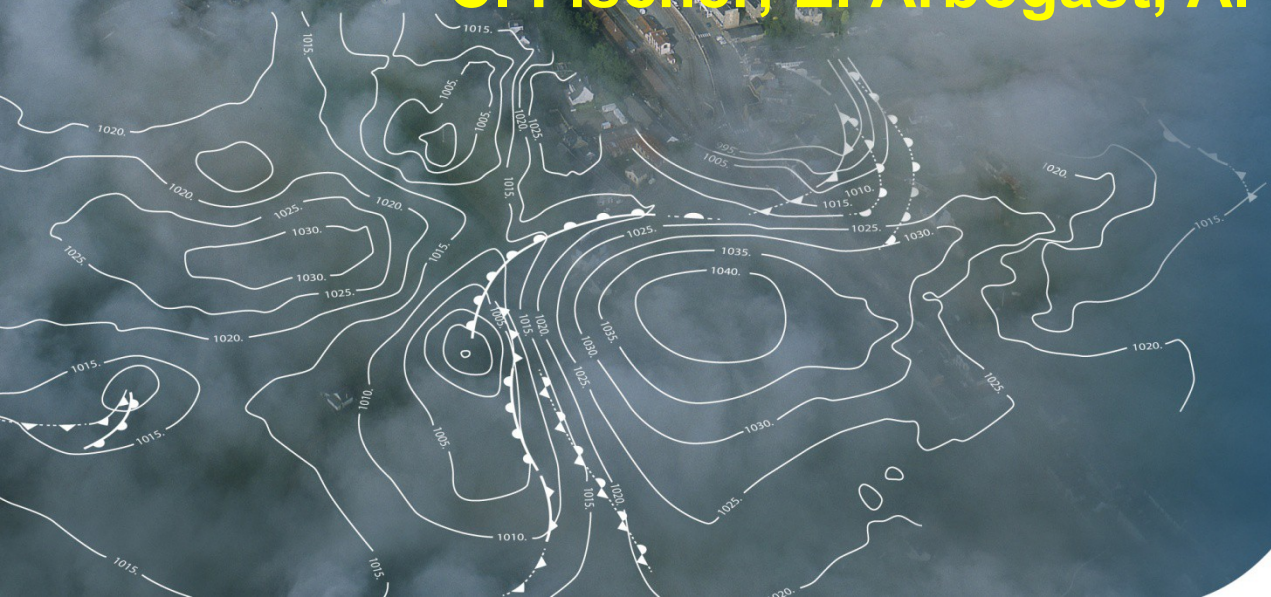
OOPS

C. Fischer, E. Arbogast, A. Mary



METEO FRANCE
Toujours un temps d'avance

Content

- Programming language aspects
- Project management

Code structure in OOPS

CNT4/STEPO (forecast):

- CNT4: model time loop, ∀ model step (init, loop, end)

⇒ OOPS

- STEPO : toolbox (calls transforms, GP computations, SP computations)

⇒ STEPO_OOPS (GP/…/GP) versus STEPO (SP/…/SP)

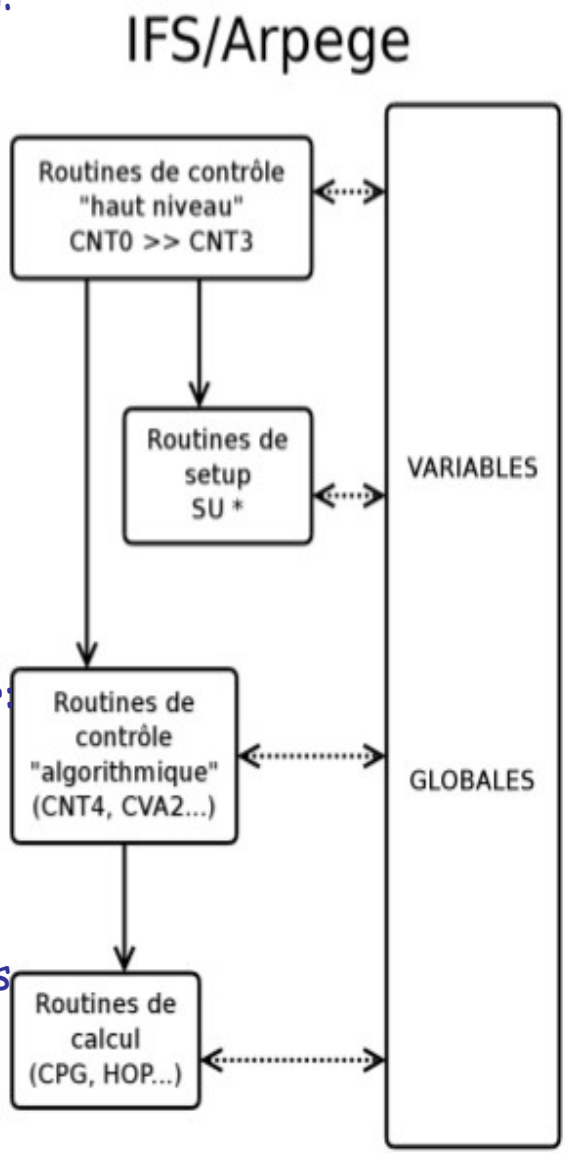
CVA2/SIM4D/Minimizer:

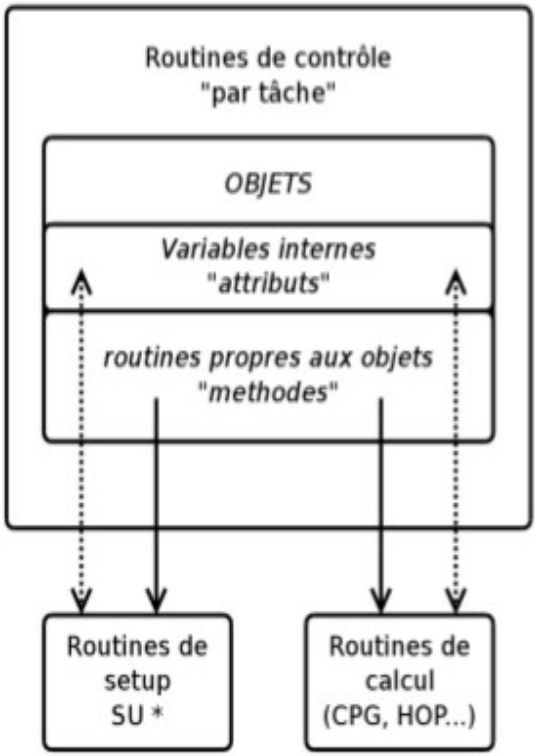
- algorithms and DA arithmetics (ex : X + dx)

⇒ OOPS

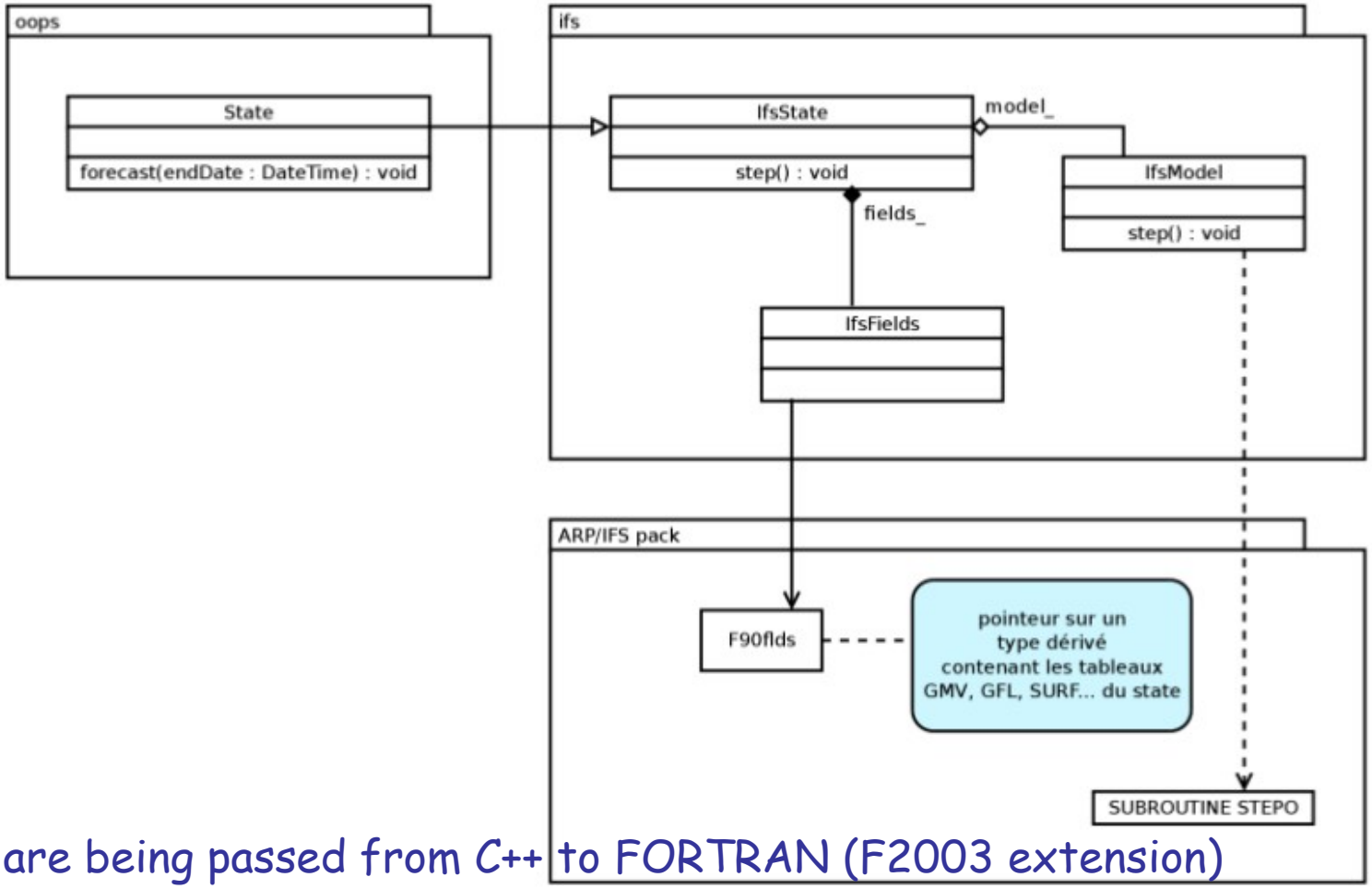
- actual compute routines and operators in FORTRAN

⇒ not OOPS





Code structure in OOPS: C++/FORTRAN interface



Pointers are being passed from C++ to FORTRAN (F2003 extension)

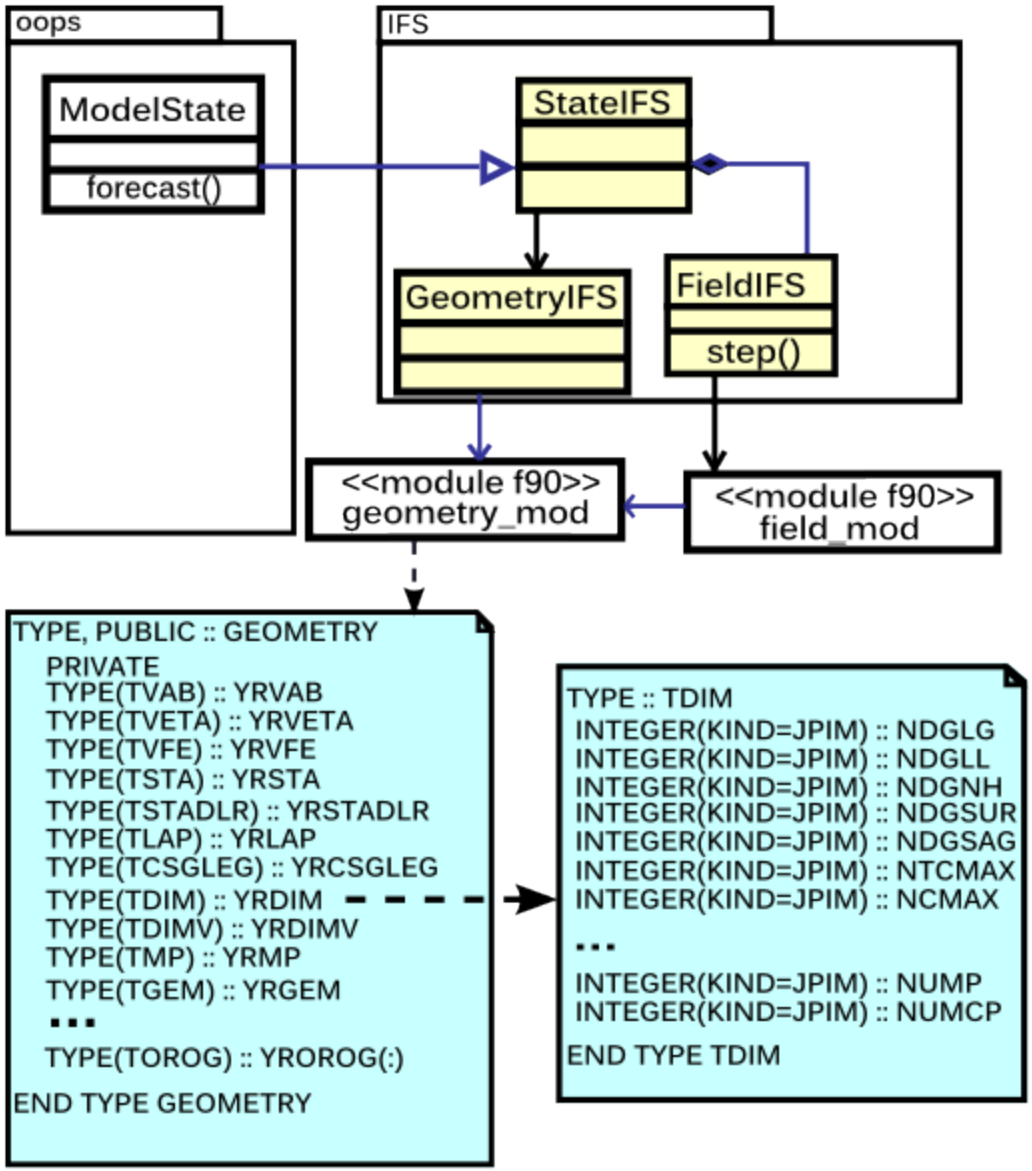
When a set of variables and data arrays is encapsulated, then the high-level pointer to the FORTRAN structure is the one coming from C++

For specific data structures that are extensively being used in the IFS, like GMVs, GFLs, the code refactoring is long and an interim solution for handling multiple pointer assignments has been implemented (so-called Tomas' trick)

Encapsulation of variables in MODULEs

- Group all variables in a MODULE into one derived type
- Pass this derived type as argument in calling seq:

  STEPO_OOPS(FIELDS FLD)
  won't use the global variable NPROMA, but instead:
  FLD%GEOM%YRDIM%NPROMA



Encapsulation of variables in MODULEs: ASSOCIATE

Use of an **alias** in order to keep source code readable:

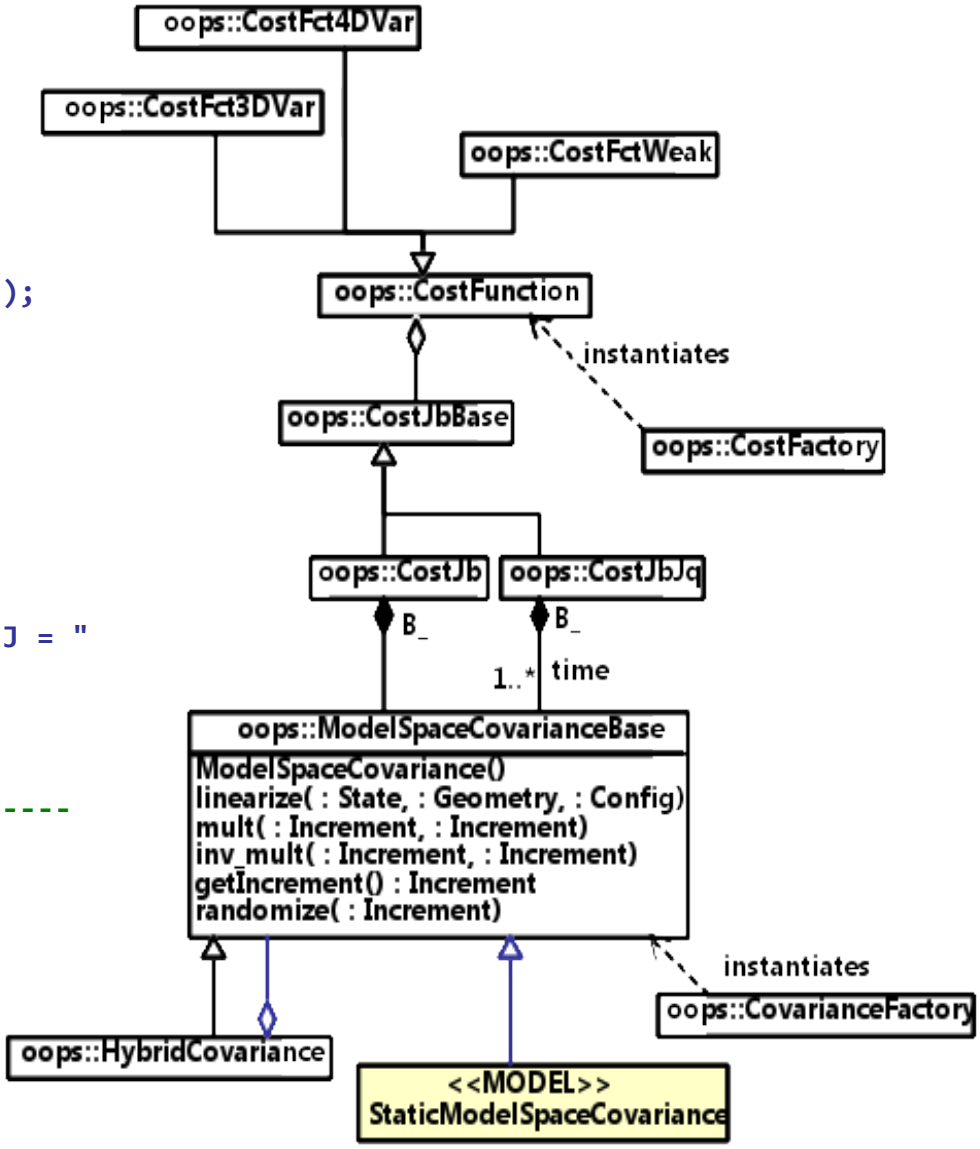
Definition of the alias

Use of the aliased variables inside compute code

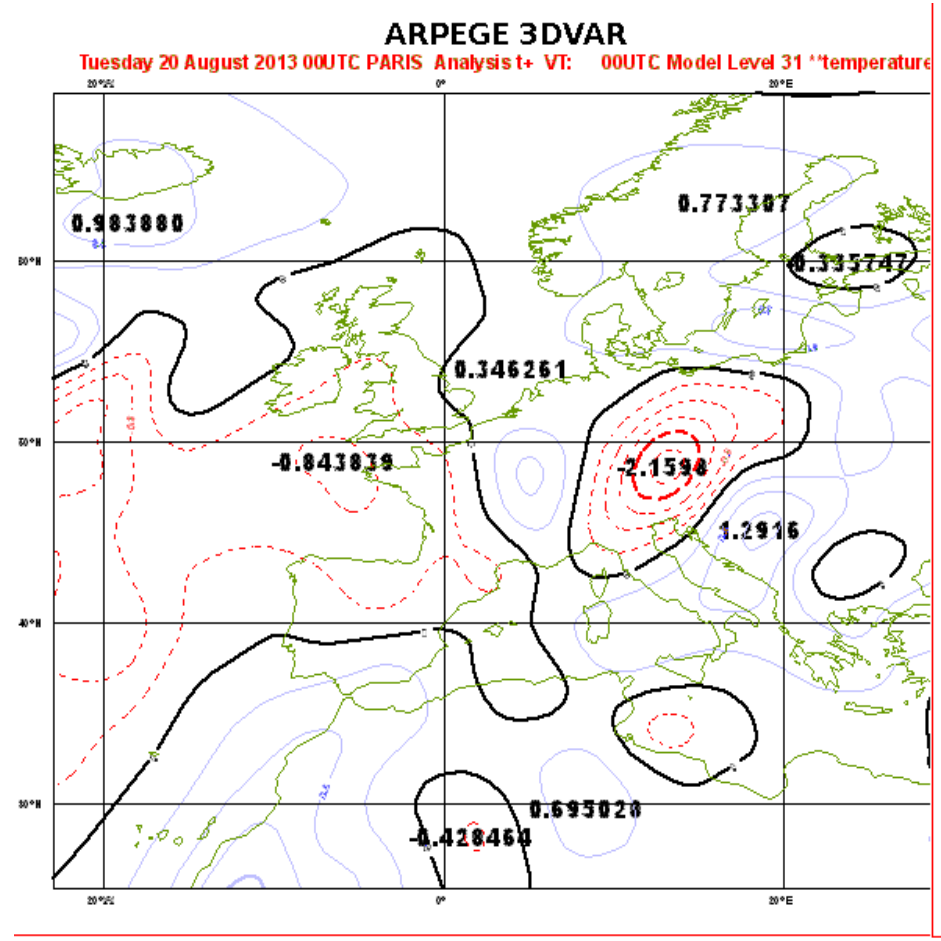
```
SUBROUTINE MF_PHYS (..., YRDIM, ...)
!-----------------------------------------------------
ASSOCIATE(NPROMA=>YRDIM%NPROMA, NPROMM=>YRDIM%NPROMM, &
 & YT0=>YRGMV%YT0, YT9=>YRGMV%YT9, YT1=>YRGMV%YT1, &
...
 & NFLEVG=>YRDIMV%NFLEVG, NFLSA=>YRDIMV%NFLSA, &
 & NFLSUL=>YRDIMV%NFLSUL)
IF (LHOOK) CALL DR_HOOK('MF_PHYS',0,ZHOOK_HANDLE)

...
IF (LLDIAB.AND.(NDPSFI == 1)) THEN
    CALL CPQSOL(NPROMA,KST,KEND,PRE0,PSP_RR(1,YSP_RR%YT
    %MP0),PQS,ZQSATS,PQSOL)
ENDIF
...



IF (LHOOK) CALL DR_HOOK('MF_PHYS',1,ZHOOK_HANDLE)
END ASSOCIATE
END SUBROUTINE MF_PHYS
```

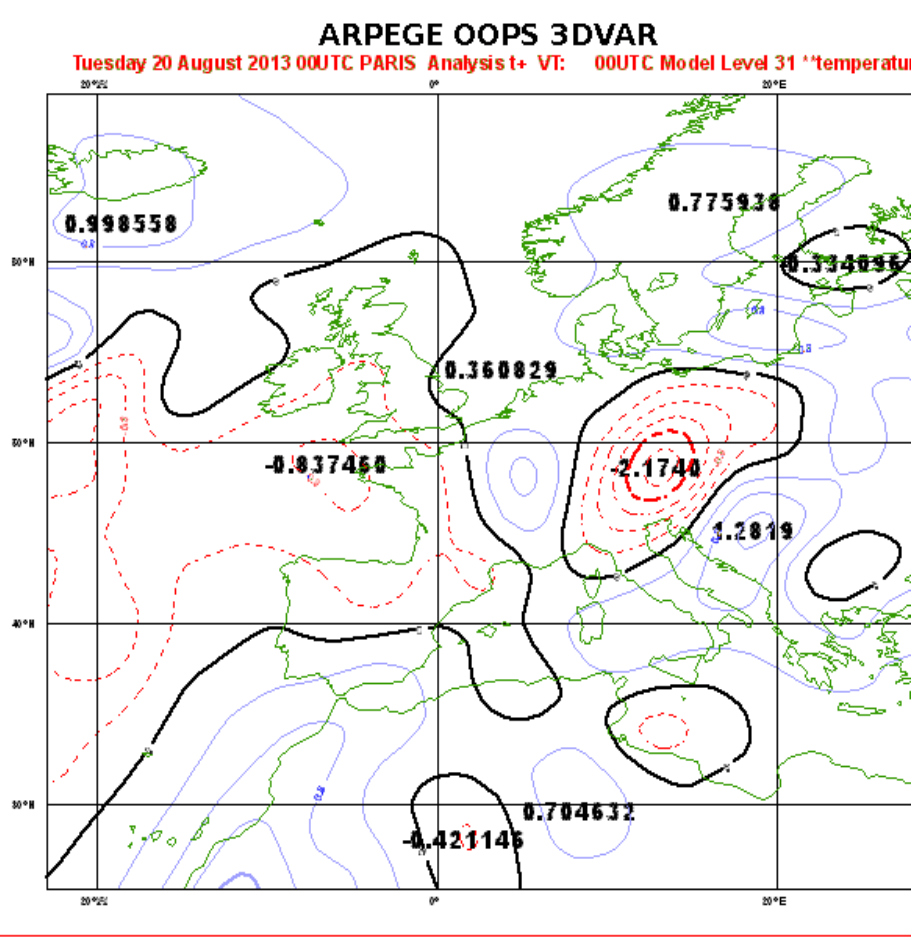
Cost functions in OOPS

```
//-----------------------------------------
double CostFunction::evaluate(const CtrlVar_ &
fguess) {
// Setup terms of cost function
  PostProcessor pp;
  pp.enrollProcessor(jterms_.initialize(fguess));

// Run non-linear model
  CtrlVar_ xx(fguess);
  this->runNL(xx, pp);

// Cost function value
  double zzz = jb_->evaluate(fguess, xx);
  zzz += jterms_.finalize();

  LOGS(Info, Test) << "CostFunction: Nonlinear J = "
<< zzz;
  return zzz;
}

//-------------------------------------------------
void CostFct4DVar::runNL(CtrlVar_ & xx,
PostProcessor & post) {
  ModelState zz(xx[0], CostFct_::getModel());
  zz.forecast(windowLength_, post);
  ASSERT(zz.validTime() == windowEnd_);
  xx[0] = zz;
}
```



Compared results between the OOPS ARPEGE 3D-VAR « *maquette* » and the classical IFS-ARP code





Project management since December 2014

- OOPS is now one project (among others) of the Scalability Program at ECMWF
- OOPS Board (chair: F. Rabier): MF/LAM representative (C. Fischer)
- OOPS Project Manager (PM): Y. Trémolet
- PM has insisted on difficulty to keep OOPS-IFS prototypes working from one IFS cycle to the next => code will be extracted when necessary from the main IFS cycles, re-factored and kept in a separate SCR (GIT-OOPS) => *duplicated IFS and OOPS-IFS SCRs and associated risks of code divergence or coordination issues (followed by Board meetings and PMs);*
- Re-factoring shall be also back-phased into main IFS cycles (ECMWF)
- ECMWF and MF have defined the obs operators as a main target for OOPS for 2nd half of 2015 (and beyond ?)
- ECMWF will also work on an adiabatic version of the IFS forecast model for early tests of 4D-VAR from OOPS. Later, the physics codes will be added.

tak for din opmærksomhed

METEO FRANCE
Toujours un temps d'avance