# 2019-08-01: Quality Control

Yannick opened the meeting by announcing that we are about to eliminate the Fortran config_mod in OOPS. This was also mentioned last week and there is a pull request now in OOPS that is ready to merge. If you are not ready for it, then this merge could break your builds and/or tests. **So, please check your code to make sure it works with the feature/remove_config_mod branch in OOPS**.

If your code is not compatible with this branch, there are two alternative approaches to fix it:

1. replace config_mod calls such as config_get_real() with the equivalent calls from fckit configuration objects. For examples of how to do this, see The remove_config_mod pull request in OOPS.
2. If you don't have time to implement the fckit configuration calls now, you can copy the config_mod.F90 file in OOPS to your own repo and add it to your CMakeLists.txt file for compilation.

After this announcement, we moved to our topic of the day, which is QC.

Andrew Collard shared some slides that were presented by Yanqiu at EMC. A pdf version follows:

EMC JEDI QC Requirements.pdf

Please consult the slides for further details; what follows is a brief summary.

From the perspective of EMC (and others), the purpose of QC is to remove unfit observations from assimilation, to adjust observation errors, and to constrain the bias correction (see slide 1). And, the current objective of the QC implementation in JEDI is to replicate the GSI functionality. One of the most important aspects of this is the cloud implementation.

Then Yanqui showed an example of the QC GSI requirements for AMSU. This example focuses on microwave radiance as an observation class; Cory mentioned that Emily is working on something similar for IR. This serves to illustrate the scope of the issues that the QC implementation must address.

ATMS-AMSU-A Al...QC Checks.xlsx

Columns on the spreadsheet include:

- QC Check: the nature of the check begin done (e.g. checking values or bounds or surface types) . These can be different for different data types (e.g. microwave or IR channels) or conditions (land/ocean, cloud cover)
- Dependencies: Variables and other data needed to compute the check
- Result if triggered
- Whether or not this check is specific to a particular instrument or data type
- Notes on particular checks that might be included as metadata

Yanqui noted that there are different cloud detection algorithms for infrared and microwave channels and referred people to the backup slides for details.

Yannick thanked Yanqui and Andrew for the presentation and then opened the floor for questions. He commented that, in addition to the listed dependencies and metadata, it is also important to know where the data came from, for example, whether it is from an Obs file, a GeoVals object, or processed by a model such as CRTM. The former (Obs file) is the most straightforward to deal with but if the data comes after processing from an obs operator then the QC approach may be more subtle.

Anna pointed out that ufo can access metadata through the ioda ObsSpace interface. She also brought up an issue that was revisited multiple times for the remainder of the meeting, namely what to do about data products that are computed in ufo by the obs operators but that are not formal outputs of H (x). If these data products are needed by other objects/applications (such as bias correction), then should they be recomputed or should they be passed or stored somewhere? If the latter, then where should they be stored?

This relates to some work that Hailing has done for the GNSSRO operator in ufo. Steve asked if these supplementary data products could be handled as supplemental obs vectors but Yannick responded that it doesn't always fit into that description. Anna agreed that it's often not well represented as an obsvector and may require some custom list of obs data objects. Yannick said that that's essentially what Hailing did but he says we need a better long-term solution. This is essentially using ObsSpace to store global variables, which is not what it was designed for.

Yanqui then asked about a related topic, pointing out that some variables are only needed for a subset of sensors or applications (e.g. DA, QC, bias correction). Should these be defined outside of QC?

Yannick responded that all variables required for the workflow should be included in the GeoVaLs and different objects can access what they need. He also commented that the emissivity examples we've been discussing are relatively challenging - some other obs types are easier to deal with.

Chris S pointed out that some input variables/dependencies needed to compute QC at a particular location may depend on whether that location is clear or cloudy. Yanqui said this is worked into their QC procedure. Tom agreed that this can occur. Yannick mentioned that this can be handled by means of missing values or zeros in those locations where a particular variable is not needed.

Yannick then opened the floor for other questions.

Chris S asked to what extent QC can be model-dependent; for example CRTM vs RTTOV processing. Yannick and Hailing gave another example of GNSSRO and its dependence on ROPP. Yannick suggested that it could be handled by metadata specified in the yaml file but we don't yet have a good way of dealing with it. There was further discussion about tradeoffs between making related QC filters generic versus specific to particular models or processing procedures. Yannick pointed out that no changes would be needed in the yaml file - you can still specify model-dependent parameters or constraints in the yaml file but these will just be ignored if they are not relevant to the ObsSpace or Obs operator objects that are instantiated by the factories.

Yanqui then mentioned that the difference between the QC diagnostics in regions with and without clouds can be used to quantify the effect of clouds. Anna asked if this would involve two separate calls to CRTM but Yanqui said it could be done in a single call. Yannick mentioned that this gets back to the issue noted before, namely how to handle data products computed by the obs operator that are not used for DA but may be useful for QC or bias correction. Anna added that the obs operator may not always compute these secondary data products so this has to be taken into account.

Anna had another question on Yanqui's first slide. She said we already have QC filters that remove observations (first bullet) and asked if they need to be more generic. Yannick said that is easy to do if needed. They both agreed that the process of QC filtering spans a large scope - we probably can't do something that's completely generic and could handle everything - we may need a few different abstract classes that handle different QC use cases. Regarding the other bullets on slide 1, namely modifying observation errors in cases where the QC indicates additional uncertainty and constraining the bias correction, it was agreed that mechanisms to do this are not currently implemented in JEDI but need to be.

Travis then emphasized the need to differentiate between different reasons for rejecting data from QC: for further discussion see the GitHub issue on ufo. Yannick agreed we can make the flags flexible but we want to use standards that don't complicate the analysis. For example, we may wish to use two integer flags, one for indicating whether the data should be used and another for indicating the reason for rejection. We want to balance flexibility and functionality.

Yannick closed the meeting by reminding people that we are alternating weekly meeting between focused topics and general round-table discussion. So, next week will be a general round-table update. We do not yet have a topic for August 15 so suggestions are welcome. More generally:

**Please let us know what topic you would like to see addressed in one of the weekly meetings**