Modernisation of the observation interpolation operator (GOM)...

... and thoughts on the quality of code in the IFS

Alan Geer, John Hague, Deborah Salmond

Thanks to: Mats Hamrud, Tomas Wilhelmsson, Karim Yessad

What is the real output of RD?

Scientific understanding?

Papers and memos?

Better forecasts?

The IFS

→ Fortran code

The IFS Fortran code…

… is often very hard to understand

… takes people years to learn

… is difficult to maintain

… is not modular

… is often extremely verbose and hand-coded

… requires modifications in hundreds of places when a new feature is added …

… and that new feature will break something else

What do we do all day?

How much time do we spend on new science?

And how much on:

writing code

fighting bugs

urgent code maintenance

trying to understand how the IFS works?

We make Fortran code and we spend a lot of time on it

We should give more priority and thought to the quality of our code

Better code now means more science in the long run

How can we improve things?
(when every new feature makes the code more complex)

Code tidying

Local rewrites and modernisation

Bigger areas of completely new code

OOPS

COPE

There are already some areas of clean, modern, modular code in the IFS

mpl_send, mpl_recv etc.

trans modules

but you'll probably not have noticed them, as they don't break or need modifications very often

Outline

GOM

Interpolates model fields to observation locations

Distributes model fields to the correct processor for the observations

A notoriously complex and difficult piece of code, even by the standards of the IFS

New GOM

An example of code modernisation using a modular approach

How to use the new GOMs

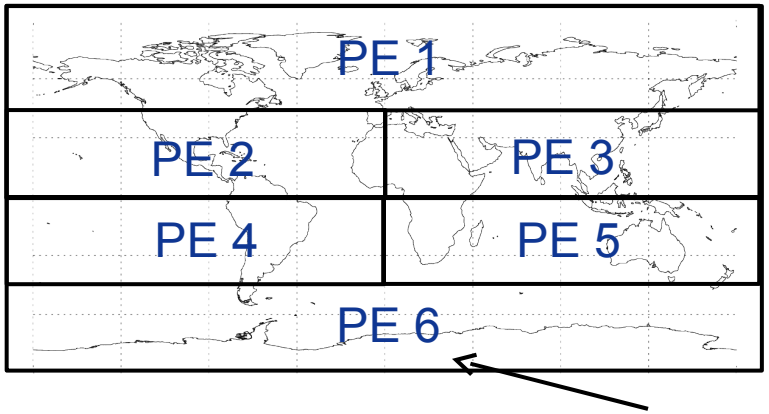
Thoughts on improving the quality of code in the IFS more generally

Ideas for representing the model state

Observation operator needs a general rewrite (OOPS and COPE?)

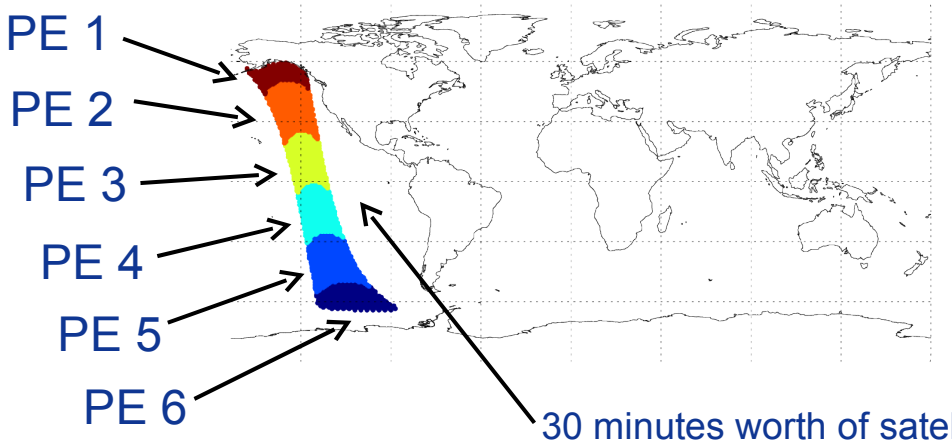
Load balancing in the IFS:
Sharing the effort equally between processes

Model space



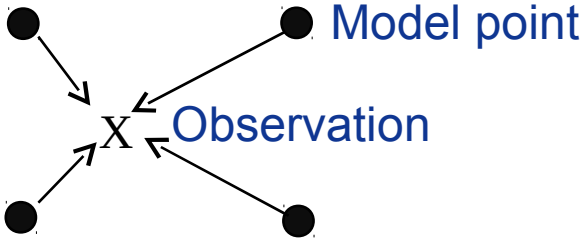
In the reduced Gaussian grid, there are fewer model points near the poles

Observation space



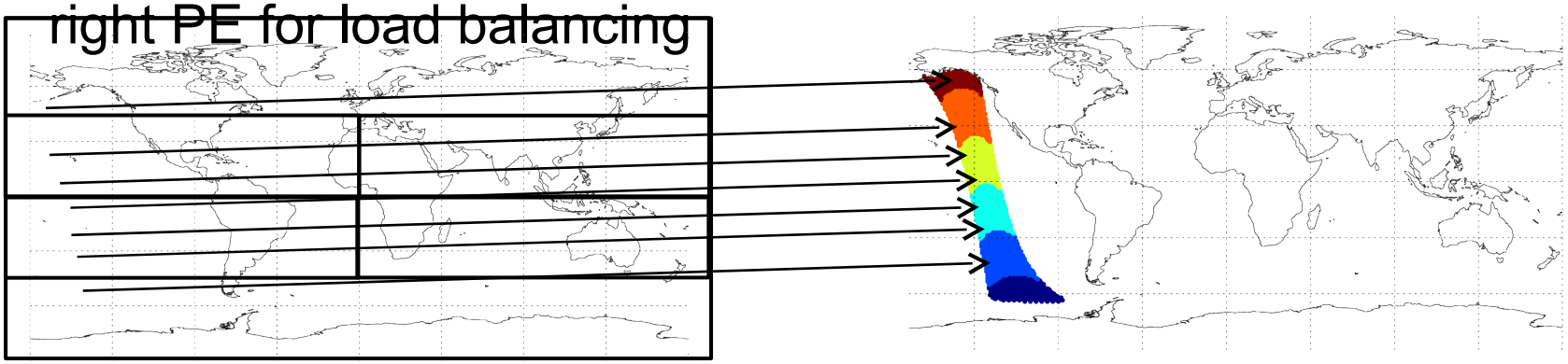
Interpolation operator

1. Horizontal interpolation to observation location



Current options:
- Bilinear 4-point
- 16-point
- Nearest-neighbour

2. Send the interpolated columns to the right PE for load balancing



That doesn't sound impossible. Where's the complexity?

Tangent linear and adjoint versions

Multiply the amount of code required by 3

Bit reproducibility of the adjoint sum

Special message passing for the adjoint

'2D' GOMs (level, grid point)

2D cross section of the atmosphere for limb-sounding

"Block" GOMs (level, variable)

GEMs reactive gases, greenhouse gases, aerosols

Neutral winds

Scatterometer special cases

Land-sea mask dependent interpolation (Meteo-France)

Two different flavours (each with completely separate code)

All-sky microwave imagers

Use callpar diagnostic quantities, available from a different part of the timestep

Current state of the GOM code

In the IFS this needs well over 10,000 lines of code

These 10,000 lines are nearly impossible to understand:

Variables change their name at every stage, and are stored in a variety of different buffers, structures, etc…

Some variables appear and disappear magically, or even briefly share the same name, only to be separated later…

Hundreds of “if” statements for special cases

Copy and paste coding

New functionality, e.g. ‘2D’ GOMs, has added a whole new parallel set of subroutines

Code that was clean 20 years ago is now almost unmaintainable

Old GOM routines (excluding tl and ad)

Description	Subroutines	Data structures	How is the temperature profile accessed?
Mapping for message passing	mkglobstab	glob	
Set up the GOM data structures	sugoms, goms_mix	ygomv, yobbuf1	lgomua(:,ydgom%ygomv%mt)
Copy variables out of model space	cobs	gfl, gmv, gmvs, psp_sb …	pgmv(:,:,yt0%mt) **1**
Put them into a buffer	cobs	pb1	pb1(:,ydobb1%mptrt+jlev-1) **2**
Fill the “SL halo”	slcomm, slextpol	pb1	
Management	obshor	obshor_cache	
Interpolate to observation locations	slint	pbf	pbf(:,ydobb1%mptrt+jlev-1)
Pack & send buffer	mpobseq_pack	pbufs	pbufs(kbp+1:kbp+nflevg) **3**
Message passing, receive buffer	mpobseq	zbufr	zbufr(kbp+1:kbp+nflevg)
Unpack buffer into GOMs	mpobseq	ydgom	ydgom%yua(inum)%t(:) **4**
Unpack the GOMs for use in hop “packets”	preint, preints	ztf5, zuf5, zvf5 ….	ztf5 **5**

The knock-on effects of difficult code:

If it's too hard to add a new GOM variable, e.g. rain flux…

…we attempt to run the observation operator from inside the model instead!

Microwave imagers in rain and cloud: 1D+4D-Var, all-sky

SMOS monitoring

Rain-radar and rain gauges

A whole new parallel infrastructure for observations: "easier" than touching the GOMs.

Microwave imager ("all-sky") observations were moved out of the model and into the observation operator for 38r1

A difficult 6-month job

A merge problem with OOPS cleaning led to a dangerous bug in the 38r1 esuite

Previous GOM cleaning:

Karim's GOM cleaning document: slint and surface fields improved

Unification of many separate areas into "cobsall" and "goms_mix" module - a good start

John spent a month just trying to clean up the treatment of surface variables, following Karim's suggestions

But there was still work to do on the "2D" GOMs

A particular difficulty was a flag used by neutral winds that turned from a logical into a number half-way through the processing chain

The result was only slightly more maintainable, but required more lines of code

Such piecemeal cleaning can often be slow, hard, and doesn't really fix the underlying problems

A small example from mpobseqad_unpack.F90

```
IF (ITYP == NSATEM .OR. ITYP == NALLSKY) THEN
  IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MU)) &
     & PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1)=PBUFR(KBP+1)
  IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MV)) &
     & PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1)=PBUFR(KBP+2)
  IF(YDOBB1_TLAD%LINTSP)PBF(IOBS,YDOBB1_TLAD%MPTRSP)=PBUFR(KBP+3)
  IF(YDOBB1_TLAD%LINTRR_T)PBF(IOBS,YDOBB1_TLAD%MPTRRR_T) = PBUFR(KBP+4)
  IF (YDOBB1_TLAD%LINTSB_Q) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRSB_Q) = PBUFR(KBP+5)
  ELSEIF (YDOBB1_TLAD%LINTRR_W) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRRR_W) = PBUFR(KBP+5)
  ENDIF
  IF(YDOBB1_TLAD%LINTSG_F)PBF(IOBS,YDOBB1_TLAD%MPTRSG_F) = PBUFR(KBP+6)
  IF(YDOBB1_TLAD%LINTVF_ZOF)PBF(IOBS,YDOBB1_TLAD%MPTRVF_ZOF) = PBUFR(KBP+7)
  IF(YDOBB1_TLAD%LINTWL)PBF(IOBS,YDOBB1_TLAD%MPTRWL) = PBUFR(KBP+8)
  IF(YDOBB1_TLAD%LINTVD_10NU)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU)= PBUFR(KBP+11)
  IF(YDOBB1_TLAD%LINTVD_10NV)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV)= PBUFR(KBP+12)
ELSEIF (ITYP == NSCATT) THEN
  !            Copy model variables for Scatterometer obs operators
  IF(YDOBB1_TLAD%LINTU)PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1)=PBUFR(KBP+1)
  IF(YDOBB1_TLAD%LINTV)PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1)=PBUFR(KBP+2)
  IF(YDOBB1_TLAD%LINTSP)PBF(IOBS,YDOBB1_TLAD%MPTRSP)=PBUFR(KBP+3)
  IF(YDOBB1_TLAD%LINTRR_T)PBF(IOBS,YDOBB1_TLAD%MPTRRR_T) = PBUFR(KBP+4)
  IF(YDOBB1_TLAD%LINTVF_ZOF)PBF(IOBS,YDOBB1_TLAD%MPTRVF_ZOF) = PBUFR(KBP+7)
  IF(YDOBB1_TLAD%LINTT)PBF(IOBS,YDOBB1_TLAD%MPTRT+NFLEVG-1)=PBUFR(KBP+9)
  IIND=YDOBB1_TLAD%MPTRGFL+NFLEVG*(YDGOM%MGOMGFL(YDGOM%YGOMV%MQ)-1)
  IF(YDOBB1_TLAD%LINTGFL)PBF(IOBS,IIND+NFLEVG-1) = PBUFR(KBP+10)
  IF(YDOBB1_TLAD%LINTVD_10NU)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU)= PBUFR(KBP+11)
  IF(YDOBB1_TLAD%LINTVD_10NV)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV)= PBUFR(KBP+12)
ELSEIF (ITYP == NSYNOP .OR. ITYP == NAIREP .OR.&
 & ITYP == NSATOB .OR. ITYP == NDRIBU .OR.&
 & ITYP == NTEMP  .OR. ITYP == NPILOT .OR.&
 & ITYP == NPAOB  .OR. ITYP == NLIMB  .OR.&
 & ITYP == NRADAR .OR. ITYP == NGBRAD .OR.&
 & ITYP == NISAC  .OR.&
 & ITYP == NLIDAR) THEN

  !            Copy model variables for conventional obs operators

  PBF(IOBS,YDOBB1_TLAD%MPTRSP)=PBUFR(KBP+3)
  IF(YDOBB1_TLAD%LINTRR_T)PBF(IOBS,YDOBB1_TLAD%MPTRRR_T) = PBUFR(KBP+4)
  IF (YDOBB1_TLAD%LINTSB_Q) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRSB_Q) = PBUFR(KBP+5)
  ELSEIF (YDOBB1_TLAD%LINTRR_W) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRRR_W) = PBUFR(KBP+5)
  ENDIF
  IF(YDOBB1_TLAD%LINTSG_F)PBF(IOBS,YDOBB1_TLAD%MPTRSG_F) = PBUFR(KBP+6)
  IF(YDOBB1_TLAD%LINTVF_ZOF)PBF(IOBS,YDOBB1_TLAD%MPTRVF_ZOF) = PBUFR(KBP+7)
  IF(YDOBB1_TLAD%LINTWL)PBF(IOBS,YDOBB1_TLAD%MPTRWL) = PBUFR(KBP+8)
  IF(YDOBB1_TLAD%LINTVD_10NU)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU)= PBUFR(KBP+11)
  IF(YDOBB1_TLAD%LINTVD_10NV)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV)= PBUFR(KBP+12)
ELSE
  WRITE(NULERR,'(A,3I8)') ' MPOBSEQAD_UNPCK:UNKNOWN OBS TYPE',&
   & JOBS,IOBS,ITYP
  CALL ABOR1('MPOBSEQAD_UNPCK:UNKNOWN OBS TYPE')
ENDIF
```

A small example from mpobseqad_unpack.F90

```
IF (ITYP == NSATEM .OR. ITYP == NALLSKY) THEN

ELSEIF (ITYP == NSCATT) THEN
    !            Copy model variables for Scatterometer obs operators

ELSEIF (ITYP == NSYNOP .OR. ITYP == NAIREP .OR.&
   & ITYP == NSATOB .OR. ITYP == NDRIBU .OR.&
   & ITYP == NTEMP  .OR. ITYP == NPILOT .OR.&
   & ITYP == NPAOB  .OR. ITYP == NLIMB  .OR.&
   & ITYP == NRADAR .OR. ITYP == NGBRAD .OR.&
   & ITYP == NISAC  .OR.&
   & ITYP == NLIDAR) THEN

    !            Copy model variables for conventional obs operators


ELSE
    WRITE(NULERR,'(A,3I8)') ' MPOBSEQAD_UNPCK:UNKNOWN OBS TYPE',&
     & JOBS,IOBS,ITYP
    CALL ABOR1('MPOBSEQAD_UNPCK:UNKNOWN OBS TYPE')
ENDIF
```

A small example from mpobseqad_unpack.F90

```
IF (ITYP == NSATEM .OR. ITYP == NALLSKY) THEN

ELSEIF (ITYP == NSCATT) THEN
    !            Copy model variables for Scatterometer obs operators

ELSE
    !            Sensible default behaviour: as conventional observations


ENDIF
```

A small example from mpobseqad_unpack.F90

```
IF (ITYP == NSATEM .OR. ITYP == NALLSKY) THEN
  IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MU)) &
     & PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1)=PBUFR(KBP+1)
  IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MV)) &
     & PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1)=PBUFR(KBP+2)
  IF(YDOBB1_TLAD%LINTSP)PBF(IOBS,YDOBB1_TLAD%MPTRSP)=PBUFR(KBP+3)
  IF(YDOBB1_TLAD%LINTRR_T)PBF(IOBS,YDOBB1_TLAD%MPTRRR_T) = PBUFR(KBP+4)
  IF (YDOBB1_TLAD%LINTSB_Q) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRSB_Q) = PBUFR(KBP+5)
  ELSEIF (YDOBB1_TLAD%LINTRR_W) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRRR_W) = PBUFR(KBP+5)
  ENDIF
  IF(YDOBB1_TLAD%LINTSG_F)PBF(IOBS,YDOBB1_TLAD%MPTRSG_F) = PBUFR(KBP+6)
  IF(YDOBB1_TLAD%LINTVF_Z0F)PBF(IOBS,YDOBB1_TLAD%MPTRVF_Z0F) = PBUFR(KBP+7)
  IF(YDOBB1_TLAD%LINTWL)PBF(IOBS,YDOBB1_TLAD%MPTRWL) = PBUFR(KBP+8)
  IF(YDOBB1_TLAD%LINTVD_10NU)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU)= PBUFR(KBP+11)
  IF(YDOBB1_TLAD%LINTVD_10NV)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV)= PBUFR(KBP+12)
ELSEIF (ITYP == NSCATT) THEN
  !            Copy model variables for Scatterometer obs operators
  IF(YDOBB1_TLAD%LINTU)PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1)=PBUFR(KBP+1)
  IF(YDOBB1_TLAD%LINTV)PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1)=PBUFR(KBP+2)
  IF(YDOBB1_TLAD%LINTSP)PBF(IOBS,YDOBB1_TLAD%MPTRSP)=PBUFR(KBP+3)
  IF(YDOBB1_TLAD%LINTRR_T)PBF(IOBS,YDOBB1_TLAD%MPTRRR_T) = PBUFR(KBP+4)
  IF(YDOBB1_TLAD%LINTVF_Z0F)PBF(IOBS,YDOBB1_TLAD%MPTRVF_Z0F) = PBUFR(KBP+7)
  IF(YDOBB1_TLAD%LINTT)PBF(IOBS,YDOBB1_TLAD%MPTRT+NFLEVG-1)=PBUFR(KBP+9)
  IIND=YDOBB1_TLAD%MPTRGFL+NFLEVG*(YDGOM%MGOMGFL(YDGOM%YGOMV%MQ)-1)
  IF(YDOBB1_TLAD%LINTGFL)PBF(IOBS,IIND+NFLEVG-1) = PBUFR(KBP+10)
  IF(YDOBB1_TLAD%LINTVD_10NU)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU)= PBUFR(KBP+11)
  IF(YDOBB1_TLAD%LINTVD_10NV)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV)= PBUFR(KBP+12)
ELSE
  !            Copy model variables for conventional obs operators
  PBF(IOBS,YDOBB1_TLAD%MPTRSP)=PBUFR(KBP+3)
  IF(YDOBB1_TLAD%LINTRR_T)PBF(IOBS,YDOBB1_TLAD%MPTRRR_T) = PBUFR(KBP+4)
  IF (YDOBB1_TLAD%LINTSB_Q) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRSB_Q) = PBUFR(KBP+5)
  ELSEIF (YDOBB1_TLAD%LINTRR_W) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRRR_W) = PBUFR(KBP+5)
  ENDIF
  IF(YDOBB1_TLAD%LINTSG_F)PBF(IOBS,YDOBB1_TLAD%MPTRSG_F) = PBUFR(KBP+6)
  IF(YDOBB1_TLAD%LINTVF_Z0F)PBF(IOBS,YDOBB1_TLAD%MPTRVF_Z0F) = PBUFR(KBP+7)
  IF(YDOBB1_TLAD%LINTWL)PBF(IOBS,YDOBB1_TLAD%MPTRWL) = PBUFR(KBP+8)
  IF(YDOBB1_TLAD%LINTVD_10NU)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU)= PBUFR(KBP+11)
  IF(YDOBB1_TLAD%LINTVD_10NV)PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV)= PBUFR(KBP+12)
ENDIF
```

A small example from mpobseqad_unpack.F90

```
IF(YDOBB1_TLAD%LINTSP)      PBF(IOBS,YDOBB1_TLAD%MPTRSP)      = PBUFR(KBP+3)
IF(YDOBB1_TLAD%LINTRR_T)    PBF(IOBS,YDOBB1_TLAD%MPTRRR_T)    = PBUFR(KBP+4)
IF(YDOBB1_TLAD%LINTVF_Z0F)  PBF(IOBS,YDOBB1_TLAD%MPTRVF_Z0F)  = PBUFR(KBP+7)
IF(YDOBB1_TLAD%LINTVD_10NU) PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU) = PBUFR(KBP+11)
IF(YDOBB1_TLAD%LINTVD_10NV) PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV) = PBUFR(KBP+12)

IF (ITYP == NSATEM .OR. ITYP == NALLSKY) THEN

  IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MU)) &
     & PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1)=PBUFR(KBP+1)
  IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MV)) &
     & PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1)=PBUFR(KBP+2)
  IF (YDOBB1_TLAD%LINTSB_Q) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRSB_Q) = PBUFR(KBP+5)
  ELSEIF (YDOBB1_TLAD%LINTRR_W) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRRR_W) = PBUFR(KBP+5)
  ENDIF
  IF(YDOBB1_TLAD%LINTSG_F)PBF(IOBS,YDOBB1_TLAD%MPTRSG_F) = PBUFR(KBP+6)
  IF(YDOBB1_TLAD%LINTWL)PBF(IOBS,YDOBB1_TLAD%MPTRWL) = PBUFR(KBP+8)

ELSEIF (ITYP == NSCATT) THEN

  IF(YDOBB1_TLAD%LINTU)PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1)=PBUFR(KBP+1)
  IF(YDOBB1_TLAD%LINTV)PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1)=PBUFR(KBP+2)

  IF(YDOBB1_TLAD%LINTT)PBF(IOBS,YDOBB1_TLAD%MPTRT+NFLEVG-1)=PBUFR(KBP+9)
  IIND=YDOBB1_TLAD%MPTRGFL+NFLEVG*(YDGOM%MGOMGFL(YDGOM%YGOMV%MQ)-1)
  IF(YDOBB1_TLAD%LINTGFL)PBF(IOBS,IIND+NFLEVG-1) = PBUFR(KBP+10)

ELSE

  IF (YDOBB1_TLAD%LINTSB_Q) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRSB_Q) = PBUFR(KBP+5)
  ELSEIF (YDOBB1_TLAD%LINTRR_W) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRRR_W) = PBUFR(KBP+5)
  ENDIF
  IF(YDOBB1_TLAD%LINTSG_F)PBF(IOBS,YDOBB1_TLAD%MPTRSG_F) = PBUFR(KBP+6)
  IF(YDOBB1_TLAD%LINTWL)PBF(IOBS,YDOBB1_TLAD%MPTRWL) = PBUFR(KBP+8)

ENDIF
```

A small example from mpobseqad_unpack.F90

```
IF(YDOBB1_TLAD%LINTSP)       PBF(IOBS,YDOBB1_TLAD%MPTRSP)       = PBUFR(KBP+3)
IF(YDOBB1_TLAD%LINTRR_T)     PBF(IOBS,YDOBB1_TLAD%MPTRRR_T)     = PBUFR(KBP+4)
IF(YDOBB1_TLAD%LINTVF_Z0F)   PBF(IOBS,YDOBB1_TLAD%MPTRVF_Z0F)   = PBUFR(KBP+7)
IF(YDOBB1_TLAD%LINTVD_10NU) PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU) = PBUFR(KBP+11)
IF(YDOBB1_TLAD%LINTVD_10NV) PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV) = PBUFR(KBP+12)

IF (ITYP /= NSCATT) THEN

  IF (YDOBB1_TLAD%LINTSB_Q) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRSB_Q) = PBUFR(KBP+5)
  ELSEIF (YDOBB1_TLAD%LINTRR_W) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRRR_W) = PBUFR(KBP+5)
  ENDIF
  IF(YDOBB1_TLAD%LINTSG_F)PBF(IOBS,YDOBB1_TLAD%MPTRSG_F) = PBUFR(KBP+6)
  IF(YDOBB1_TLAD%LINTWL)PBF(IOBS,YDOBB1_TLAD%MPTRWL) = PBUFR(KBP+8)

ENDIF

IF (ITYP == NSATEM .OR. ITYP == NALLSKY) THEN

  IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MU)) &
     & PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1)=PBUFR(KBP+1)
  IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MV)) &
     & PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1)=PBUFR(KBP+2)

ELSEIF (ITYP == NSCATT) THEN

  IF(YDOBB1_TLAD%LINTU)PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1)=PBUFR(KBP+1)
  IF(YDOBB1_TLAD%LINTV)PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1)=PBUFR(KBP+2)

  IF(YDOBB1_TLAD%LINTT)PBF(IOBS,YDOBB1_TLAD%MPTRT+NFLEVG-1)=PBUFR(KBP+9)
  IIND=YDOBB1_TLAD%MPTRGFL+NFLEVG*(YDGOM%MGOMGFL(YDGOM%YGOMV%MQ)-1)
  IF(YDOBB1_TLAD%LINTGFL)PBF(IOBS,IIND+NFLEVG-1) = PBUFR(KBP+10)

ENDIF
```

A small example from mpobseqad_unpack.F90

```
IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MU)) &
  &                          PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1) = PBUFR(KBP+1)
IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MV)) &
  &                          PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1) = PBUFR(KBP+2)

IF(YDOBB1_TLAD%LINTSP)       PBF(IOBS,YDOBB1_TLAD%MPTRSP)          = PBUFR(KBP+3)
IF(YDOBB1_TLAD%LINTRR_T)     PBF(IOBS,YDOBB1_TLAD%MPTRRR_T)        = PBUFR(KBP+4)
IF(YDOBB1_TLAD%LINTVF_Z0F)   PBF(IOBS,YDOBB1_TLAD%MPTRVF_Z0F)      = PBUFR(KBP+7)
IF(YDOBB1_TLAD%LINTVD_10NU) PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU)     = PBUFR(KBP+11)
IF(YDOBB1_TLAD%LINTVD_10NV) PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV)     = PBUFR(KBP+12)

IF (ITYP /= NSCATT) THEN

  IF (YDOBB1_TLAD%LINTSB_Q) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRSB_Q) = PBUFR(KBP+5)
  ELSEIF (YDOBB1_TLAD%LINTRR_W) THEN
    PBF(IOBS,YDOBB1_TLAD%MPTRRR_W) = PBUFR(KBP+5)
  ENDIF
  IF(YDOBB1_TLAD%LINTSG_F)PBF(IOBS,YDOBB1_TLAD%MPTRSG_F) = PBUFR(KBP+6)
  IF(YDOBB1_TLAD%LINTWL)PBF(IOBS,YDOBB1_TLAD%MPTRWL) = PBUFR(KBP+8)

ELSE

  IF(YDOBB1_TLAD%LINTT)PBF(IOBS,YDOBB1_TLAD%MPTRT+NFLEVG-1)=PBUFR(KBP+9)
  IIND=YDOBB1_TLAD%MPTRGFL+NFLEVG*(YDGOM%MGOMGFL(YDGOM%YGOMV%MQ)-1)
  IF(YDOBB1_TLAD%LINTGFL)PBF(IOBS,IIND+NFLEVG-1) = PBUFR(KBP+10)

ENDIF
```

A small example from mpobseqad_unpack.F90

```
IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MU)) &
  &                           PBF(IOBS,YDOBB1_TLAD%MPTRU+NFLEVG-1) = PBUFR(KBP+1)
IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MV)) &
  &                           PBF(IOBS,YDOBB1_TLAD%MPTRV+NFLEVG-1) = PBUFR(KBP+2)
IF(YDOBB1_TLAD%LINTSP)        PBF(IOBS,YDOBB1_TLAD%MPTRSP)         = PBUFR(KBP+3)
IF(YDOBB1_TLAD%LINTRR_T)      PBF(IOBS,YDOBB1_TLAD%MPTRRR_T)       = PBUFR(KBP+4)

IF (ITYP /= NSCATT) THEN
  IF (YDOBB1_TLAD%LINTSB_Q) THEN
                              PBF(IOBS,YDOBB1_TLAD%MPTRSB_Q)       = PBUFR(KBP+5)
  ELSEIF (YDOBB1_TLAD%LINTRR_W) THEN
                              PBF(IOBS,YDOBB1_TLAD%MPTRRR_W)       = PBUFR(KBP+5)
  ENDIF
  IF(YDOBB1_TLAD%LINTSG_F)    PBF(IOBS,YDOBB1_TLAD%MPTRSG_F)       = PBUFR(KBP+6)
  IF(YDOBB1_TLAD%LINTWL)      PBF(IOBS,YDOBB1_TLAD%MPTRWL)         = PBUFR(KBP+8)
ENDIF

IF(YDOBB1_TLAD%LINTVF_Z0F)    PBF(IOBS,YDOBB1_TLAD%MPTRVF_Z0F)     = PBUFR(KBP+7)
IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MT)) &
  &                           PBF(IOBS,YDOBB1_TLAD%MPTRT+NFLEVG-1) = PBUFR(KBP+9)
IIND=YDOBB1_TLAD%MPTRGFL+NFLEVG*(YDGOM%MGOMGFL(YDGOM%YGOMV%MQ)-1)
IF(.NOT.YDGOM%LGOMUA(ITYP,YDGOM%YGOMV%MQ)) &
  &                           PBF(IOBS,IIND+NFLEVG-1)              = PBUFR(KBP+10)
IF(YDOBB1_TLAD%LINTVD_10NU)   PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NU)    = PBUFR(KBP+11)
IF(YDOBB1_TLAD%LINTVD_10NV)   PBF(IOBS,YDOBB1_TLAD%MPTRVD_10NV)    = PBUFR(KBP+12)
```

A small example from mpobseqad_unpack.F90

Every line looks a bit like this:

```
IF(YDOBB1_TLAD%LINTRR_T)     PBF(IOBS,YDOBB1_TLAD
%MPTRRR_T) = PBUFR(KBP+4)
```

A small example from mpobseqad_unpack.F90

With a more generic approach, we could use a simple loop:

```
  for each GOM variable do begin
    if interpolation is required then

      pbf(this variable) = pbufr(this variable)

    endif
  enddo
```

```
43 lines        →   5 lines
6 point font    →   20 point font
Impossible  →   understandable
```

Great, only 9957 lines of code still to go!

Really, piecemeal cleaning is not a viable option

How to clean the GOMs then?

Throw away those 10,000 lines of old code

Imagine how things should really be

Make a prototype outside the IFS

Create a “test harness” for GOM that simulates (in a simple, idealised way) the IFS inputs, calls, and outputs

Write a prototype module for the GOMs

Quickly explore new ideas, test, debug

Each iteration of the debug/learning cycle takes minutes, not hours

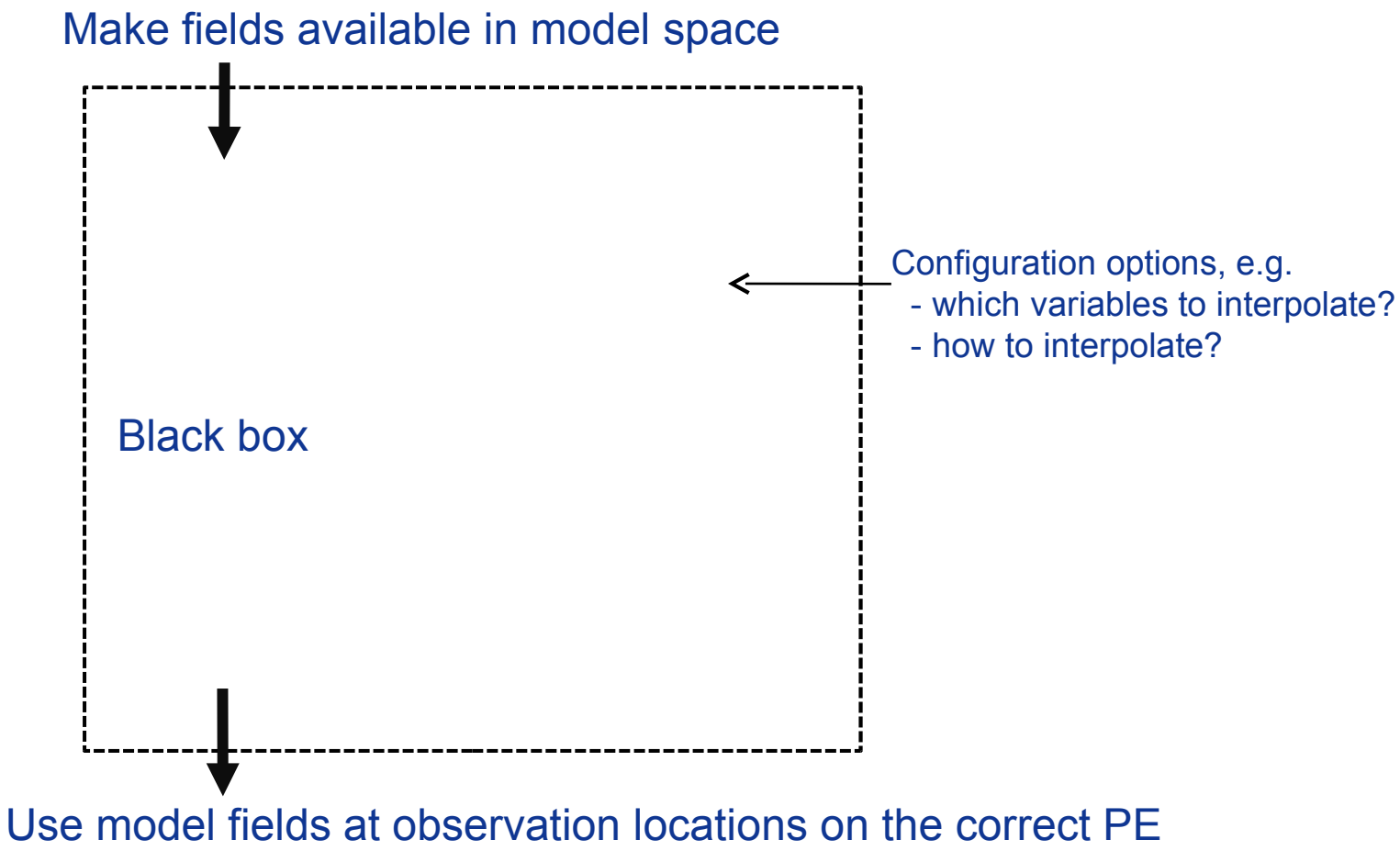
Result is “new_gom.F90” – a few hundred line of code, easy for people to review, make suggestions, etc.

The prototype becomes the basis of a new module for the IFS: “gom_mod.F90”

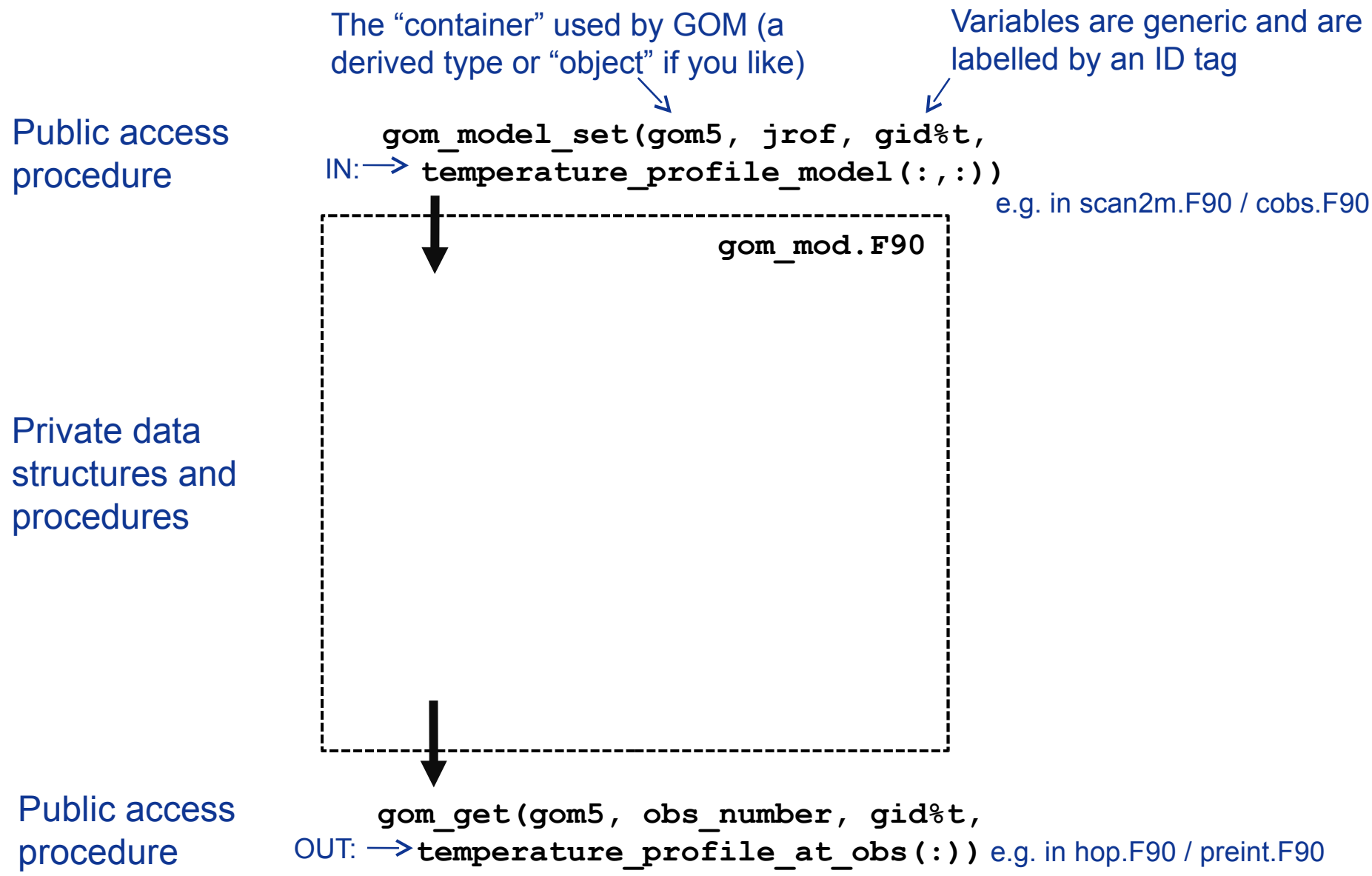
Time spent on new GOMs

Task	Time
Understand how the old GOMs work	6 months
Plan, think, discuss the new approach	7 days
Make the prototype	3 days
Integrate the prototype into the direct code	1 month
Extend to TL and adjoint (John)	1 month
Make it ready for 38r2: - keep on cleaning and improving the new code - find the remaining bugs - test to operational standards - performance tuning	2 months
Reintroduce “2D” GOMs and land-sea mask dependent interpolation	ongoing

New GOM overview



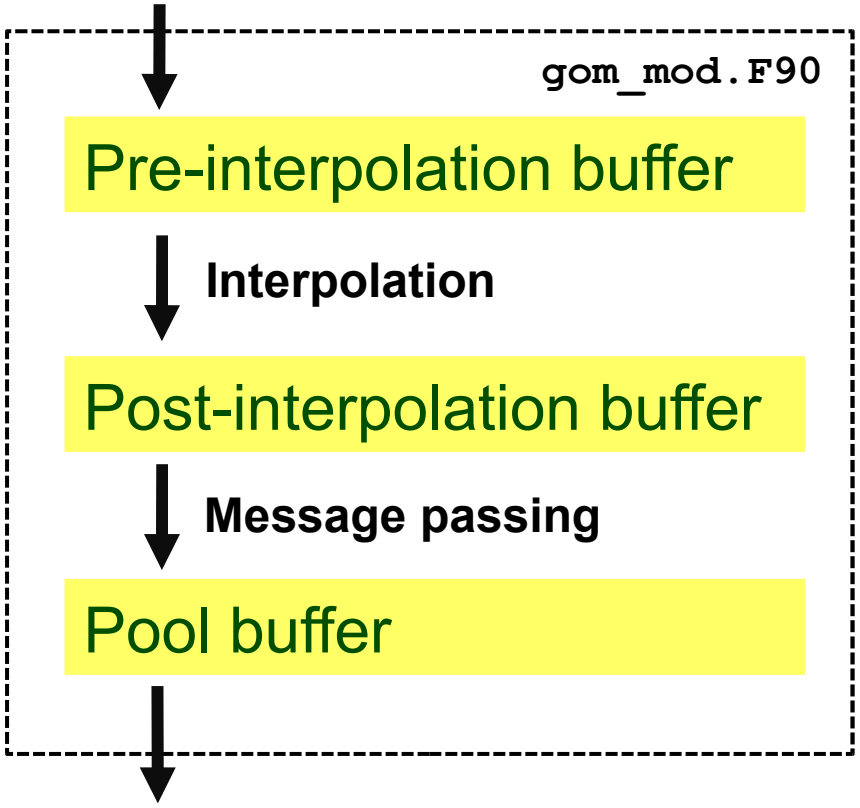
New GOM overview



New GOM overview

Public access procedure

```
gom_model_set(gom5, jrof, gid%t,
  temperature_profile_in_model(:))
```



Fields in model space

Private data structures and procedures

Interpolated to observation locations, but on wrong PE

Model fields at observation locations on the correct PE

Public access procedure

```
gom_get(gom5, obs_number, gid%t,
  temperature_profile_at_obs(:))
```

How to add a new GOM variable – part 1/3

1) Increase the number of GOM variables appropriately:

```
integer(kind=jpim), parameter :: ngomvar = 64 ! Number of GOM variables (GOM**1)
```

2) Give an ID name to the new variable (which should be consistent with naming conventions in the model world):

```
! One ID name for each variable in the GOMS. (GOM**2)
type type_gom_id
  integer(kind=jpim) :: u, v, t, q, o3, l, i, a, ghg, grg, aero, phys, s, r, g, &
    & ul, vl, tl, ql, sp, orog, cori, gnordl, gnordm, sb_q, sg_f, rr_t, rr_w, rr_ic, cl_tcls, &
    & cl_hucls, x2_prwa, x2_prsn, vf_z0f, vf_albf, vf_emisf, vf_lsm, vf_veg, vf_cvl, &
    & vf_cvh, vf_tvl, vf_tvh, vf_ci, vf_ucur, vf_vcur, vv_arg, vv_sab, vv_hv, vv_z0h, &
    & ws_char, vd_10nu, vd_10nv, vd_upd, vd_z0f, vx_oro, vx_tsc, vx_sno, est, esn, et2, eh2, &
    & ev1, ez, eh
end type
```

3) Add ID numbers in sequence:

```
! (GOM**3) ID pointers for the GOM variables (F95 allows default initialisations
! in the type definition above, e.g. u=1, v=2 etc. which would be better)
type(type_gom_id), parameter :: gid=type_gom_id(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,&
  & 16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,&
  & 43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64)
```

How to add a new GOM variable – part 2/3

4) Define the ID name as a string, and specify the number of dimensions:

```
! GOM**4 GOM names and number of dimensions
! NOTE: due to Fortran 90 rules, each string here must be padded to 10 characters.
type(type_gvar), parameter :: gvar(ngomvar) = (/ &
  ! GMV
  & type_gvar('u         ',2), & ! Horizontal wind component
  & type_gvar('v         ',2), & ! Horizontal wind component
  & type_gvar('t         ',2), & ! Temperature
  ! GFL
  & type_gvar('q         ',2), & ! Specific humidity
```

5) In model space, copy the variable (e.g. from GMVS, GFL arrays) into the GOMs (in cobs.F90):

```
IF (GOM%PRE%ON(GID%T)) CALL GOM_MODEL_SET(GOM, IROF, GID%T, PGMV(JROF,:,YT0%MT))
```

6) In observation space, get the variable from the GOMs (e.g. in preint.F90):

```
LLGOT = GOM_GET(GOM, IPOSM(JOBS), GID%T, PTF5(JOBS,1:NFLEVG))
```

How to add a new GOM variable – part 3/3

7) Turn on interpolation for the relevant observation types (sugoms.F90) – default is everything OFF:

```
GOM%LINTERP(:      ,GID%T) = (NFTHER /= 0)
GOM%LINTERP(NSCATT ,GID%T) = .FALSE.
IF (LDOBSTL) GOM%LINTERP(NSATOB ,GID%T) = .FALSE. ! Not needed in TL/AD
IF (LDOBSTL) GOM%LINTERP(NPILOT ,GID%T) = .FALSE. ! Not needed in TL/AD
```

8) Define the interpolation method (sugoms.F90), selecting from

- default (either bilinear_4pt or bilinear_16pt according to NOBSHOR)
- bilinear_4pt,
- bilinear_16pt
- nearest_neighbour

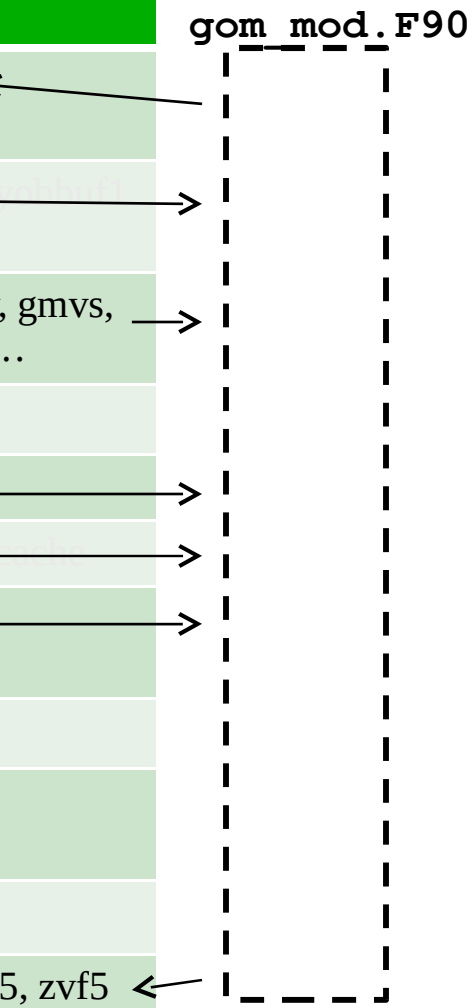
```
GOM%INTERP_TYPE(:,GID%T)    = GINTERP%DEFAULT
GOM%INTERP_TYPE(NALLSKY,:) = GINTERP%NEAREST_NEIGHBOUR
```

Old GOMs → New GOMs

Description	Subroutines	Data structures
Mapping for message passing	mkglobstab	glob
Set up the GOM data structures	sugoms, goms_mix	ygomv, yobbuf1
Copy variables out of model space	cobs	gfl, gmv, gmvs, psp_sb …
Put them into a buffer	cobs	pb1
Fill the “SL halo”	slcomm, slextpol	pb1
Management	obshor	obshor_cache
Interpolate to observation locations	slint	pbf
Pack & send buffer	mpobseq_pack	pbufs
Message passing, receive buffer	mpobseq	zbufr
Unpack buffer into GOMs	mpobseq	ydgom
Unpack the GOMs for use in hop “packets”	preint, preints	ztf5, zuf5, zvf5 ….

Old GOMs → New GOMs

Description	Subroutines	Data structures
Mapping for message passing	mkglobstab	glob
Set up the GOM data structures	sugoms	
Copy variables out of model space	cobs	gfl, gmv, gmvs, psp_sb …
Put them into a buffer	cobs	pb1
Fill the “SL halo”	slcomm, slextpol	pb1
Management	obshor	
Interpolate to observation locations	slint	pbf
Pack & send buffer	mpobseq_pack	pbufs
Message passing, receive buffer	mpobseq	zbufr
Unpack buffer into GOMs	mpobseq	ydgom
Unpack the GOMs for use in hop “packets”	preint, preints	ztf5, zuf5, zvf5 ….



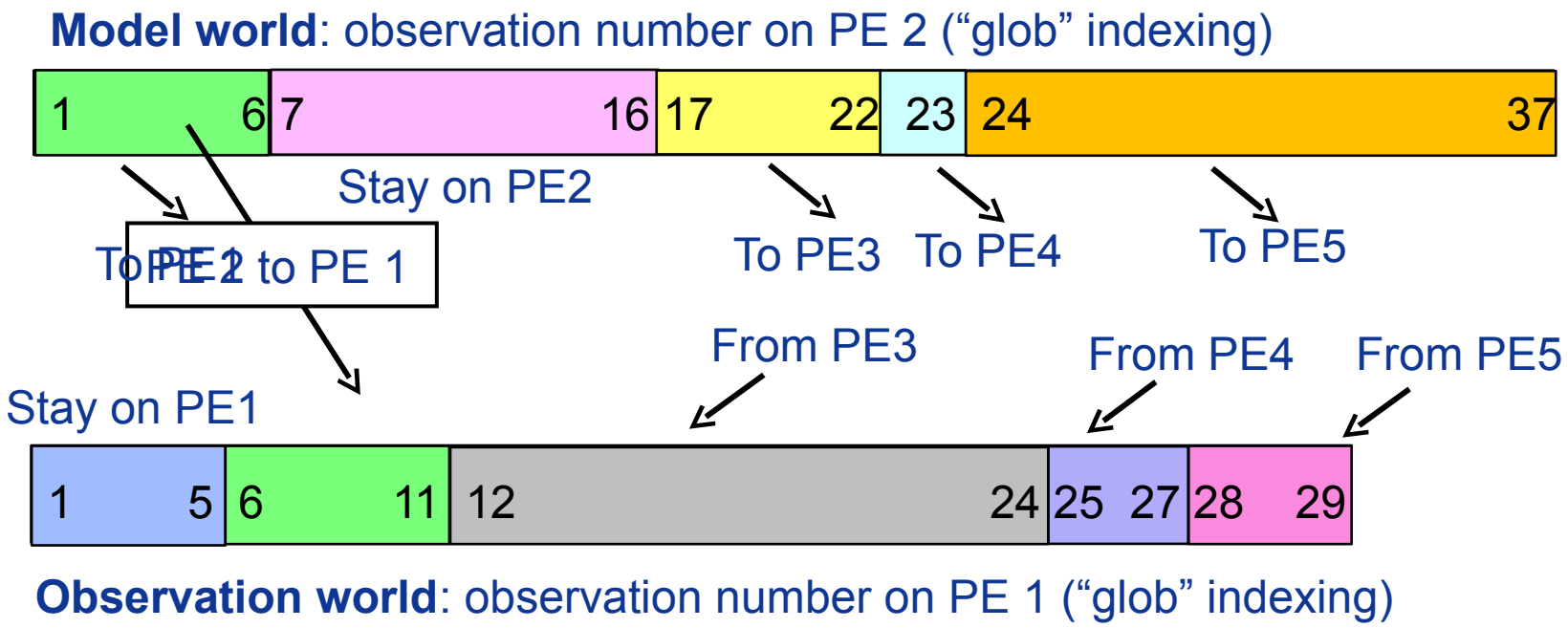
Private details:

Message passing

mpobseq.F90

creates the mapping between model PEs and observation PEs, which is stored in the “GLOB” tables

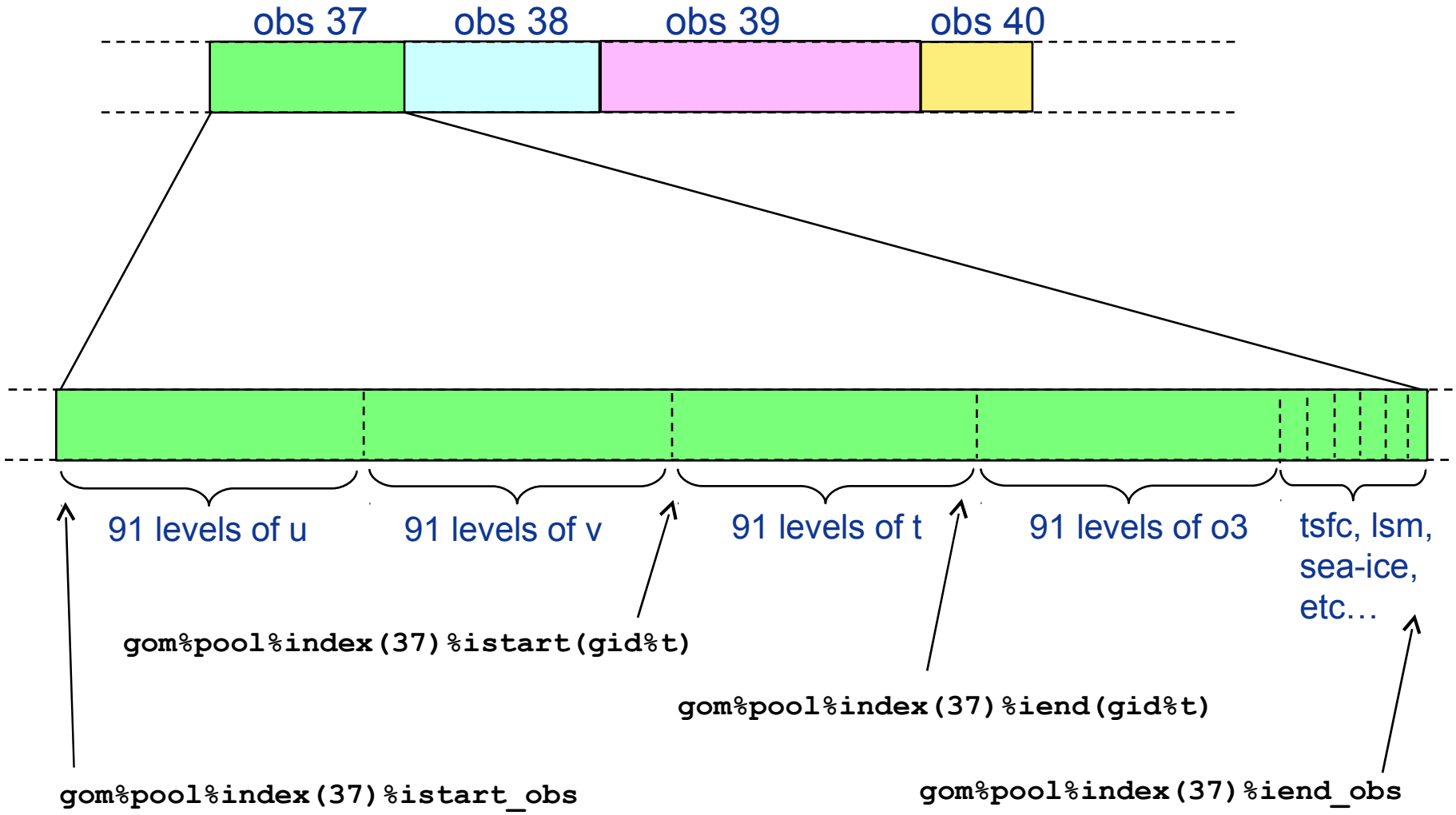
not rewritten: just a few minor tweaks to the indexing



Private details:
GOM storage areas

`gom%post%storage(:)` Before message passing

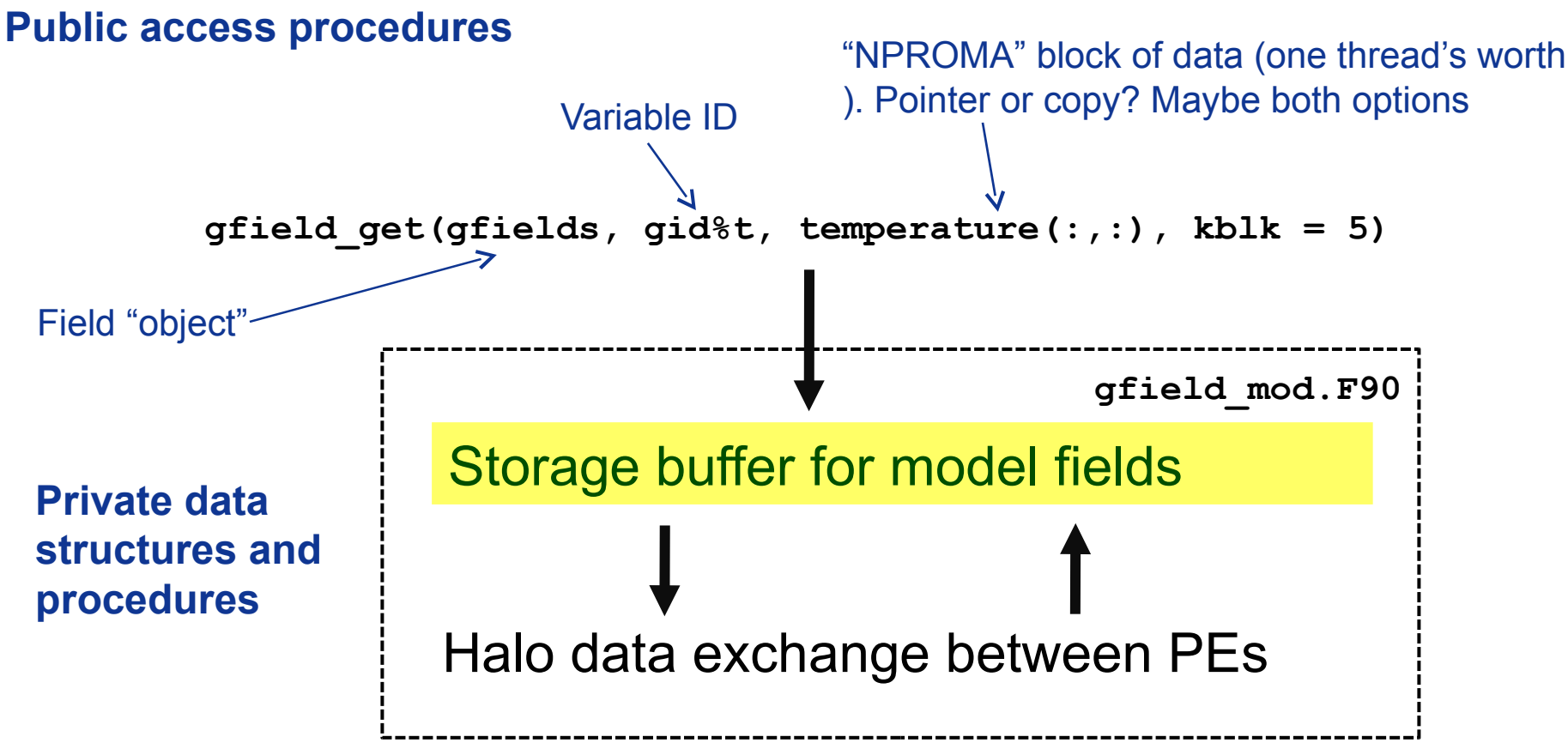
`gom%pool%storage(:)` After message passing



Some other ideas

And now for the model fields?

Aim: replace GMV, GFL, GMVS and all surface arrays



And now for the model fields?

Public access procedures

Get a profile at a single model point

```
gfield_get(gfields, gid%t, temperature(:), krof = 20)
```

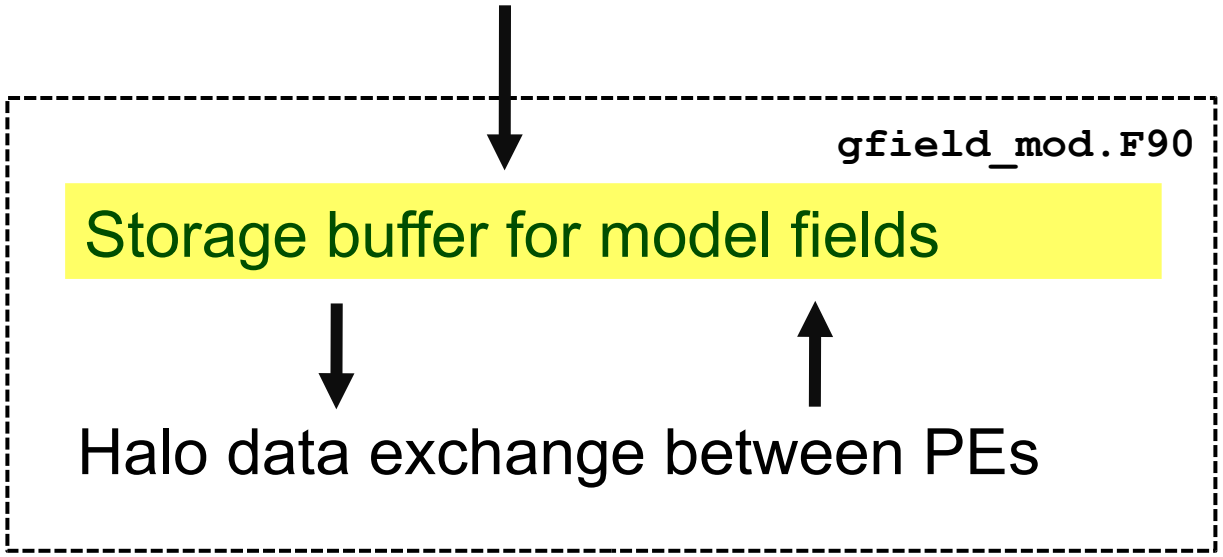
Get a thread's worth of profiles to work on – no more "kstglo" maths

```
gfield_get(gfields, gid%t, temperature(:,:), kblk = 5)
```

Get profiles including the halo from neighbouring PEs

```
gfield_get(gfields, gid%t, temperature(:,:), kblk = 5, ldhalo=.true.)
```

Private data structures and procedures



And now for the model fields?

Initialise all fields to zero:

```
gfield_set_all(gfields, 0.0)
```

Initialise just the surface fields to zero:

```
gfield_set_all(gfields, 0.0, ktype = gfield_type%surface)
```

Do the timestepping for physics fields:

```
gfield_copy(gfields_t0, gfields_t1, ktype = gfield_type%physics)
gfield_add (gfields_t1, gfields_tendency, ktype = gfield_type%physics)
```

Call callpar:

```
call callpar(gfields)
```

The observation operator problem

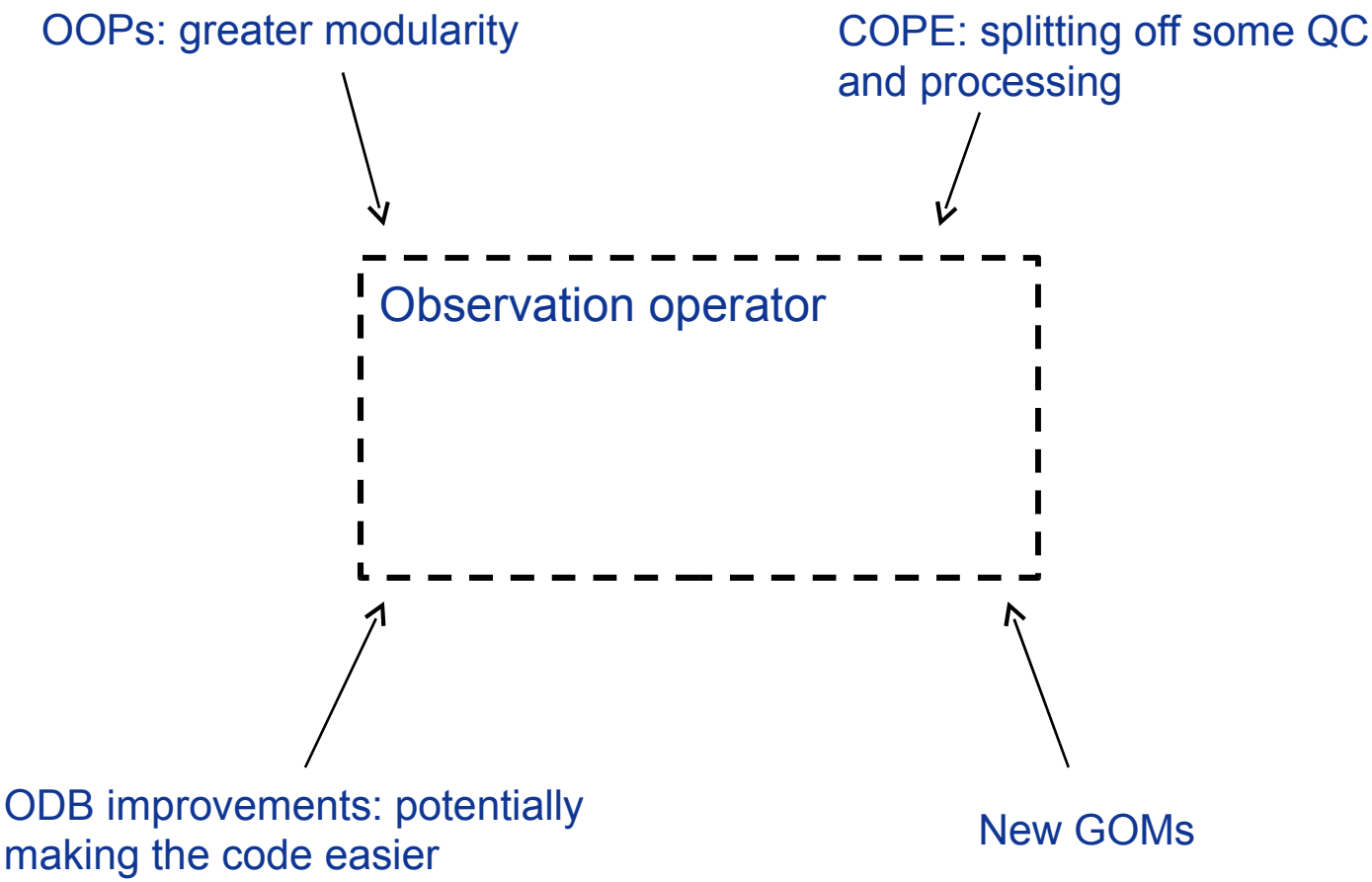
The observation code is verbose, complex and often badly written

A “copy and paste” approach is a typical way to create new code: just add a new subroutine or a new “if” block for each new observation type (× 100 places)

Vast amounts of code duplication

Area	Number of routines	Number of lines of code
op_obs	240	71340
pp_obs	112	24252
obs_preproc	210	70205

Developments affecting the observation operator



Conclusions

New GOMs

Modular, generic approach based on private data and public access functions.

Implementation details are hidden

Implementation details can change without affecting the rest of the code

Nobody should need to know about the internal details!

10,000 lines of code reduced to about 2,000

Much easier to add new variables

We can now start thinking about new science:

time averaging and accumulation

space averaging

Time to move SMOS, rain gauge and rain radar out of the model physics?

Coding in the IFS

Some problems with the IFS code:

"Copy and paste" approach, particularly prevalent in the observation world

New features require code changes in dozens of different places

Hard to understand

Hardcoding, hacks, and many other "coding horrors"

Good practice (the tip of an iceberg):

Modularity

- Public interfaces
- Private data; private internal procedures

Generic code - never duplicate!

Self-documenting code

Bad code makes it hard to maintain existing scientific features, let alone add new ones

We need better coding in the IFS

Most IFS code is written by scientists, not software engineers

How do software engineers improve their coding skills?

Short courses:

- Learn advanced language features
- Practice using them

Inspiration from well-written existing code:

- Sometimes difficult to find in the IFS!

Books:

- “Code Complete” by Steve McConnell
- don’t add any more “coding horrors”

Code reviews:

- Changes are not allowed into the codebase without review by an experienced engineer

We need to improve the existing code

Local cleaning

Sometimes useful

Sometimes very slow and does not fix the underlying problem

“Bottom up” modularisation

Example: new GOMs

Completely new code

Use an offline prototype to make development quicker

“Top down” modularisation

Required by OOPs

Future ideas

More cleaning and modularisation in the gom and interpolation: cobsall, obshor, slint, mkglobstab

Apply “new GOM” principles to the management of model fields

The observation operator:

Overlapping developments happening in OOPS, COPE, new ODB, general cleaning (e.g. new GOMs)

Do we really have a coherent plan?

Doesn’t the observation operator need a complete rewrite anyway?