad_equiv_ad  -> HOPAD
  - rad_variables
  - rad_locations

- **Observation Vectors**
  - obsvec_create
  - obsvec_delete
  - obsvec_copy
  - obsvec_assign
  - obsvec_add
  - obsvec_sub
  - obsvec_mul
  - obsvec_axpy
  - obsvec_dotprod
  - obsvec_diff  -> HDEPART
  - obsvec_read  -> GETDB
  - obsvec_save  -> PUTDB
List of Methods needed by OOPS contd.

**Covariance Matrices: B & R**

- **Background Errors**
  - `b_setup`
  - `b_delete`
  - `b_mult`
  - `b_imul`
  - `b_sqrt` -> CVAR3IN
  - `b_sqrtad` -> CVAR3INAD
  - `b_sqrti` ->
  - `b_sqrtiad` ->

- **Observation Errors**
  - `r_setup`
  - `r_delete`
  - `r_mult`
  - `r_imul`
  - `r_sqrt` -> OBSERR
  - `r_sqrtad` -> OBSERR
  - `r_sqrti` -> OBSERR
  - `r_sqrtiad` -> OBSERR
Warning ...
Some aspects are still under discussion
Your input is important
Questions?