OOPS developments at ECMWF

MFLA and documentation of the Harmonie DA configuration

Roel Stappers

[1]Norwegian Meteorological Institute
roels@met.no

28th ALADIN Wk & HIRLAM ASM
Toulouse
16–20 April 2018

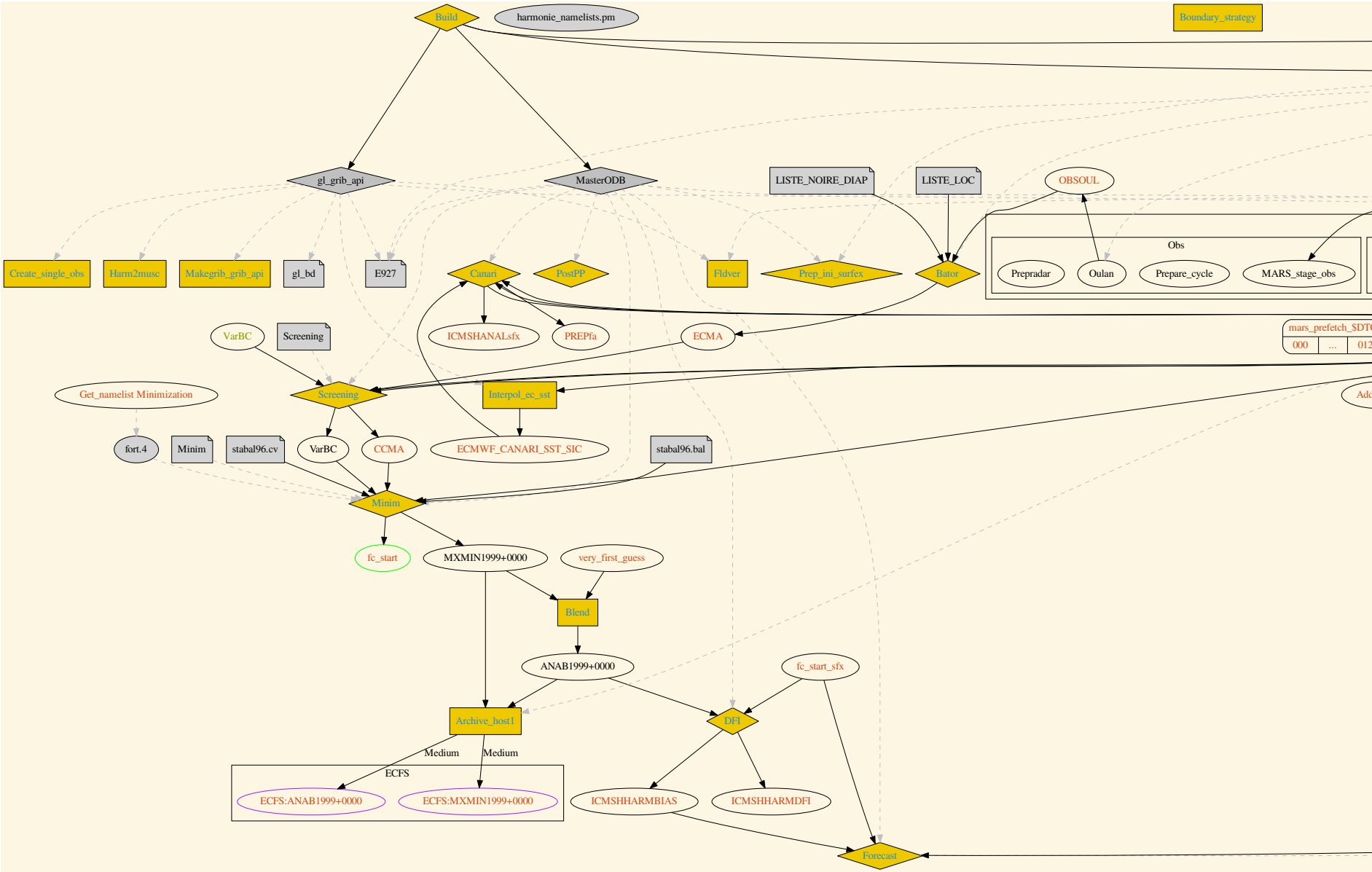
Overview

1. Documentation of the Harmonie-Arome DA system
2. OOPS
3. (Desroziers statistics for DA systems with VarBC)

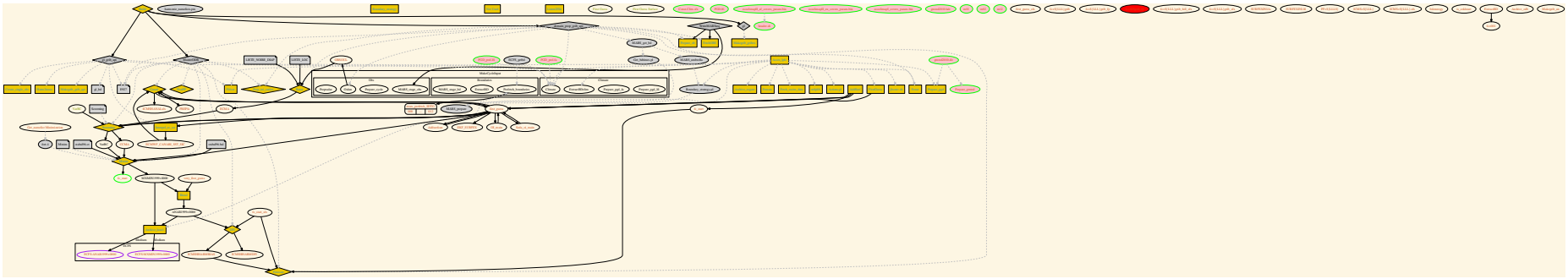
Documentation of scripting system

- From Nov. 2017 new position as Hirlam code analyst for Data assimilation
- Documentation of the Harmonie DA scripting system as preparation for the Harmonie-Arome DA CMC.

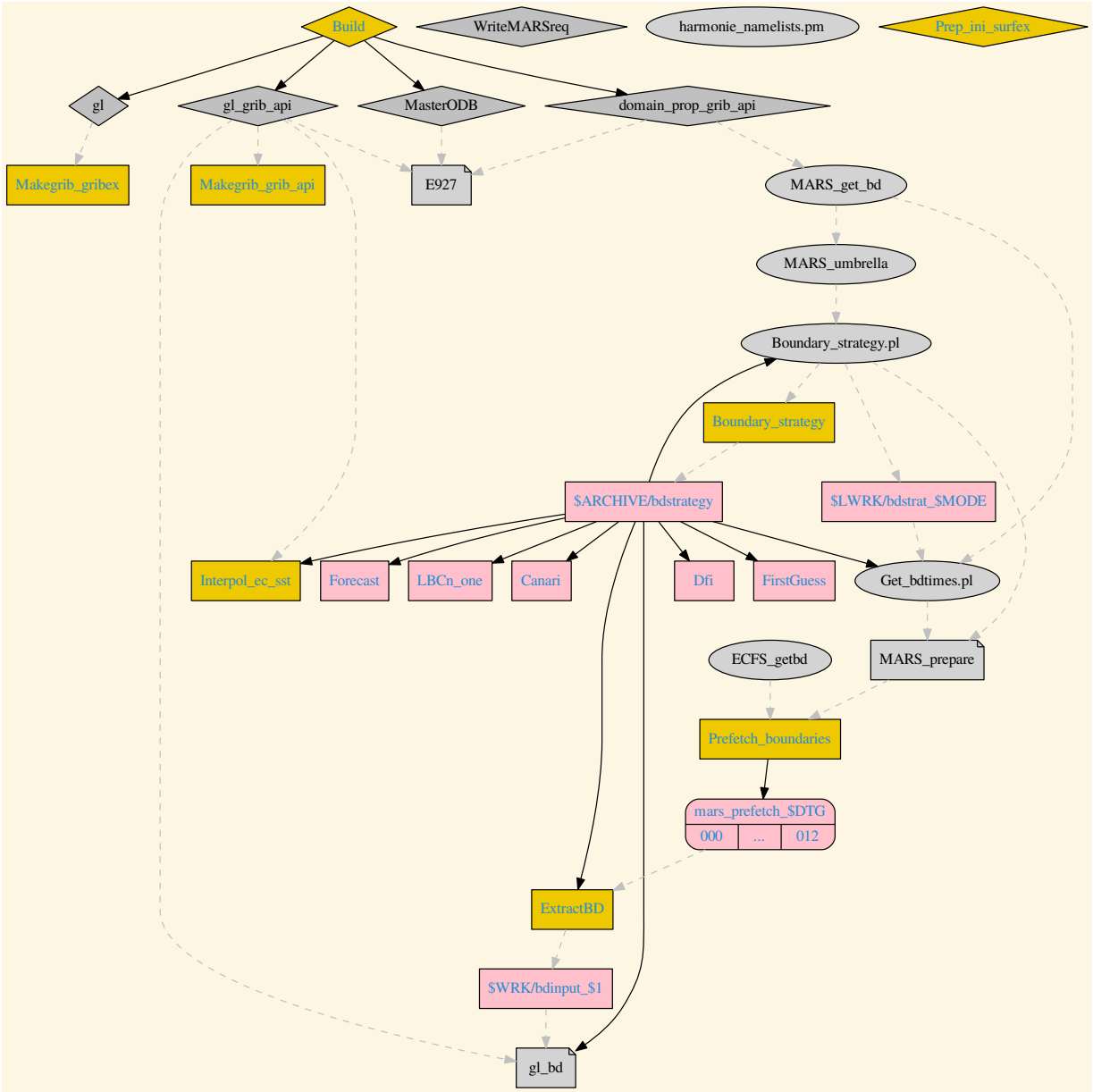
Documentation of scripting system (work in progress)



Documentation of scripting system (work in progress)



Documentation of scripting system (work in progress)



Documentation of scripting system (work in progress)

Open issues

- Grouping by observations, boundaries, climate.
- Ensemble configurations. How to represent?
- 24 hour cycle of VarBC, ...
- Operational versus research mode in script.
- Manual generation of documentation is tedious. Documentation will get out of date. Need a code structure where this can be done automatically.

Documentation of scripting system (work in progress)

Open issues

- Grouping by observations, boundaries, climate.
- Ensemble configurations. How to represent?
- 24 hour cycle of VarBC, ...
- Operational versus research mode in script.
- Manual generation of documentation is tedious. Documentation will get out of date. Need a code structure where this can be done automatically.
- Generate suite definition file automatically
- Reduce the number of if-statements in the scripts

Introduction (why OOPS)

Formulations of DA and flexibility in OOPS

Primal formulation ($\mathbf{d} = \mathbf{y} - \mathcal{H}(x_0^g)$, $b = x_0^b - x_0^g$)

$$(\mathbf{B}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})\delta x_0 = \mathbf{B}^{-1}b + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{d}$$

Formulations of DA and flexibility in OOPS

Primal formulation ($\mathbf{d} = \mathbf{y} - \mathcal{H}(x_0^g)$, $b = x_0^b - x_0^g$)

$$(\mathbf{B}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})\delta x_0 = \mathbf{B}^{-1}b + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{d}$$

Saddle point formulation

$$\begin{bmatrix} \mathbf{B}^{-1} & \mathbf{H}^T \\ \mathbf{H} & -\mathbf{R} \end{bmatrix} \begin{bmatrix} \delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{B}^{-1}b \\ \mathbf{d} \end{bmatrix}$$

Dual formulation (3D/4D-PSAS)

$$(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R})\lambda = -\mathbf{d} + \mathbf{H}b$$

$$\delta x = -\mathbf{B}\mathbf{H}^T\lambda + b$$

Formulations of DA and flexibility in OOPS

Primal formulation ($\mathbf{d} = \mathbf{y} - \mathcal{H}(x_0^g)$, $b = x_0^b - x_0^g$)

$$(\mathbf{B}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})\delta x_0 = \mathbf{B}^{-1}b + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{d}$$

Saddle point formulation

$$\begin{bmatrix} \mathbf{B}^{-1} & \mathbf{H}^T \\ \mathbf{H} & -\mathbf{R} \end{bmatrix} \begin{bmatrix} \delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{B}^{-1}b \\ \mathbf{d} \end{bmatrix}$$

Dual formulation (3D/4D-PSAS)

$$(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R})\lambda = -\mathbf{d} + \mathbf{H}b$$
$$\delta x = -\mathbf{B}\mathbf{H}^T\lambda + b$$

Weak constraint 4D-VAR

$$(\mathbf{L}^T\mathbf{D}^{-1}\mathbf{L} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})\delta\mathbf{x} = \mathbf{L}^T\mathbf{D}^{-1}\mathbf{b} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{d}$$

- Saddle point weak constraint 4D-VAR
- Flexibility to change Krylov methods (PCG, MINRES, RPCG, GMRES)
- For $\mathbf{A}\mathbf{x} = \mathbf{b}$ solution $\mathbf{x}^* \in \mathrm{span}(\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \cdots, \mathbf{A}^n\mathbf{b})$

Increments w.r.t background (not in OOPS yet)

Primal formulation ($\mathbf{d} = \mathbf{y} - \mathcal{H}(x_0^g)$, $b = x_0^b - x_0^g$)

$$(\mathbf{B}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})\delta x_0^g = \mathbf{B}^{-1}b + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{d}$$

Primal formulation increment w.r.t background $\delta x_0^b = \delta x_0^g - b$

$$(\mathbf{B}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})\delta x_0^b = \mathbf{H}^T\mathbf{R}^{-1}(\mathbf{d} + \mathbf{H}b)$$

OOPS team

- Project management: Stephen English
- Core OOPS: Mats Hamrud, Deborah Salmon, Marcin Chrust, Olivier Marsden.
- And: Ryad El Khatib, Alan Geer, Peter Lean, Elias Holm etc.

Code structure

- OOPS src/oops (SLOC 12000 C++, separate git repository)

```
src/oops/assimilation/ls *Minimizer.h
GMRESRMinimizer.h    RPCGMinimizer.h         DualMinimizer.h       IPCGMinimizer.h
Minimizer.h          PLanczosMinimizer.h     RPLanczosMinimizer.h DRIPCGMinimizer.h
DRPFOMMinimizer.h    FGMRESMinimizer.h       LBGMRESRMinimizer.h  MINRESMinimizer.h
PrimalMinimizer.h    SaddlePointMinimizer.h DRMinimizer.h         DRPLanczosMinimizer.h
GMRESRMinimizer.h    LBMinimizer.h           PCGMinimizer.h        RPCGMinimizer.h

src/oops/assimilation/ls *Matrix.h
BMatrix.h     HessianMatrix.h  HtMatrix.h       LBHessianMatrix.h  SaddlePointLMPMatrix.h
HBHtMatrix.h  HMatrix.h        HtRinvHMatrix.h  RinvMatrix.h       SaddlePointMatrix.h
SaddlePointPrecondMatrix.h     (UtHtRinvHUMatrix.h)
```

Code structure

- OOPS src/oops (SLOC 12000 C++, separate git repository)

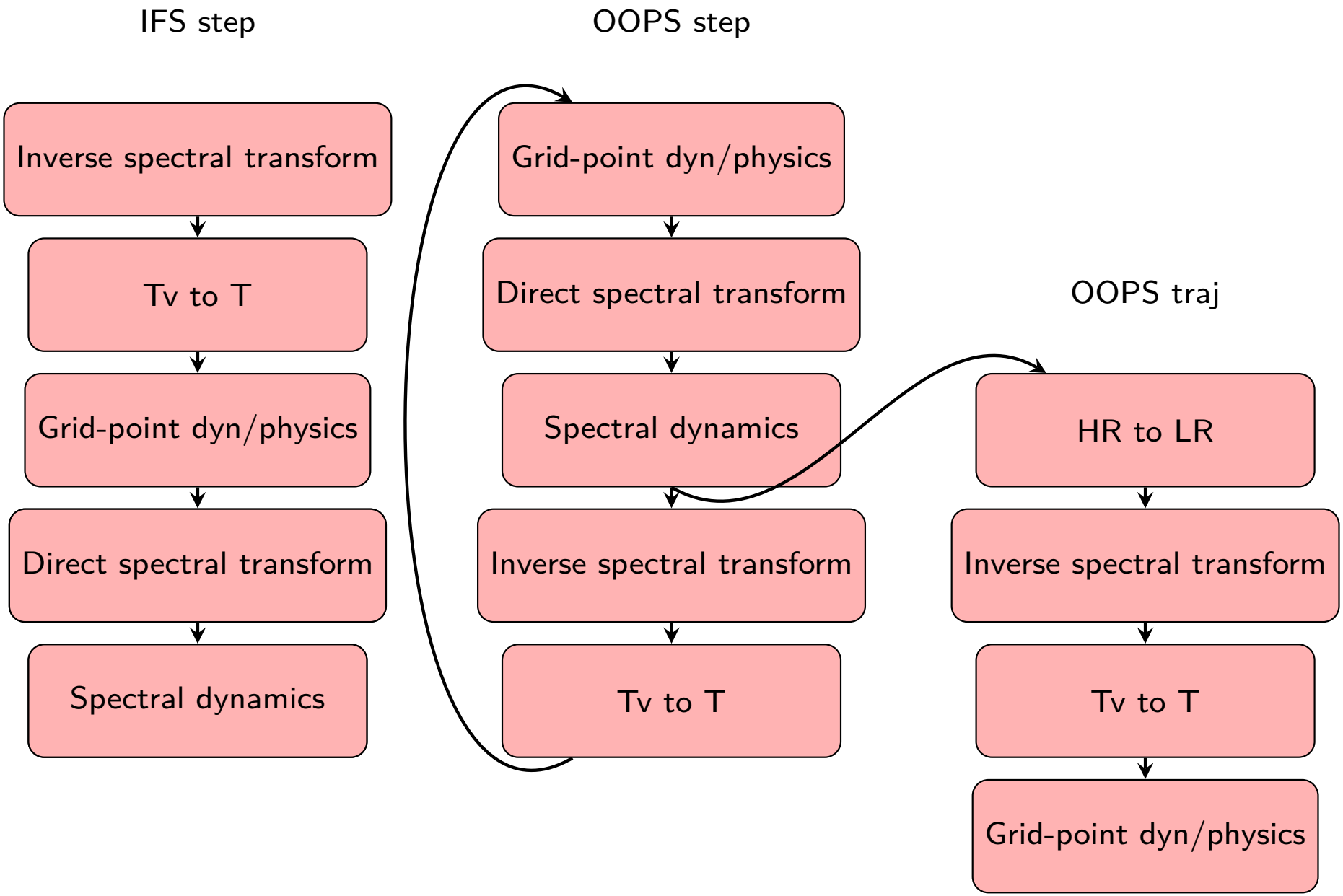
```
src/oops/assimilation/ls *Minimizer.h
GMRESRMinimizer.h     RPCGMinimizer.h          DualMinimizer.h       IPCGMinimizer.h
Minimizer.h           PLanczosMinimizer.h      RPLanczosMinimizer.h  DRIPCGMinimizer.h
DRPFOMMinimizer.h     FGMRESMinimizer.h        LBGMRESRMinimizer.h   MINRESMinimizer.h
PrimalMinimizer.h     SaddlePointMinimizer.h   DRMinimizer.h         DRPLanczosMinimizer.h
GMRESRMinimizer.h     LBMinimizer.h            PCGMinimizer.h        RPCGMinimizer.h

src/oops/assimilation/ls *Matrix.h
BMatrix.h      HessianMatrix.h  HtMatrix.h        LBHessianMatrix.h  SaddlePointLMPMatrix.h
HBHtMatrix.h   HMatrix.h        HtRinvHMatrix.h   RinvMatrix.h       SaddlePointMatrix.h
SaddlePointPrecondMatrix.h      (UtHtRinvHUMatrix.h)
```

- ifs-source/oopsifs/src/ifs (SLOC 2500 C++/ 2000 Fortran)

```
ls *.interface.F90
AllObsCovariance.interface.F90   FieldsIFS.interface.F90           LocalizationMatrixIFS.interface.F90
ObsBiasIncrement.interface.F90   VariablesIFS.interface.F90        AllObs.interface.F90
GeometryIFS.interface.F90        LocationsIFS.interface.F90        ObsBias.interface.F90
AllObsTLAD.interface.F90         GomData.interface.F90             ModelIFS.interface.F90
ObsSpaceODB.interface.F90        ErrorCovariance3D.interface.F90   LinearModelIFS.interface.F90
ObsBiasCovariance.interface.F90  ObsVector.interface.F90
```

- ifs-source/src/ifs/oops (SLOC 6000 Fortran)

Multi-Resolution (from Mats)



Spamming

- For cycles ≤ 41 , the IFS stored most of its mutable data in modules, and routines accessed this data implicitly via `USE MODULE, ONLY : DATA`
- A necessary but not sufficient step towards functional OOPS has been to modify all IFS routines to work on data passed as arguments. This was done "automatically".

Spamming

- For cycles ≤ 41 , the IFS stored most of its mutable data in modules, and routines accessed this data implicitly via `USE MODULE, ONLY : DATA`
- A necessary but not sufficient step towards functional OOPS has been to modify all IFS routines to work on data passed as arguments. This was done "automatically".
- From cycle 43 onwards, argument passing is implemented for geometry-related data flow field data
- From cycle 45 onwards, model-related information will be passed around.

Spamming

- For cycles ≤ 41 , the IFS stored most of its mutable data in modules, and routines accessed this data implicitly via `USE MODULE, ONLY : DATA`
- A necessary but not sufficient step towards functional OOPS has been to modify all IFS routines to work on data passed as arguments. This was done "automatically".
- From cycle 43 onwards, argument passing is implemented for geometry-related data flow field data
- From cycle 45 onwards, model-related information will be passed around.
- `SUBROUTINE GP_MODEL(YDGEOM,YDFLDS,YDDIMF, YDDPHY, YDSLPHY, YGFL, YDDYN, YDRIP, YDERIP, YDTLSCAW, YDTRSCAW, YDTSCO, YDTCCO, YDCHEM, YDECLD, YDECND, YDCUMFS, YDEGWD, YDEGWWMS, YDRADF, YDERAD, YDENEUR, YDELWRAD, YDEAERD, YDAERD15, YDEAERATM, YDEUVRAD, YDECLDP, YDEWCOU, YDEPHLI, YDPHNC, YDPHLC, YDEAERLID, YDEAERMAP, YDEAERSNK, YDEAERSRC, YDEAERVOL, YDEDBUG, YDEPHY, YDMCC, YDCOM, YDCOU, YDSLREP, YDNCL, YDEGWDWMS, YDSRFTLAD, YDRCOEF, YDTNH, YDWM , YDCDDH, YDLDDH, YDMDDH, YDSDDH, YDTDDH, YDGPDDH, YDPADDH, YDSPDDH, YDPTRSLB1, YDPTRSLB2, YDPTRSLB15, RADGRID, YDERDI, YDCFU, YDXFU, YDSTOPH, YDECUCONVCA, YDCOAPHY, YDTRC, GPHIST, YDSPNG, YDEDYN, YDEGEO, YDEGSL, YDEMP, YDCVMNH, YDPHY, YDPHY0, YDPHY1, YDPHY2, YDPHY3, YDPHYDS, YDTOPH, YDVDOZ, YDSIMPHL, YDARPHY, YDMSE, CDCONF)`

Grouping of modules

An additional level of derived types has been added, to attempt some grouping of concerns :

```
type model_general_conf_type      GEOM, YRDIMF, YGFL, YRRIP
type model_atmos_ocean_coupling_type YRMCC, YRCOM, YRCOU
type model_wave_coupling_typed    YREWCOU
type model_lam_coupling_type      YRELBC0B, YRELBC0C

type model_dynamics_type          YRDYN, YRSPNG, YRPTRGPPC, YYTLSCAW, YYTLSCAWH,
                                  YYTRSCAW,YYTRSCAWH, YYTSCO, YYTCCO, YRSLREP,
                                  YRPTRSLB1,YRPTRSLB2,YRPTRSLB15, YRTNH
type model_physics_general_type   YRDPHY YRSLPHY YRCOAPHY
type model_physics_ecmwf_type     YREPHY YRECLD YRECLDP YRECND YRECUMF YRECUCONV
                                  YREGWD YREGWWMS
type model_physics_simplinear_type YREPHLI,YRCUMFS,YREGWDWMS, YRECUMF2,YRPHLC,
                                   YRPHNC,YRNCL, YRSRFTLAD, GPHIST
```

OOPS planning

Q1-2 2017

- Model field containers
- Model refactoring
- Multi-resolution interpolations
- VarBC
- VarQC
- Singular vectors
- Restart Mechanism
- Integration and testing at all stages

Q3-4 2017

- Optimisation and parallelisation
- IFS QC and data selection screening for OOPS
- Cycling multiple outer loop incremental 4D-Var
- Physics (model, TL, AD)
- TOVSCV
- Observation error covariance
- Fullpos + DDH processing
- Integration and testing at all stages

[1]

[1] For more see https://software.ecmwf.int/wiki/display/OOPS/Project+Documentation

Current status of OOPS

Recent update from Marcin Chrust

- Sqrt-B formulation is working
- Varbc working (with IFS preconditioning)
- tovscv working
- Constraint varbc working
- Second level preconditioning working
- VarQC implemented tests ongoing

Current status of OOPS

Recent update from Marcin Chrust

- Sqrt-B formulation is working
- Varbc working (with IFS preconditioning)
- tovscv working
- Constraint varbc working
- Second level preconditioning working
- VarQC implemented tests ongoing

- Still some problems with all sky observations. Related to the trajectory calculation of the obs op.
- Restart mechanisms need to be discussed with operations.

Current status of OOPS

Recent update from Marcin Chrust

- Sqrt-B formulation is working
- Varbc working (with IFS preconditioning)
- tovscv working
- Constraint varbc working
- Second level preconditioning working
- VarQC implemented tests ongoing

- Still some problems with all sky observations. Related to the trajectory calculation of the obs op.
- Restart mechanisms need to be discussed with operations.

Instead of anchoring channels in constraint VarBC an additional term is added to the cost function

$$J(x, \beta) = \|b(x, \beta)\|_{R_b^{-1}}$$

Where b is the bias in observation space. For OOPS this choice is a bit unfortunate as it requires additional interfaces. Would have been easier to use to existing VarBC code and just set to background of the params to zero for the anchoring channels.

Some issues with the OOPS C++ layer

- Default constructors that don't initialize objects fully
- Single executable for all DA configurations (primal, dual, double, saddle).
- Classification of algorithms based on primal dual double saddle
- Deep inheritance hierachies
- Template on MODEL everywhere
- Modifying argument throught the argument list instead of returning by value
- Absence of move semantics
- Absence of a class for transposed (adjoint) vector

Some issues with the OOPS C++ layer

- Default constructors that don't initialize objects fully
- Single executable for all DA configurations (primal, dual, double, saddle).
- Classification of algorithms based on primal dual double saddle
- Deep inheritance hierachies
- Template on MODEL everywhere
- Modifying argument throught the argument list instead of returning by value
- Absence of move semantics
- Absence of a class for transposed (adjoint) vector

- Minimization algorithms are generic. Perfect!

Matrix free linear algebra

Hessian in OOPS

```
  void multiply(const CtrlInc_ & dx, CtrlInc_ & dz) const {
//  Setup TL terms of cost function
    PostProcessorTL<Increment_> costtl;
    JqTermTL_ * jqtl = j_.jb().initializeTL();
    costtl.enrollProcessor(jqtl);
    unsigned iq = 0;
    if (jqtl) iq = 1;
    for (unsigned jj = 0; jj < j_.nterms(); ++jj) {
      costtl.enrollProcessor(j_.jterm(jj).setupTL(dx));
    }

//  Run TLM
    j_.runTLM(dx, costtl);

//  Finalize Jb+Jq
//  Get TLM outputs, multiply by covariance inverses and setup ADJ forcing terms
    PostProcessorAD<Increment_> costad;
    dz.zero();
    CtrlInc_ dw(j_.jb());

//  Jb
    CtrlInc_ tmp(j_.jb());
    j_.jb().finalizeTL(jqtl, dx, dw);
    j_.jb().multiplyBinv(dw, tmp);
    JqTermAD_ * jqad = j_.jb().initializeAD(dz, tmp);
    costad.enrollProcessor(jqad);
    j_.zeroAD(dw);

//  Jo + Jc
    for (unsigned jj = 0; jj < j_.nterms(); ++jj) {
      boost::scoped_ptr<GeneralizedDepartures> ww(costtl.releaseOutputFromTL(iq+jj));
      boost::shared_ptr<GeneralizedDepartures> zz(j_.jterm(jj).multiplyCoInv(*ww));
      costad.enrollProcessor(j_.jterm(jj).setupAD(zz, dw));
    }

//  Run ADJ
    j_.runADJ(dw, costad);
    dz += dw;
    j_.jb().finalizeAD(jqad);
  }
```

Matrix free linear algebra in oops

IncrementalAssimilation.h

```
HMatrix<MODEL>      H(J);
UMatrix<MODEL>      U(J);       // B = U*U^T
RinvMatrix<MODEL>   Rinv(J);

using namespace mfla; // provides operator*, operator+ and operator~
 // ~U = U^T

auto rhs      = ~U*~H*Rinv*d;
auto Hessian  = I + ~U*~H*Rinv*H*U;

auto dchi   = pcg(Hessian, rhs);
auto dx     = U*dchi;
```

Matrix free linear algebra in oops

IncrementalAssimilation.h

```
HMatrix<MODEL>      H(J);
UMatrix<MODEL>      U(J);       // B = U*U^T
RinvMatrix<MODEL>   Rinv(J);

using namespace mfla; // provides operator*, operator+ and operator~
 // ~U = U^T

auto rhs      = ~U*~H*Rinv*d;
auto Hessian  = I + ~U*~H*Rinv*H*U;

auto dchi   = pcg(Hessian, rhs);
auto dx     = U*dchi;
```

- ~ Swaps the tl with the ad call
- A restructuring of the C++ layer is necessary. In particular the signature of the Matrix `multiply(const X& x,Y& y)` should become `Y multiply(const X& x)`
- Currently `H` maps a `ControlIncrement` to `DualVector` more flexibility is needed here.

Prod.h

```
template<class S, class T>
class Prod {
 private:
  typedef typename std::remove_reference<S>::type::domain_type dom1;
  typedef typename std::remove_reference<T>::type::codomain_type cod2;
  static_assert(std::is_same<dom1, cod2>::value, "domain1 != codomain2");
 public:
  typedef typename std::remove_reference<T>::type::domain_type   domain_type;
  typedef typename std::remove_reference<S>::type::codomain_type codomain_type;

  Prod(S && s, T && t) : _s(std::forward<S>(s)), _t(std::forward<T>(t)) { }
  Prod(const Prod&)              = delete;
  Prod& operator=(const Prod&)   = delete;
  Prod& operator=(Prod&&)        = delete;
  Prod(Prod&& )                  = default;

  void multiply(const domain_type & v, codomain_type & w )  const {
      cod2 t;
     _t.multiply(v,t);
     _s.multiply(t,w);
  }

//  codomain_type operator*(const domain_type & v)  const {return _s*(_t*v); }

 private:
  S _s;
  T _t;
};

// Creator function
template<class S, class T>
Prod<S, T> operator*(S&& s,T&& t) {
  return Prod<S, T>(std::forward<S>(s),std::forward<T>(t));
}
```

Desroziers with VarBC

$$\mathbf{d}_b^a = \mathcal{H}(x^a) + \mathbf{P}\beta^a - \mathcal{H}(x^b) - \mathbf{P}\beta^b$$
$$\mathbf{f}_b^a = \mathcal{H}(\mathbf{x}^a) - \mathcal{H}(\mathbf{x}^b)$$
$$\mathbf{g}_b^a = \mathbf{P}\beta^a - \mathbf{P}\beta^b$$

We get using the assumptions in Desroziers 2005.

$$E\left[\mathbf{d}_b^a \left(\mathbf{d}_b^o\right)^T\right] = \mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{P}\mathbf{B}_\beta\mathbf{P}^T$$
$$E\left[\mathbf{d}_a^o \left(\mathbf{d}_b^o\right)^T\right] = \mathbf{R}$$
$$E\left[\mathbf{d}_b^o \left(\mathbf{d}_b^o\right)^T\right] = \mathbf{R} + \mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{P}\mathbf{B}_\beta\mathbf{P}^T$$
$$E\left[\mathbf{f}_b^a \left(\mathbf{d}_b^o\right)^T\right] = \mathbf{H}\mathbf{B}\mathbf{H}^T$$
$$E\left[\mathbf{g}_b^a \left(\mathbf{d}_b^o\right)^T\right] = \mathbf{P}\mathbf{B}_\beta\mathbf{P}^T$$
$$E\left[\mathbf{d}_b^a \left(\mathbf{d}_a^o\right)^T\right] = \mathbf{H}\mathbf{A}_x\mathbf{H}^T + \mathbf{P}\mathbf{A}_\beta\mathbf{P}^T$$

Where $\mathbf{A}_x$ and $\mathbf{A}_\beta$ are the analysis error covariance matrices for the state and the VarBC parameters respectively (See the next section).

Analysis errors for state and params

$$\mathbf{A}_x = E\left[\epsilon_x^a \left(\epsilon_x^a\right)^T\right] = \mathbf{B} - \mathbf{B}\mathbf{H}^T(\mathbf{R} + \mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{P}\mathbf{B}_\beta\mathbf{P}^T)^{-1}\mathbf{H}\mathbf{B} \tag{1}$$

$$= \left(\mathbf{B}^{-1} + \mathbf{H}^T(\mathbf{R} + \mathbf{P}\mathbf{B}_\beta^{-1}\mathbf{P}^T)^{-1}\mathbf{H}\right)^{-1} \tag{2}$$

$$\mathbf{A}_\beta = E[\epsilon_\beta^a(\epsilon_\beta^a)^T] = \mathbf{B}_\beta - \mathbf{B}_\beta\mathbf{P}^T\left(\mathbf{R} + \mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{P}\mathbf{B}_\beta\mathbf{P}^T\right)^{-1}\mathbf{P}\mathbf{B}_\beta \tag{3}$$

$$= \left(\mathbf{B}_\beta^{-1} + \mathbf{P}^T\left(\mathbf{R} + \mathbf{H}\mathbf{B}\mathbf{H}^T\right)^{-1}\mathbf{P}\right)^{-1} \tag{4}$$

Using the assumption that the propagator for the params is the identity this could be used to implement a Kalman filter for the params.