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
- minioops

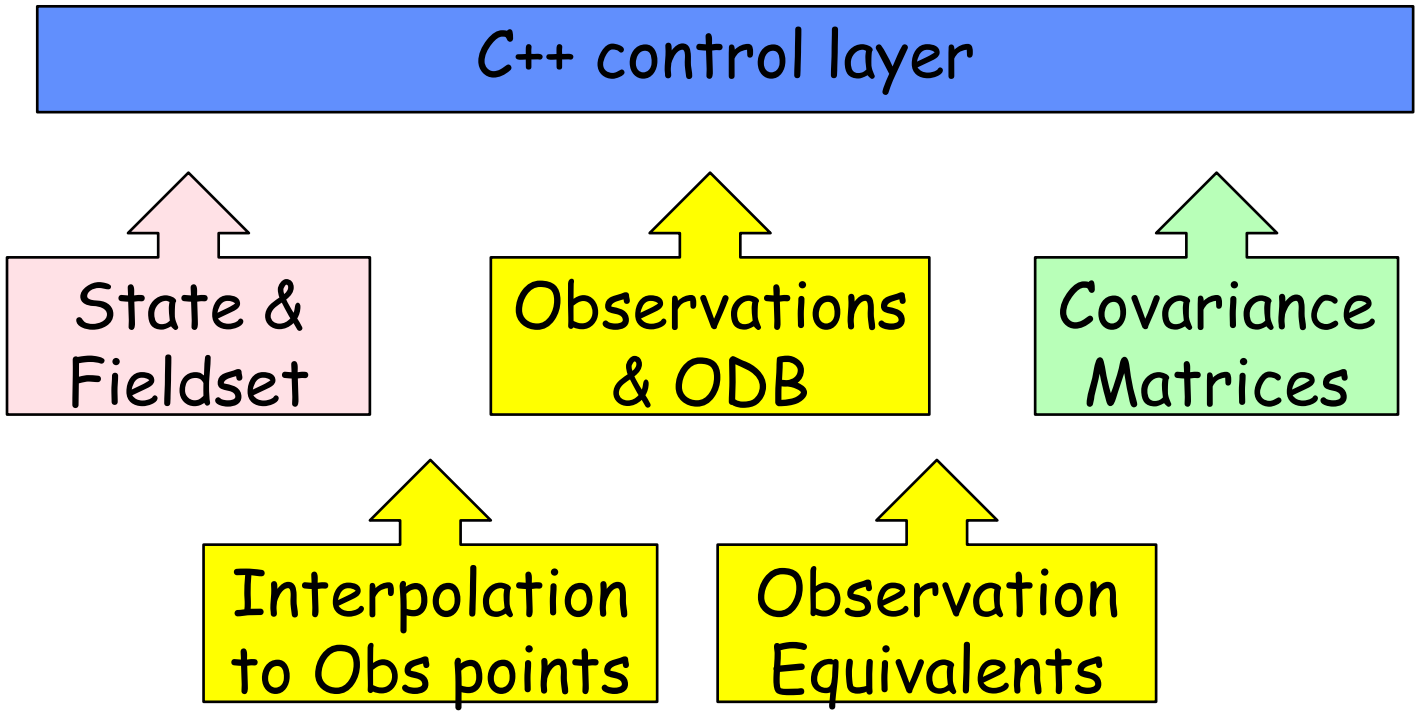
Yannick Trémolet, Mike Fisher, Anne Fouilloux, John Hague,
Tomas Wilhelmsson, George Mozdzynski, Mats Hamrud,
Karim Yessad, Jean-Noël Thépaut and Deborah Salmond

Warning …
Work-in-Progress

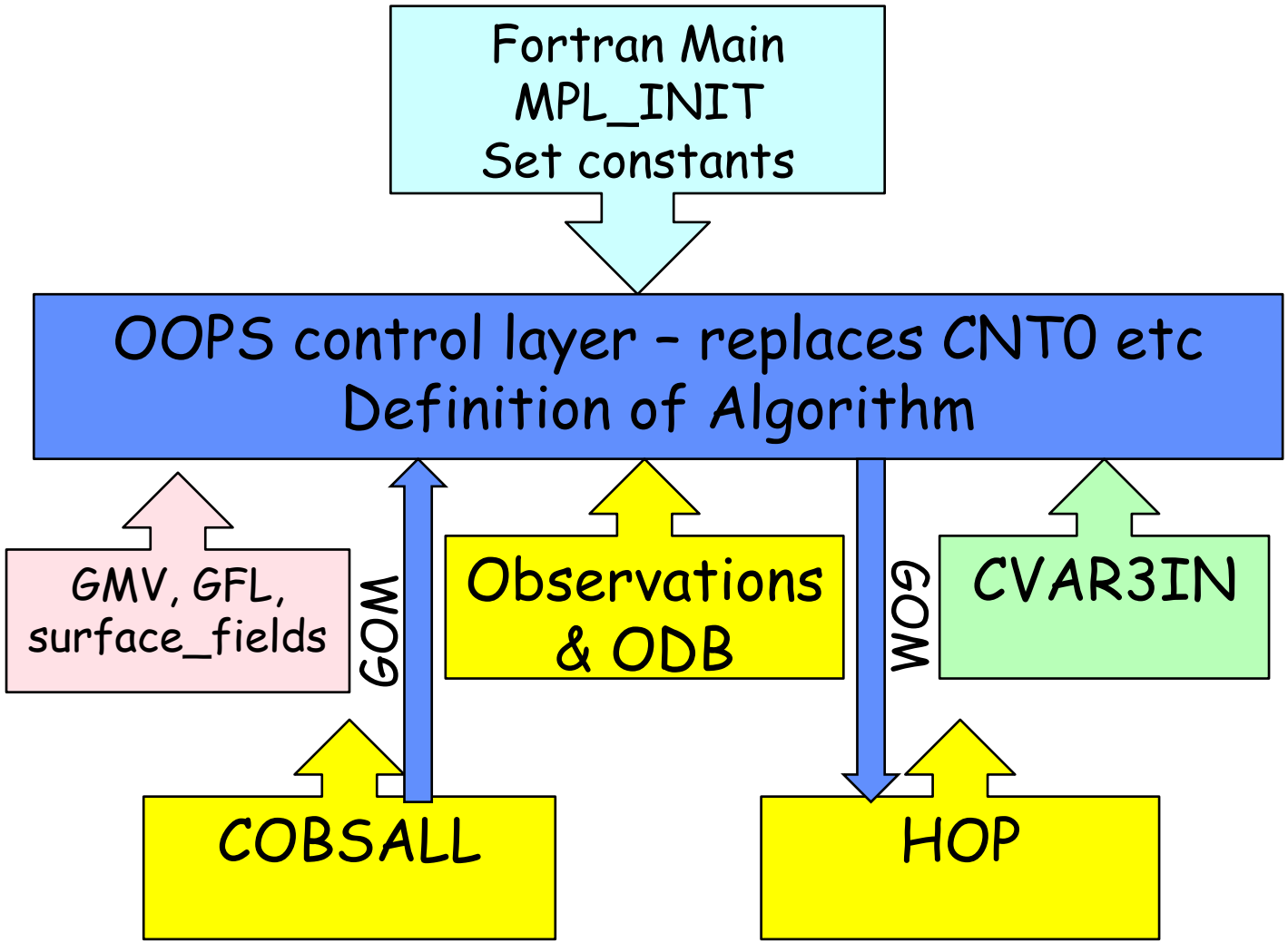
3D-Var demonstration of IFS in OOPS

- 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

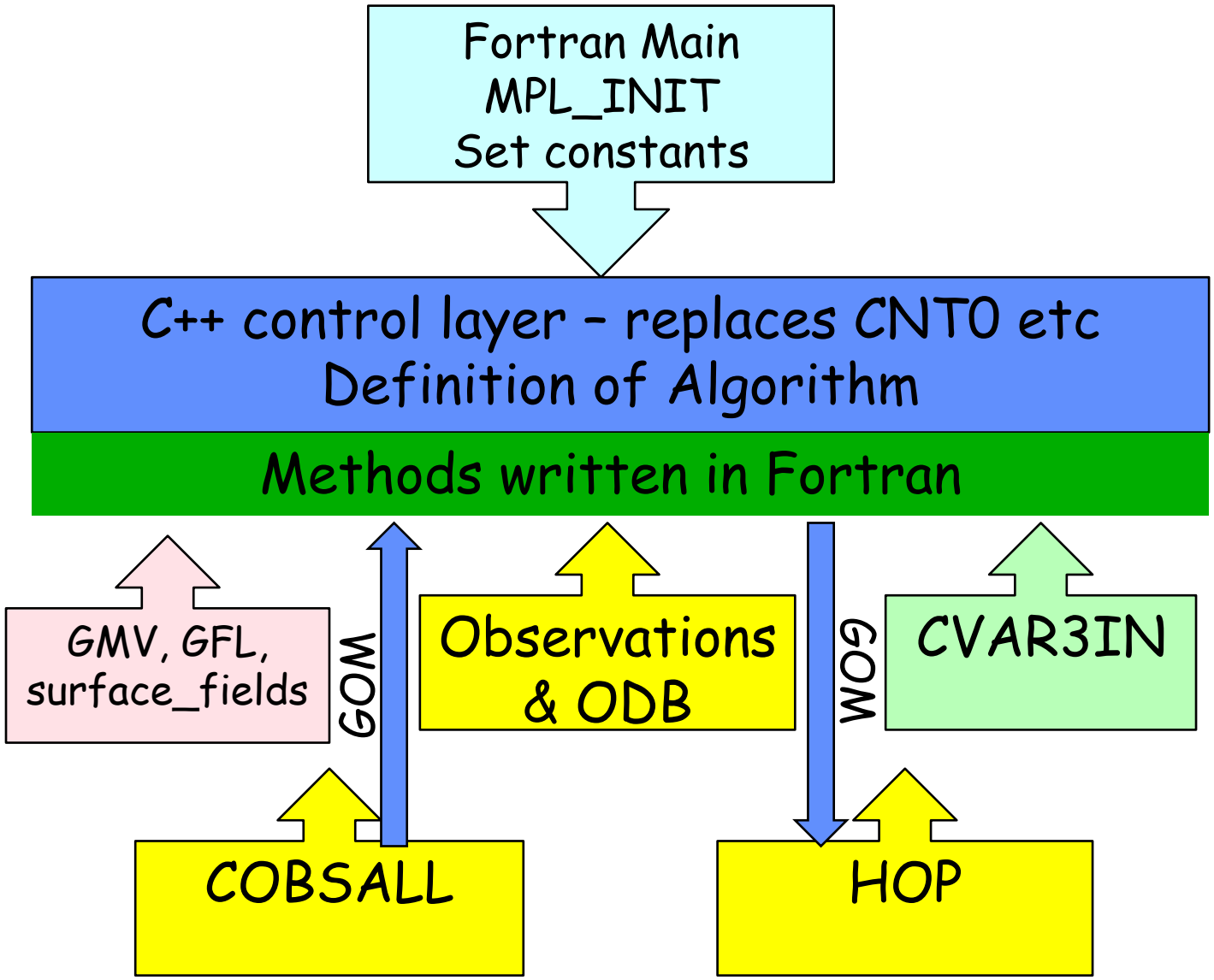
Proposed OOPS-Framework



OOPS-IFS


Fortran Main
MPL_INIT
Set constants
OOPS control layer – replaces CNT0 etc
Definition of Algorithm
GMV, GFL,
surface_fields
GOM
Observations
& ODB
WOG
CVAR3IN
COBSALL
HOP

OOPS-IFS

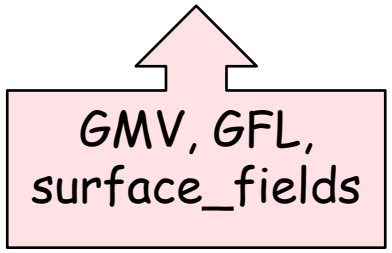

Fortran Main
MPL_INIT
Set constants
C++ control layer – replaces CNT0 etc
Definition of Algorithm
Methods written in Fortran
GMV, GFL,
surface_fields
GOM
Observations
& ODB
GOM
CVAR3IN
COBSALL
HOP

IFS Fieldset

Tomas

Derived Types for IFS Fieldset:

```
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.
```

Multiple instances of data in IFS MODULES

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

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

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

END MODULE YOMGFL
```

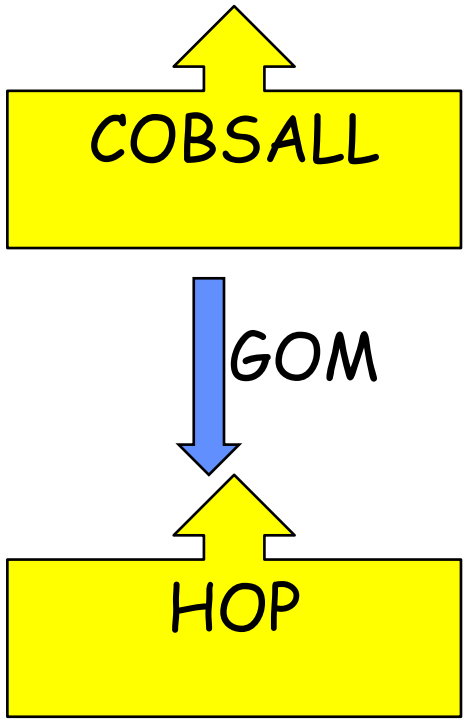
Original

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

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

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 — Anne

- 94 subroutine calls directly from HOP and 231 IF tests

- In hop.F90 we have:

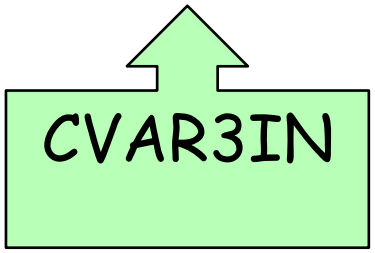
```
...
  CALL GETDB('HOP', NUPTRA, IRET, INFO, 0, ZINFO, 0, &
   & KSET, ITSLOT, -1, IOBSTYPE, ICDTYP_TOVS, ISENSOR_GETDB)

IF (ICDTYP_TOVS == NSTRO3) then
...
elseif  (ICDTYP_TOVS == NGTHRB)  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 == NGTHRB)  then
       llexecv(3) = .TRUE.               ! For sat_atovs.sql
...
```

Jb – CVAR3IN

Mike



- 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
```

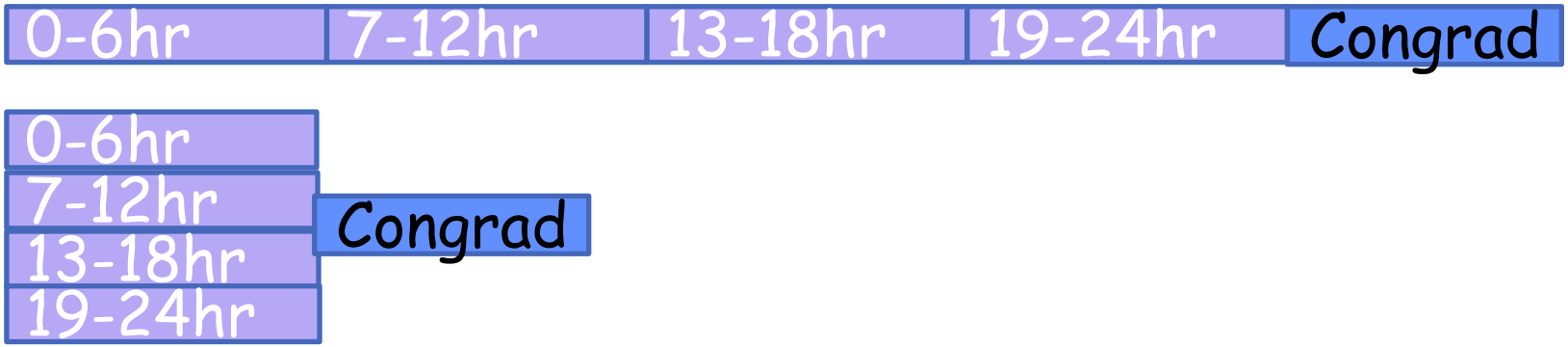
- Fortran does not allow elements of a derived type to be specified by in a namelist – so namelist read must contain name of entire derived type ☹

Scalability

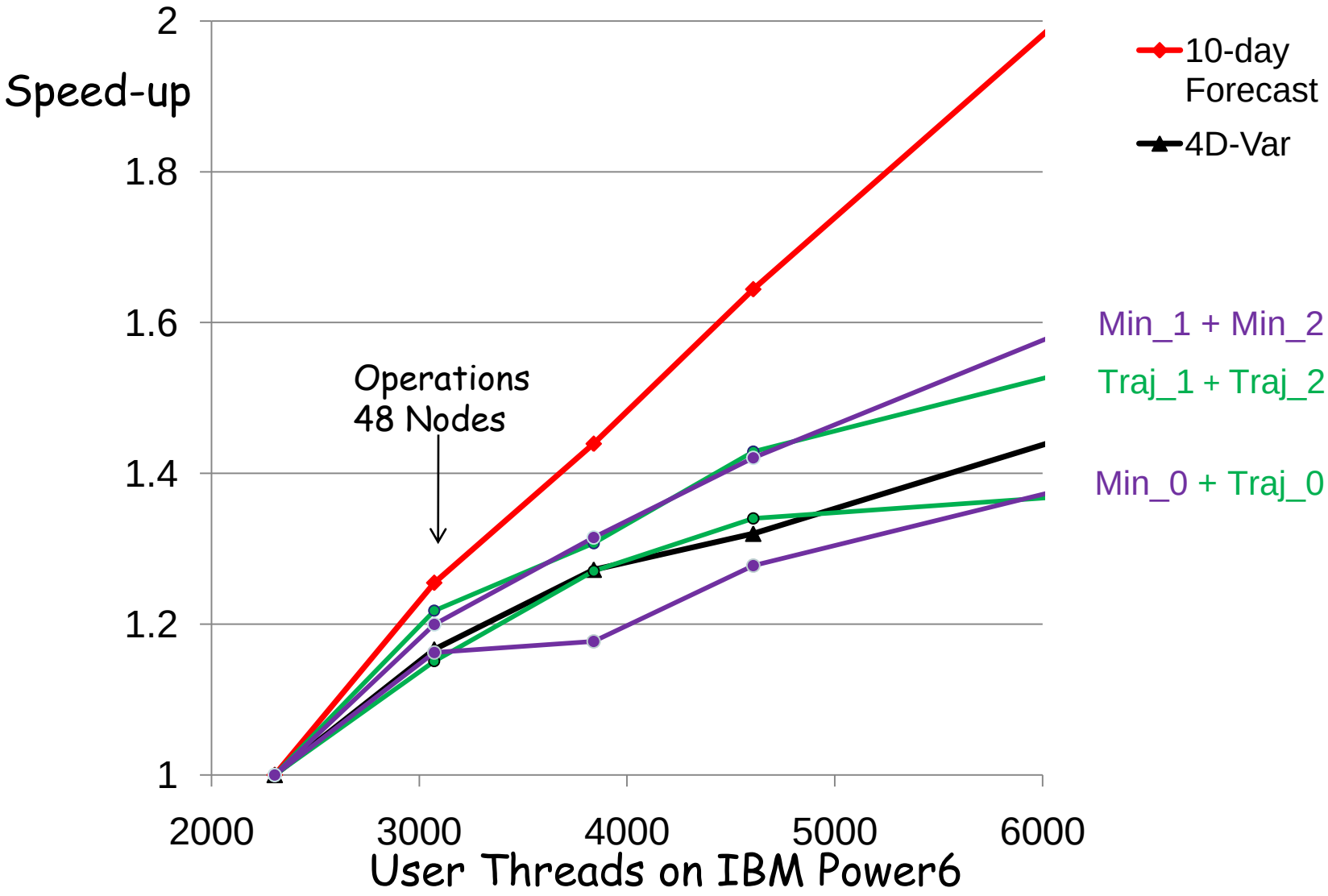
- OOPS will allow us to uncover more parallelism in 4D-Var

- 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



ECMWF

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

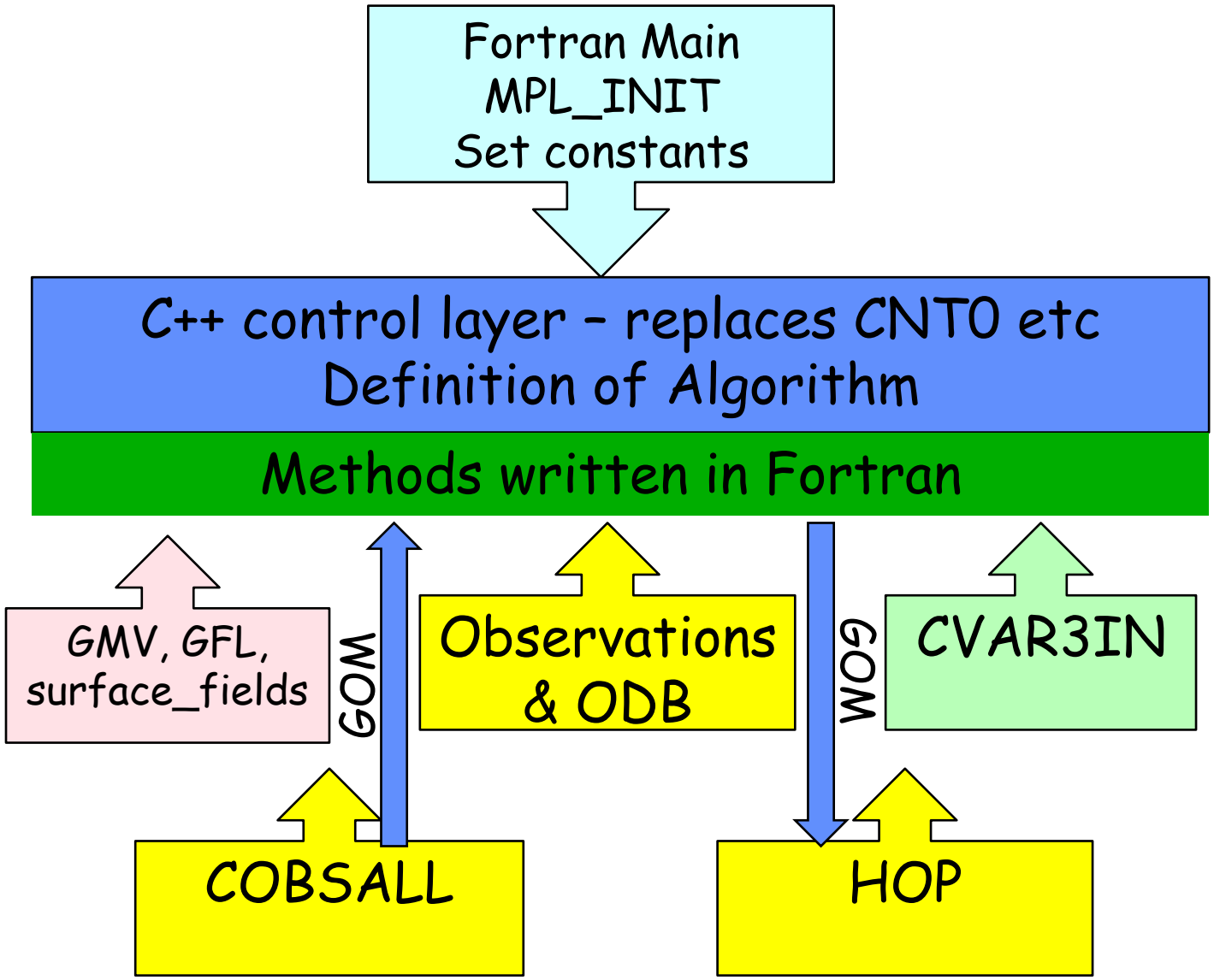
Coding Norms for Cycle 38

- 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

	CY36	CY37	CY37R3
Useless declarations	1057	452	188
USE without ONLY	927	869	619
> 50 arguments	1326	1280	1249
GOTO	706	673	639

- New coding norm document Olivier Rivière & Mike Fisher

OOPS-IFS


Fortran Main
MPL_INIT
Set constants
C++ control layer – replaces CNT0 etc
Definition of Algorithm
Methods written in Fortran
GMV, GFL,
surface_fields
GOM
Observations
& ODB
GOM
CVAR3IN
COBSALL
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

```
radiance_setup
radiance_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
```

Minioops.cc

```
// Test for obsvec methods to be used with 3D-Var (to be called by Fortran Main)
// Always use HOP SQL

extern "C" {
int minioops_() {

// obs1 is obs value
  void * obs1;
// obs2 is obs equiv
  void * obs2;
// obs3 is departure
  void * obs3;

//Constructors
  obsvec_create_(&obs1,"obsvalue");
  obsvec_create_(&obs2,"obsequiv");
  obsvec_create_(&obs3,"obsdepar");

// read obs value from ODB
  obsvec_read_(&obs1);

// Calculate obs equiv from HOP
  obsvec_equiv_(&obs2);

// Calculate departure obs3=obs2-obs1
  obsvec_diff_(&obs3,&obs2,&obs1);
```

Ifs_obsvecs.F90

```
MODULE IFS_OBSVEC

USE ISO_C_BINDING, ONLY : C_CHAR, C_PTR, C_LOC,  C_F_POINTER

USE PARKIND1
USE YOMDB

IMPLICIT NONE

PRIVATE

PUBLIC :: OBSVEC,OBSVEC_CREATE,OBSVEC_READ,OBSVEC_DIFF,OBSVEC_DELETE,OBSVEC_EQUIV

TYPE OBSVEC

  REAL(KIND=JPRB),POINTER         :: VAL(:)
  INTEGER(KIND=JPIM)              :: ISIZE
  INTEGER(KIND=JPIM)              :: IOBSCOL
  CHARACTER(LEN=128)              :: CNAME
  INTEGER(KIND=JPIM),POINTER      :: ISTART(:)
  INTEGER(KIND=JPIM),POINTER      :: IEND(:)

END TYPE OBSVEC

! -----------------------------------------------------------------------------
CONTAINS
! -----------------------------------------------------------------------------
```

Ifs_obsvecs.F90

```
SUBROUTINE OBSVEC_DIFF(C_SELF,C_OBSVAL,C_EQUIV) BIND(C,NAME='obsvec_diff_')

TYPE(C_PTR),INTENT(INOUT) :: C_SELF
TYPE(C_PTR),INTENT(IN)    :: C_OBSVAL
TYPE(C_PTR),INTENT(IN)    :: C_EQUIV


IMPLICIT NONE

TYPE(OBSVEC), POINTER :: SELF, OBSVAL, EQUIV

CALL C_F_POINTER(C_SELF,SELF)
CALL C_F_POINTER(C_OBSVAL,OBSVAL)
CALL C_F_POINTER(C_EQUIV,EQUIV)


SELF%VAL(1:SELF%ISIZE)=OBSVAL%VAL(1:SELF%ISIZE)-EQUIV%VAL(1:SELF%ISIZE)

RETURN
END SUBROUTINE OBSVEC_DIFF
```

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
r_sqrtad
r_sqrti    ->/OBSERR
r_sqrtiad  ->/OBSERR
```

Warning …
Some aspects are still under discussion

Questions?