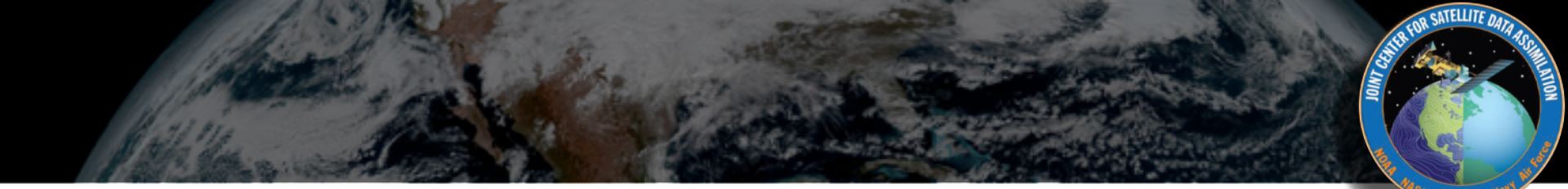

The Joint Effort for Data assimilation Integration

Observation Filters and Quality Control

Joint Center for Satellite Data Assimilation (JCSDA)

Observation Filters

Observation filters can be applied before the observation operator itself, or after if the simulated observation value is needed.

Applying filters before simulating observations can significantly reduce computational cost.

A list of filters is applied: filters can be chained.

Adding filters to the list is controlled from configuration file (json or yaml).

ObsFilter Interface

Observation filters must implement the public interface:

```
class ObsFilter {
 public:
  ObsFilter(const ioda::ObsSpace &, const eckit::Configuration &);
  ~ObsFilter();

  void priorFilter(const GeoVaLs &) const;
  void postFilter(const ioda::ObsVector &) const;
};
```

Filters have access to:

- Observations metadata (through the ObsSpace)
- Fields values at observations locations (GeoVaLs)
- Simulated observation values (`postFilter` only)

Filters often modify QC flags in the metadata (or observation error values)

Example: background check

Observation Space

User configuration

```
subroutine ufo_bgcheck_create(self, obspace, conf)
implicit none
type(ufo_bgcheck), intent(inout) :: self
type(c_ptr), value, intent(in)   :: obspace
type(c_ptr), intent(in)          :: conf

self%variable = config_get_string(conf, max_length, "variable")
self%threshold = config_get_real(conf, "threshold")
if (self%threshold<=0.0_kind_real) call abor1_ftn(”Error threshold")
self%obsdb = obspace

end subroutine ufo_bgcheck_create
```

Keep observation space information

Example: background check

```
subroutine ufo_bgcheck_post(self, hofx)
type(ufo_bgcheck), intent(in) :: self
real(c_double),  intent(in)   :: hofx(:)
! Some declarations skipped

missing = obspace_missing_value()
iobs = obsspace_get_nlocs(self%obsdb)
allocate(yobs(iobs))
allocate(yerr(iobs))
allocate(flags(iobs))
flags(:) = 0

call obsspace_get_db(self%obsdb, "ObsValue", trim(self%variable), yobs)
call obsspace_get_db(self%obsdb, "ObsError", trim(self%variable), yerr)

do jobs = 1, iobs
  if (hofx(jobs)/=missing .and. yobs(jobs)/=missing .and. yerr(jobs)/=missing) then
    if (abs(hofx(jobs)-yobs(jobs)) > yerr(jobs)*self%threshold) then
      flags(jobs) = 2
    endif
  else
    flags(jobs) = 1
  endif
Enddo

call obsspace_put_db(self%obsdb, "QC", trim(self%variable), flags)

deallocate(yobs, yerr, flags)

end subroutine ufo_bgcheck_post
```

Background check configuration

```
window_begin: '2018-04-14T21:00:00Z'
window_end: '2018-04-15T03:00:00Z'
Observations:
  ObsTypes:
  - ObsType: Radiosonde
    ObsData:
      ObsDataIn:
        obsfile: Data/sondes_obs_2018041500_m.nc4
      ObsDataOut:
        obsfile: Data/sondes_obs_2018041500_m_out.nc4
    variables:
    - air_temperature
    ObsFilters:
    - Filter: Background Check
      variable: air_temperature
      threshold: 3.0
    - Filter: Buddy Check
      radius: 1000.0
    ObsBias: {}
```

Practical (tomorrow)

There are a number of issues in the background check example

How would you improve it?