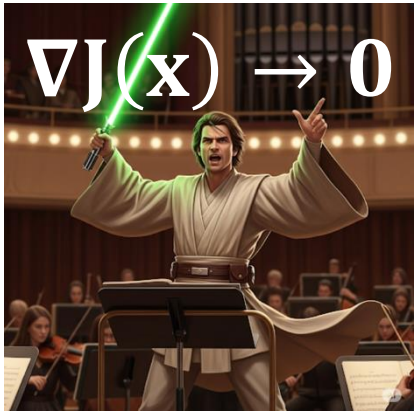
Observation bias correction and QC in JEDI-Intro to UFO

MPAS-JEDI Aug 2026
Francois Vandenberghe
for Christian Sampson
Contributors: Nate Crossette, Fabio Diniz

JEDI Components

OOPS



SABER

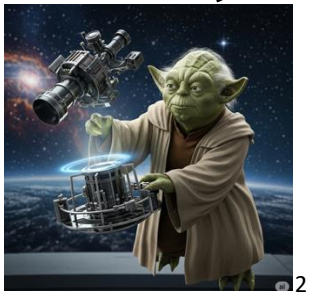
UFO, CRTM

$$J(x) = \frac{1}{2}\left(x - x^b\right)^T B^{-1}\left(x - x^b\right) + \frac{1}{2}\left(y^0 - Hx\right)^T R^{-1}\left(y^o - Hx\right)$$

VADER

(If model variables differ from analysis variables)

IODA

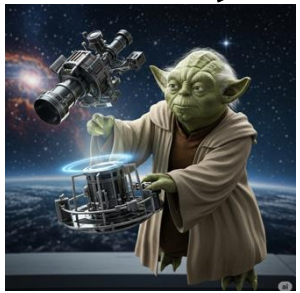


JEDI Components

UFO, CRTM

$$+\frac{1}{2}(y^0 - Hx)^T R^{-1}(y^o - Hx)$$

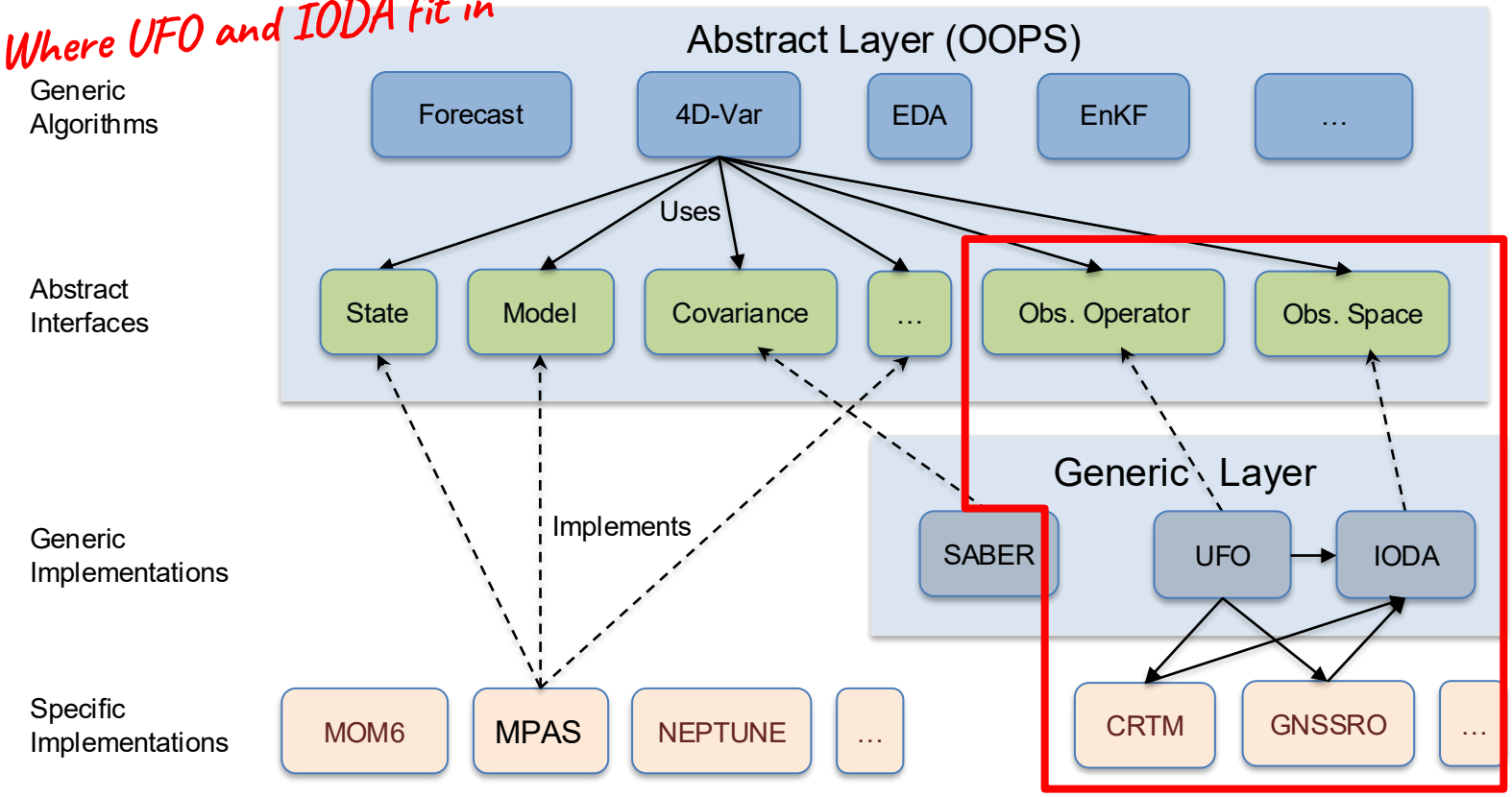
IODA



In this talk we will focus on the “Obs” part of JEDI

JEDI - Abstraction and Genericity

Where UFO and IODA fit in

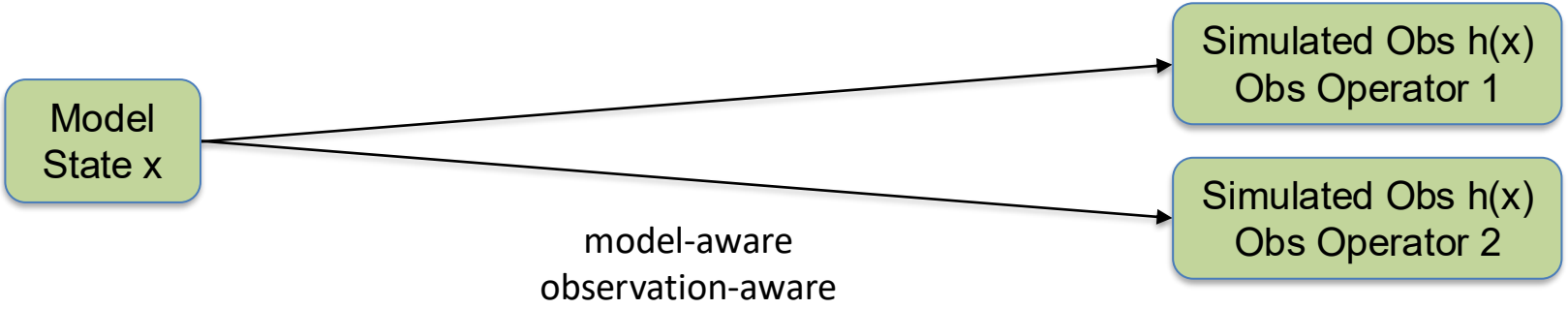


Abstract, model-agnostic (generic) DA system

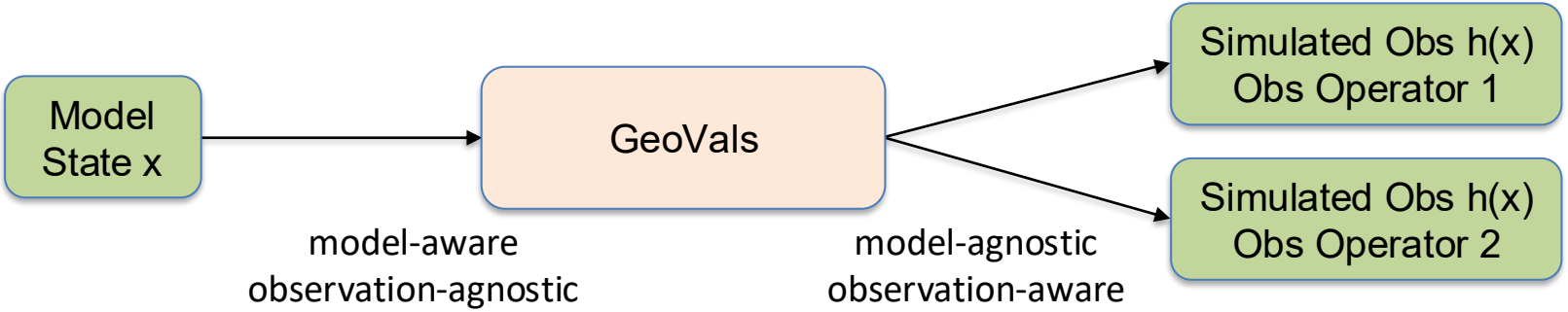
OOPS is complemented by generic (shared) components.

Some about the "U" in UFO

The "traditional" approach

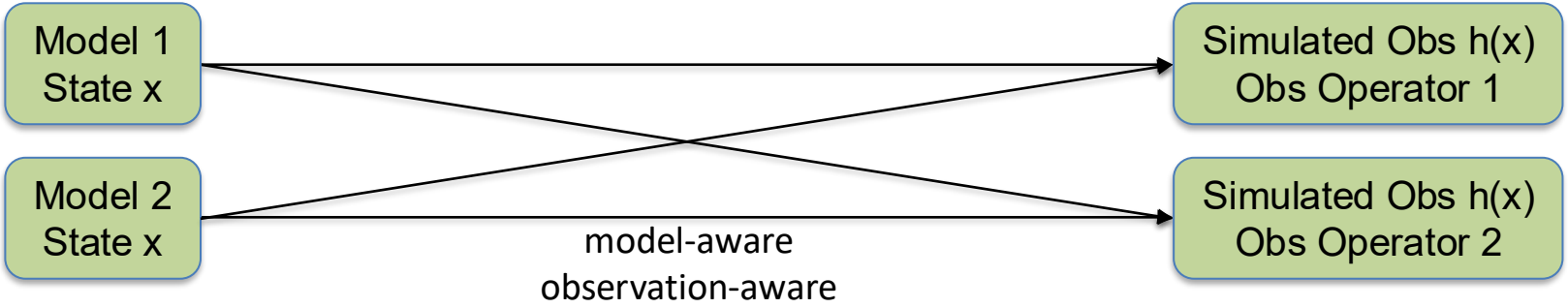


The JEDI approach

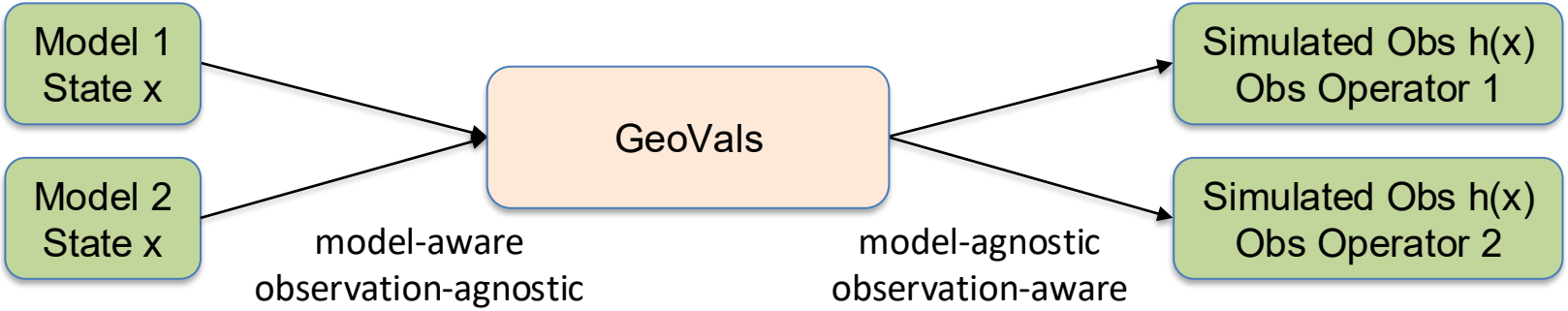


Some about the "U" in UFO

The "traditional" approach

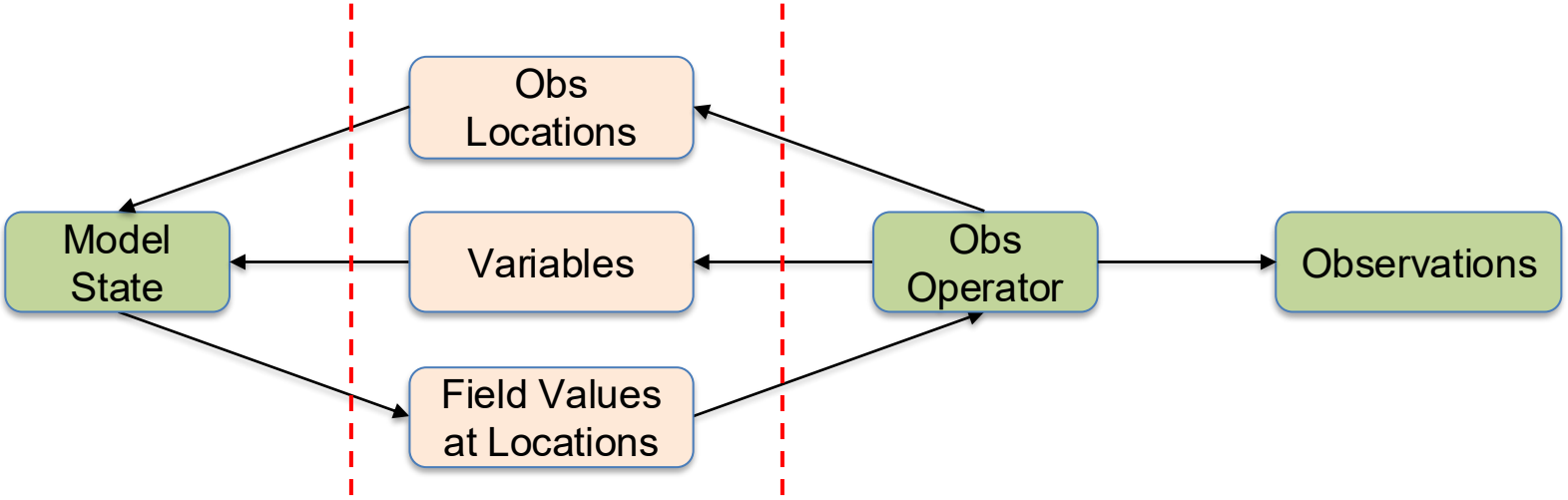


The JEDI approach



A closer look at the JEDI approach

- The forward operator defines which variables it needs from the model to simulate observations.
- The model interface then returns GeoVals, which are basically vertical profiles of requested model variables at observation locations (spatial and temporal).



For the case of radiosondes, it might request vertical profiles of the following atmospheric variables:

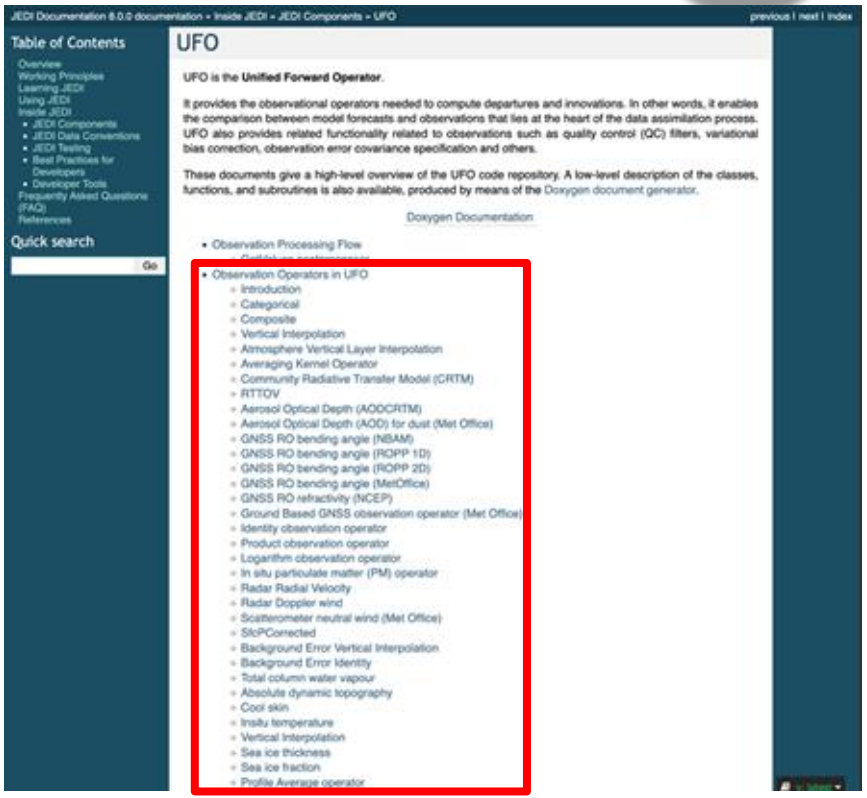
- air temperature
- humidity-related (e.g., specific humidity, relative humidity)
- wind-related (e.g., zonal & meridional wind components, wind direction & speed)
- vertical coordinate (e.g., pressure- or height-based)

The "app-store" of observation operators

A wide variety of operators have been interfaced with JEDI, a few highlights:

(and are constantly being)

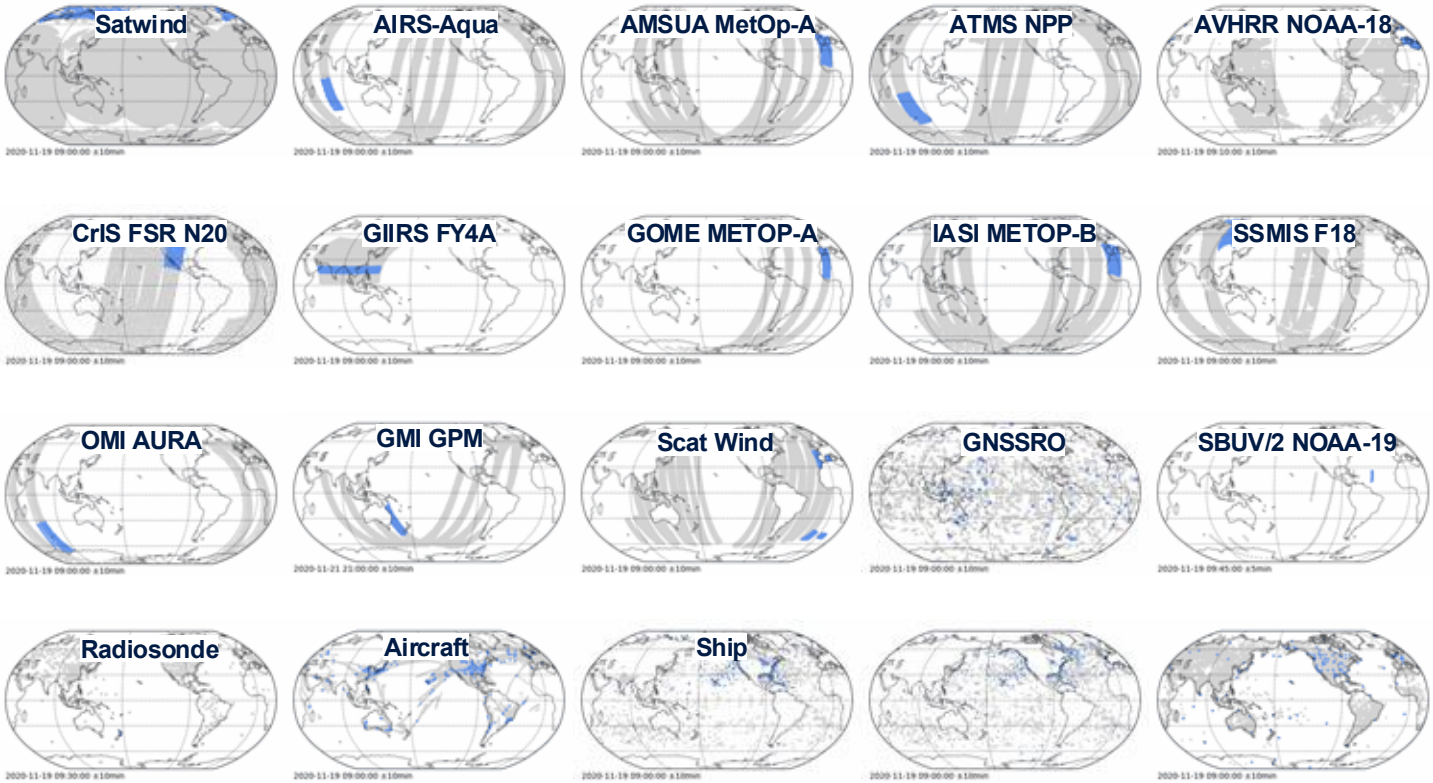
- Conventional:
  - Vertical Interpolation (VertInterp)
- Radiances:
  - Community Radiative Transfer Model (CRTM)
  - Radiative Transfer for TOVS (RTTOV)*
- Radio occultation (RO):
  - NCEP's Bending Angle Method (NBAM)
  - MetOffice RO one-dimensional
  - Radio Occultation Processing Package 1D (ROPP 1D)*
  - Radio Occultation Processing Package 2D (ROPP 2D)*
- Ground-based GNSS:
  - MetOffice GNSS
- Aerosol Optical Depth (AOD):
  - CRTM AOD
  - MetOffice AOD for dust
- Ocean surface winds:
  - MetOffice scatterometer neutral wind
  - Wind speed operator
- Reflectances:
  - CRTM
- Radar:
  - Reflectivity (DirectZDA)
  - Doppler wind (RadarDopplerWind)
  - Radial Velocity (RadarRadialVelocity)



See detailed information at Observation Operators in UFO.

* requires registration by their providers

A few examples of the available operators

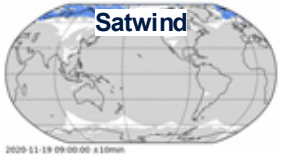


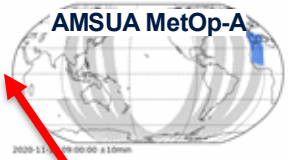
A few examples of the available operators

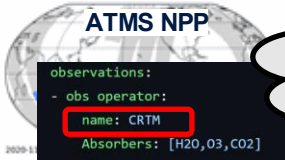
Excerpt from satwind.yaml

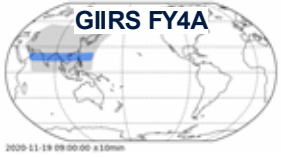
```
observations:
- obs operator:
    name: VertInterp
    observation alias file: ../resources/namemap/test_name_map.yaml
  linear obs operator:
    name: VertInterp
    observation alias file: ../resources/namemap/test_name_map.yaml
    vertical coordinate: air_pressure
    observation vertical coordinate: pressure
    interpolation method: log-linear
  obs space:
    name: Satwind
    obsdatain:
      engine:
        type: H5File
        obsfile: Data/ufo/testinput_tier_1/satwind_obs_2018041500_m.nc4
#   obsdataout:
#     engine:
#       type: H5File
#       obsfile: Data/satwind_obs_2018041500_m_opr_out.nc4
    simulated variables: [windEastward, windNorthward]
  geovals:
    filename: Data/ufo/testinput_tier_1/satwind_geoval_2018041500_m.nc4
  vector ref: GsiHofX
  tolerance: 1.0e-04
```

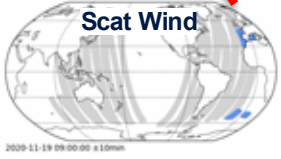
A few examples of the available operators

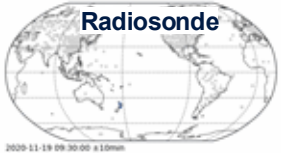


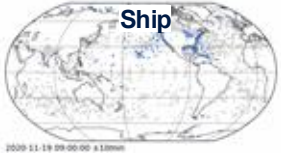








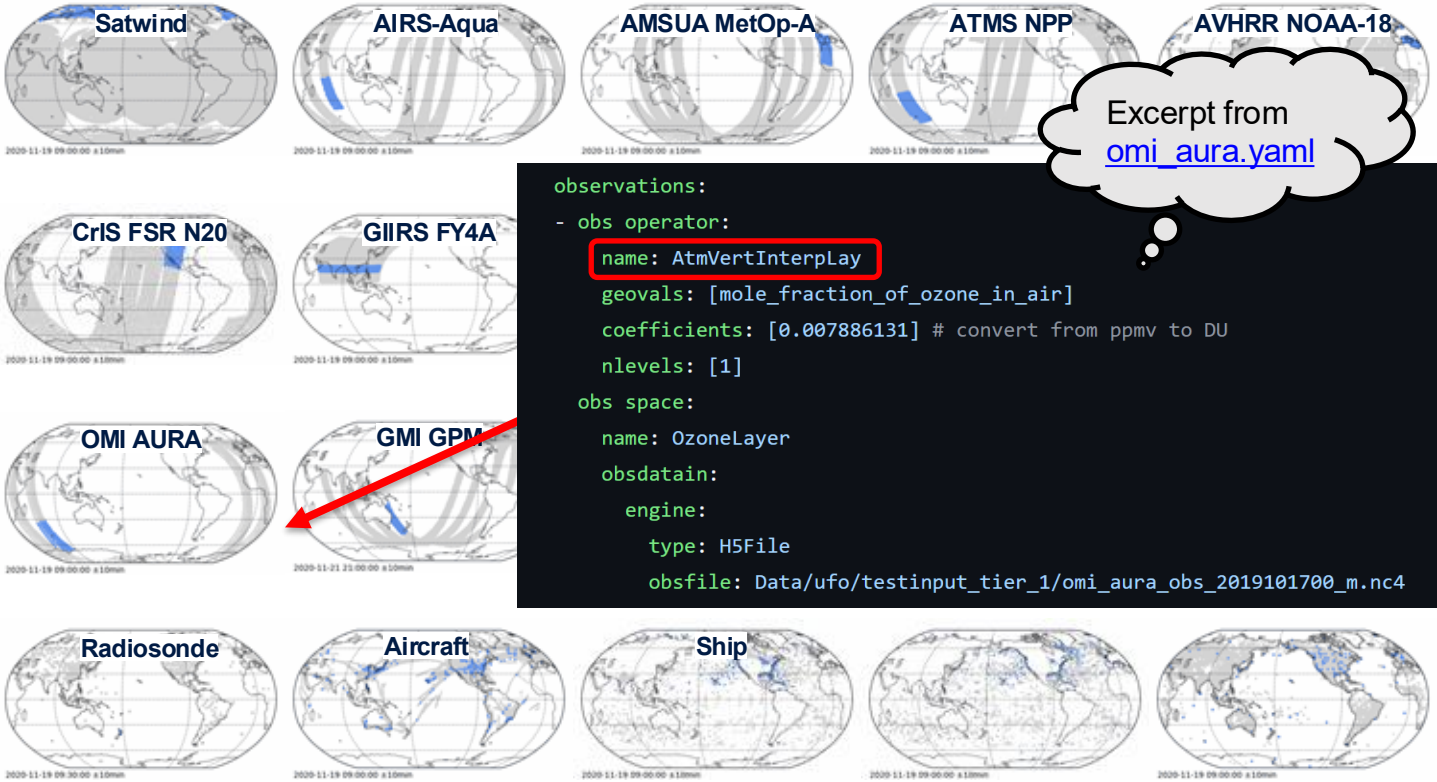




Excerpt from
[airs_aqua_gfs_HofX.yaml]

```
observations:
- obs operator:
    name: CRTM
    Absorbers: [H2O,O3,CO2]
    obs options:
      Sensor_ID: airs_aqua
      EndianType: little_endian
      CoefficientPath: Data/
  obs space:
    name: airs_aqua
    obsdatain:
      engine:
        type: H5File
        obsfile: Data/ufo/testinput_tier_1/instruments/radiance/airs_aqua_obs_2020110112_m.nc4
    simulated variables: [brightnessTemperature]
    channels: 1, 6, 7, 10, 11, 15, 16, 17, 20, 21, 22, 24,
              27, 28, 30, 36, 39, 40, 42, 51, 52, 54, 55, 56, 59, 62, 63, 68, 69, 71,
              72, 73, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 86, 92, 93, 98, 99, 101,
              104, 105, 108, 110, 111, 113, 116, 117, 123, 124, 128, 129, 138, 139,
              144, 145, 150, 151, 156, 157, 159, 162, 165, 168, 169, 170, 172, 173,
              174, 175, 177, 179, 180, 182, 185, 186, 190, 192, 198, 201, 204, 207,
              210, 215, 216, 221, 226, 227, 232, 252, 253, 256, 257, 261, 262, 267,
              272, 295, 299, 300, 305, 310, 321, 325, 333, 338, 355, 362, 375, 453,
              475, 484, 497, 528, 587, 672, 787, 791, 843, 870, 914, 950, 1003, 1012,
              1019, 1024, 1030, 1038, 1048, 1069, 1079, 1082, 1083, 1088, 1090, 1092,
              1095, 1104, 1111, 1115, 1116, 1119, 1120, 1123, 1130, 1138, 1142, 1178,
              1199, 1206, 1221, 1237, 1252, 1260, 1263, 1266, 1285, 1301, 1304, 1329,
              1371, 1382, 1415, 1424, 1449, 1455, 1466, 1477, 1500, 1519, 1538, 1545,
              1565, 1574, 1583, 1593, 1614, 1627, 1636, 1644, 1652, 1669, 1674, 1681,
              1694, 1708, 1717, 1723, 1740, 1748, 1751, 1756, 1763, 1766, 1771, 1777,
              1780, 1783, 1794, 1800, 1803, 1806, 1812, 1826, 1843, 1852, 1865, 1866,
```

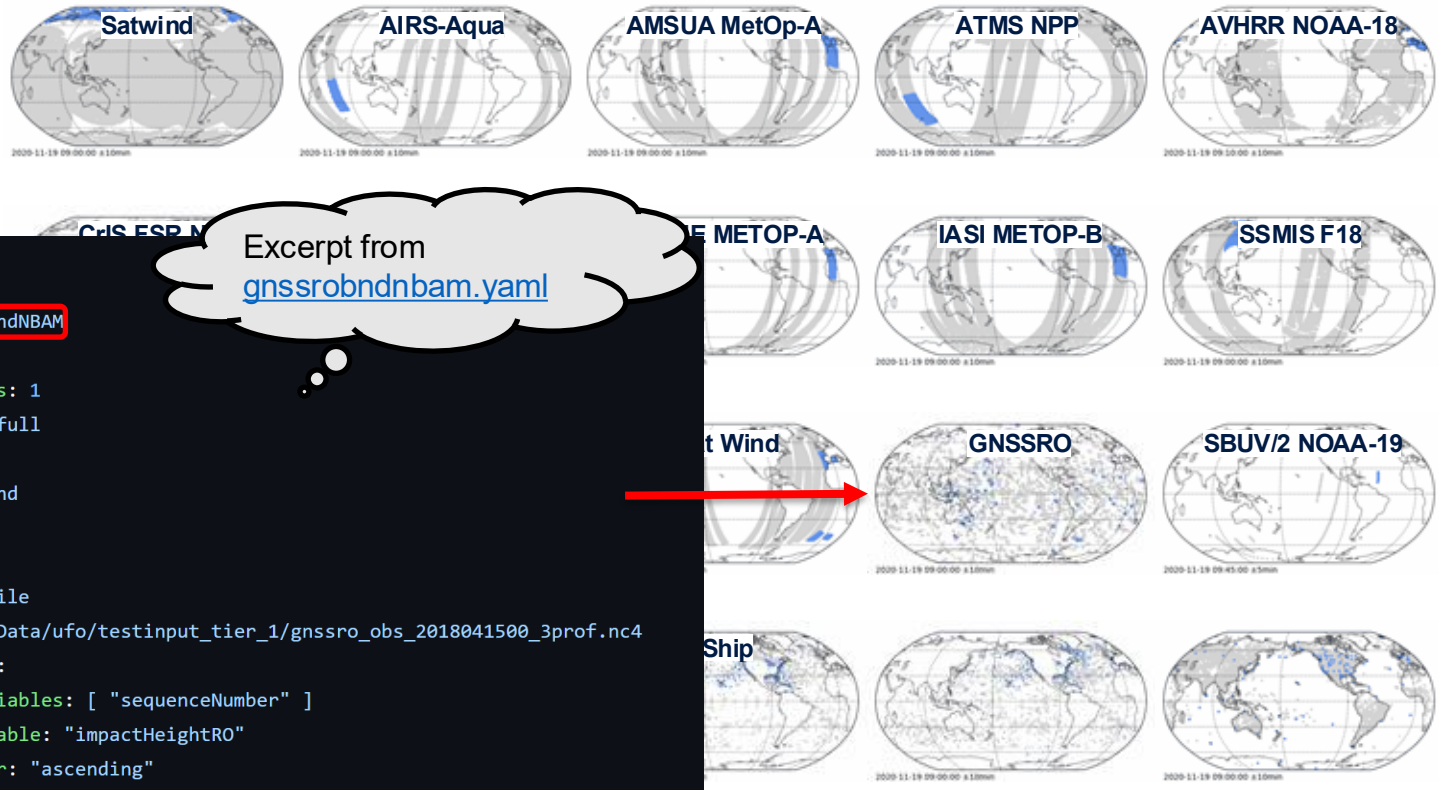
A few examples of the available operators



Excerpt from [omi_aura.yaml]

```yaml
observations:
- obs operator:
    name: AtmVertInterpLay
    geovals: [mole_fraction_of_ozone_in_air]
    coefficients: [0.007886131] # convert from ppmv to DU
    nlevels: [1]
  obs space:
    name: OzoneLayer
    obsdatain:
      engine:
        type: H5File
        obsfile: Data/ufo/testinput_tier_1/omi_aura_obs_2019101700_m.nc4
```

A few examples of the available operators



Excerpt from [gnssrobndnbam.yaml]

```
observations:
- obs operator:
    name: GnssroBndNBAM
    obs options:
      use_compress: 1
      vertlayer: full
  obs space:
    name: GnssroBnd
    obsdatain:
      engine:
        type: H5File
        obsfile: Data/ufo/testinput_tier_1/gnssro_obs_2018041500_3prof.nc4
      obsgrouping:
        group variables: [ "sequenceNumber" ]
        sort variable: "impactHeightRO"
        sort order: "ascending"
```

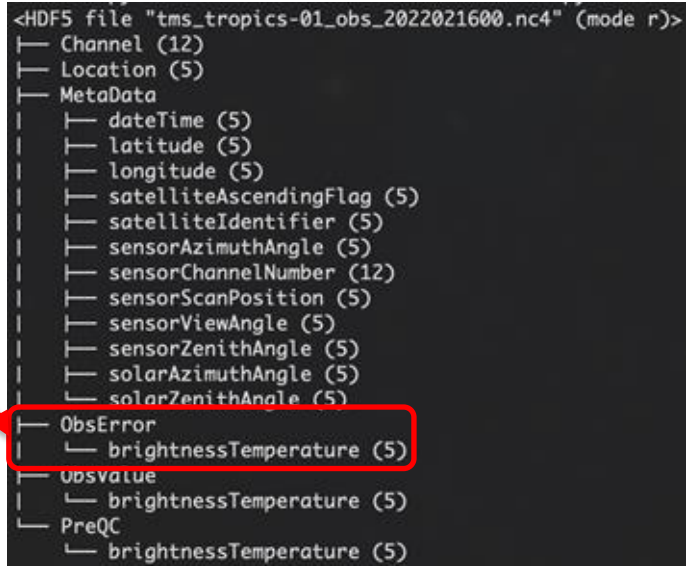
Observation Uncertainties in UFO

In JEDI, observation errors can be prescribed:

- when encoding the IODA file:
  - by assigning standard deviation values for each observation in the file, via the ObsError group.
- at runtime of JEDI via YAML:
  - by assigning standard deviation values through an observation filter
  - with the option to be combined with observation functions.

Regardless of the option chosen, they can be inflated/deflated based on situation-dependent cases.

Important to mention that the observation error values obtained after running all the filters is the one passed for the DA solver.



Observation Uncertainties in UFO

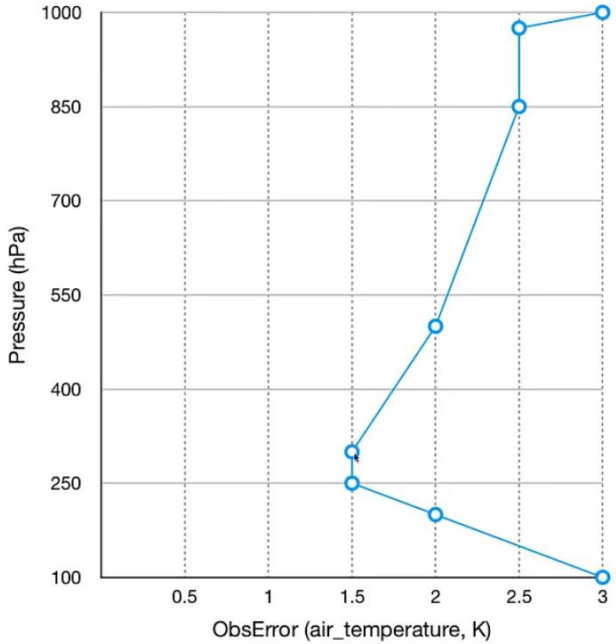
In JEDI, observation errors can be prescribed:

- when encoding the IODA file:
  - by assigning standard deviation values for each observation in the file, via the ObsError group

```
- filter: Perform Action
  filter variables:
  - name: airTemperature
  - name: virtualTemperature
  action:
    name: assign error
```

Excerpts from

```
# The next series of filters assigns obsError as a linear interpolation from
# in another variable, such as pressure/altitude.
  - filter: Perform Action
    filter variables:
    - name: air_temperature
    action:
      name: assign error
      error function:
        name: ObsErrorModelStepwiseLinear@ObsFunction
        options:
          xvar:
            name: air_pressure@ObsValue
          xvals: [10000, 20000, 25000, 30000, 50000, 85000, 97500, 100000]
          errors: [3.0, 2.0, 1.5, 1.5, 2.0, 2.5, 2.5, 3.0]
```



Greg Thompson

Observation Uncertainties in UFO

In JEDI, observation errors can be prescribed:

- when encoding the IODA file:
  - by assigning standard deviation values for each observation in the file, via the ObsError group.
- at runtime of JEDI via YAML:
  - by assigning standard deviation values through an observation filter
  - with the option to be combined with observation functions.

Regardless of the option chosen, they can be inflated/deflated based on situation-dependent cases.

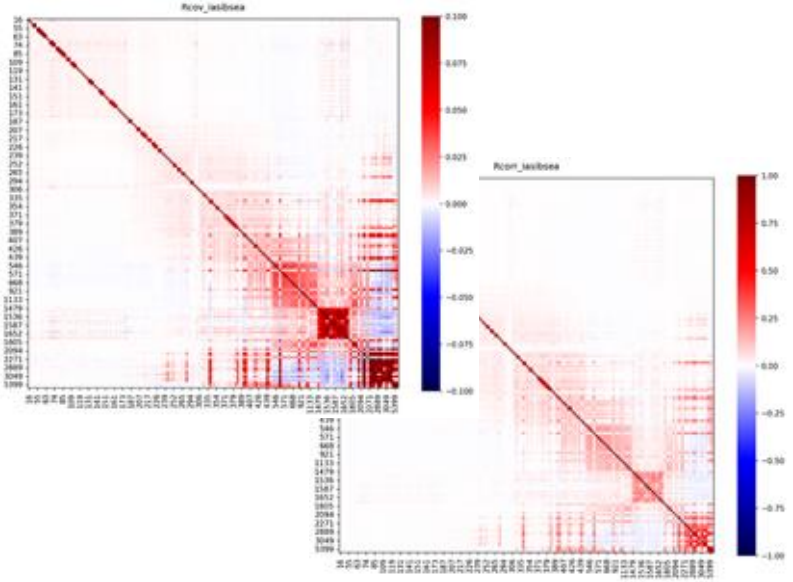
Important to mention that the observation error values obtained after running all the filters is the one passed for the DA solver.

Additionally, UFO is capable to handle the following covariance models:

- diagonal
- cross variable covariances
- within group covariances

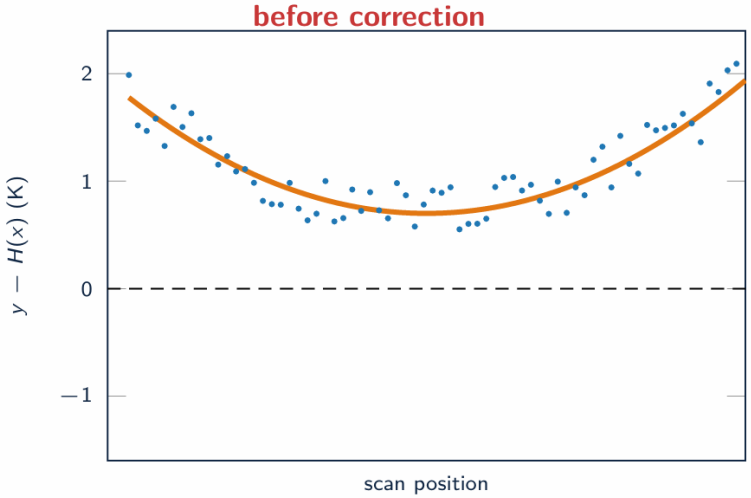
Excerpt from
obserrorcrossvarcov.yaml

```
obs error:
  covariance model: cross variable covariances
  input file: Data/ufo/testinput_tier_1/Rcov_iasi_metop-b_sea.nc4
```

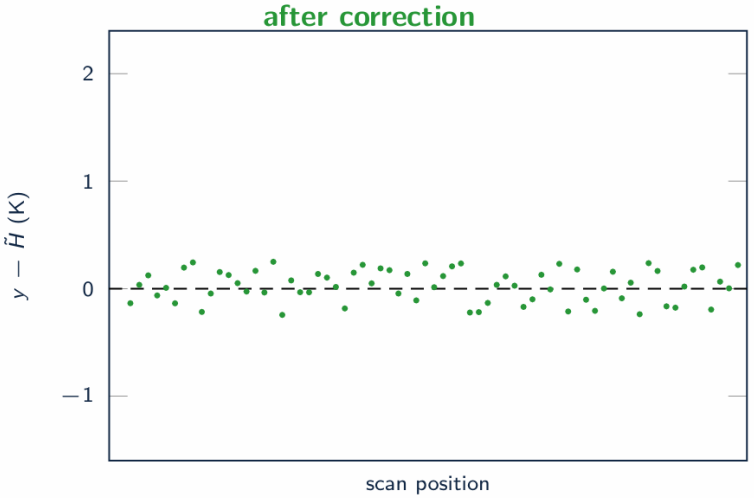


The Bias Problem

Satellite and some insitu obs carry a systematic bias (calibration drift, radiative-transfer error, scan position and air-mass effects) The issue: DA assumes $E[y - H(x)] = 0$



mean $O - B \neq 0$; structured shape



mean $\approx$ 0; only real signal remains

VarBC learns the structured part of the departure and removes it.

Modeling the Bias

$$\underbrace{\widetilde{H}(x)}_{\text{corrected}} = \underbrace{H(x)}_{\text{physics}} + \underbrace{\sum_{\{i=1\}}^{n} \beta_i p_i\left(x^b\right)}_{\text{learned correction}}$$

- $p_i\left(x^b\right)$: a predictor → a known function of state/metadata (a fixed “shape”)
- β_i: a scalar coefficient → how much of that shape to add to the correction.
- The coefficients are learned during the variational assimilation simultaneously with the state update!

UFO has Variational Bias Correction

Variational bias correction is an adaptive bias correction technique. In a DA system with VarBC, observation operators and control variables are extended to include bias correction procedures. The bias correction coefficients (β_i) are optimized as control variables in each analysis.

This is accomplished with an extended cost function:

$$J(x^T, \beta^T) = \frac{1}{2}\left(x - x^b\right)^T B^{-1}\left(x - x^b\right) + \frac{1}{2}\left(\tilde{H}(x) - y^o\right)^T R^{-1}\left(\tilde{H}(x) - y^o\right) + \frac{1}{2}\left(\beta - \beta^b\right)^T B_\beta^{-1}\left(\beta - \beta^b\right)$$

As well as an extended observation operator,

$$\tilde{H}(x) = H(x) + \sum_{\{i=1\}}^{n} \beta_i p_i\left(x^b\right),$$

utilizing predictors p_i of which many are available in UFO.

Available Predictors

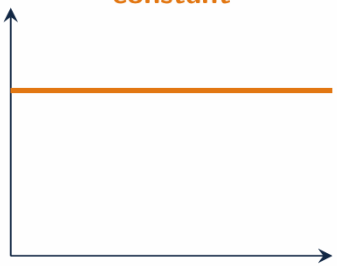
Predictor	Description
constant	A flat offset bias — value of 1.0 at every location (optionally restricted to land or sea). Captures a uniform bias common to all observations of a channel.
sensorScanAngle	The nth power of the sensor scan/view angle in degrees. Used (often with orders 1–4 together) to correct scan-position-dependent biases in cross-track scanning instruments.
legendre	Legendre polynomial of order n evaluated on the normalized scan position. Orthogonal basis for scan-dependent bias — better conditioned than raw scan-angle powers.
lapseRate	Weighted sum of layer transmittances times temperature differences. Captures air-mass-dependent biases tied to the vertical temperature structure.
thickness	Atmospheric layer thickness (km) between two pressure levels, normalized to zero mean and unit variance. Captures biases tied to mean layer temperature.
cloudWaterContent	Cloud liquid water retrieved from specific microwave channel brightness temperatures. Currently supports SSMIS. Corrects cloud-affected biases.
emissivityJacobian	Surface emissivity term from observation diagnostics. Important for microwave radiances where surface emissivity strongly affects brightness temperatures.
sineOfLatitude	sin(latitude) — captures latitudinally-dependent biases (e.g., RT model errors varying with solar zenith angle).
cosineOfLatitudeTimesOrbitNode	cos(latitude) signed by ascending/descending orbit node. Captures biases tied to orbital geometry and the diurnal cycle.
satelliteOrbitalAngle	Fourier term of the orbital angle: sin(n·θ) or cos(n·θ). Captures cyclical biases as the satellite traverses its orbit.
interpolateDataFromFile	Reads bias values from an external CSV/NetCDF file and interpolates them by metadata keys (e.g., station ID, pressure). Typically used for static BC.
satelliteSelector	Returns 1.0 for a selected satellite ID, 0.0 otherwise. Enables per-satellite coefficients in multi-satellite datasets.
obsMetadataPredictor	Generic predictor — raises any ObsSpace metadata variable to a specified power.
readBias	Reads previously estimated bias coefficients from ObsSpace as a predictor input. Used for cycling bias corrections across DA cycles.

Available Predictors (the error shapes)

Predictor	Description
constant	A flat offset bias — value of 1.0 at every location (optionally restricted to land or

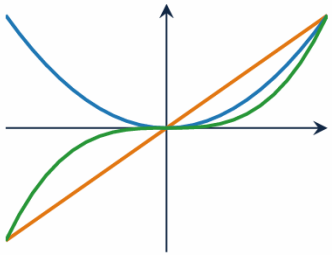
A predictor is evaluated at every obs location (fills an `ObsVector`); all derive from `PredictorBase`.

constant



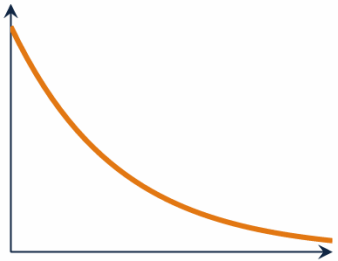
flat offset

sensorScanAngle



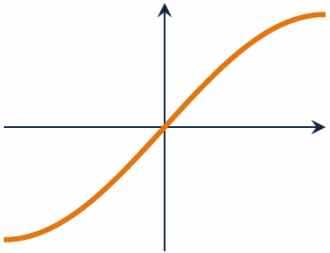
orders 1–4

lapseRate



air-mass / profile

sineOfLatitude



latitudinal

VarBC just decides *how much* of each shape to add: $\sum_i \beta_i \, p_i$.

satelliteOrbitalAngle	Fourier term of the orbital angle: sin(n·θ) or cos(n·θ). Captures cyclical biases as the satellite traverses its orbit.
interpolate_data_from_file	Reads bias values from an external CSV/NetCDF file and interpolates them by metadata keys (e.g., station ID, pressure). Typically used for static BC.
satelliteSelector	Returns 1.0 for a selected satellite ID, 0.0 otherwise. Enables per-satellite coefficients in multi-satellite datasets.
obsMetadataPredictor	Generic predictor — raises any ObsSpace metadata variable to a specified power.
read_bias	Reads previously estimated bias coefficients from ObsSpace as a predictor input. Used for cycling bias corrections across DA cycles.

Observation Quality Control in UFO

All the observation quality control procedures in UFO are performed through the so-called "observation filters".

Filters are generic, entirely configured through YAML files, and already exist for a variety of applications:

- ✓ gross error checks;
- ✓ background checks;
- ✓ list of stations to be accepted/rejected;
- ✓ thinning;
- ✓ superobbing;
- ✓ derive new variables;
- ✓ inflate/deflate observation errors;
- ✓ thigh error bounds;
- ✓ and many more.

By construction, they can be used by many observation types once they are written.

Although the generic filters cover most needs, some additional specialized information is still needed. The so-called "observation functions" become handy in those cases.

```
group: EffectiveQC {
  variables:
    int airTemperature(Location) ;
      airTemperature:_FillValue = -2147483643 ;
  data:

   airTemperature = 0, 0, 12, 12, 12, 0, 0, 0, 0, 0, 0,
```

```
namespace QCflags {
  constexpr int pass            = 0;  // we like that one!
  constexpr int passive         = 1;  // H(x) is computed (for monitoring, BC...) but obs not assimilated
// Single digit values reserved for DA use.
// For now only 0, 1 and >1 are used but keeping space for other potential use cases.

// Actual rejection flags
  constexpr int missing         = 10;  // missing values prevent use of observation
  constexpr int preQC           = 11;  // observation rejected by pre-processing
  constexpr int bounds          = 12;  // observation value out of bounds
  constexpr int domain          = 13;  // observation not within domain of use
  constexpr int black           = 14;  // observation black listed
  constexpr int Hfailed         = 15;  // H(x) computation failed
  constexpr int thinned         = 16;  // observation removed due to thinning
  constexpr int diffref         = 17;  // metadata too far from reference
  constexpr int clw             = 18;  // observation removed due to cloud field
  constexpr int fguess          = 19;  // observation too far from guess
  constexpr int seaice          = 20;  // observation based sea ice detection, also flags land points
  constexpr int track           = 21;  // observation removed as inconsistent with the rest of track
  constexpr int buddy           = 22;  // observation rejected by the buddy check
  constexpr int derivative      = 23;  // observation removed due to metadata derivative value
  constexpr int profile         = 24;  // observation rejected by at least one profile QC check
  constexpr int onedvar         = 25;  // observation failed to converge in 1dvar check
  constexpr int bayesianQC      = 26;  // observation failed due to Bayesian background check
  constexpr int modelobthresh   = 27;  // observation failed modelob threshold check
  constexpr int history         = 28;  // observation failed when compared with historical data
  constexpr int processed       = 29;  // observation processed but deliberately H(x) not calculated
  constexpr int superrefraction = 30;  // observation rejected by GNSSRO super refraction QC
  constexpr int superob         = 31;  // superob value not set at this location
```

Excerpt from UFO (src/ufo/filters/QCflags.h)

Observation Functions

Here's an example that assigns a constant observation error of 120 Pa to surface pressure observations, but only in the tropics:

```yaml
- filter: Perform Action
  filter variables:
  - name: stationPressure
  where:
  - variable:
      name: MetaData/latitude
    minvalue: -30
    maxvalue: 30
  action:
    name: assign error
    error parameter: 120
```

A more involved example: Assign errors interpolated from a file, but only where the observation has valid pressure metadata:

```yaml
- filter: Perform Action
  filter variables:
  - name: airTemperature
  where:
  - variable:
      name: MetaData/pressure
    value: is_valid
  action:
    name: assign error
    error function:
      name: ObsFunction/DrawValueFromFile
      options:
        file: Data/obs_error_table.nc4
        interpolation:
        - name: MetaData/pressure
          method: linear
```

Setting up an ObsSpace

```yaml
observations:
  observers:
  - obs space:
      name: AMSUA-NOAA19                              # Descriptive name (used in logs)
      obsdatain:
        engine:
          type: H5File                                # Backend type for reading obs
          obsfile: Data/obs/amsua_n19_obs.nc4         # Path to input observation file
      obsdataout:                                     # Optional: write processed obs
        engine:
          type: H5File
          obsfile: Data/obs/amsua_n19_out.nc4
      simulated variables: [brightnessTemperature]  # Variables that H(x) will produce
      channels: 1-15                                  # For multi-channel instruments

    obs operator:
      name: CRTM                                      # Which forward operator to use
      Absorbers: [H2O, O3]
      obs options:
        Sensor_ID: amsua_n19
        CoefficientPath: Data/crtm/

    obs error:
      covariance model: diagonal

    obs bias:                                         # Optional: bias correction
      input file: Data/satbias_amsua_n19.nc4          # Prior coefficients from previous cycle
      output file: Data/satbias_amsua_n19_out.nc4 # Updated coefficients for next cycle
      variational bc:
        predictors:                                   # Predictors estimated during minimization
        - name: constant
        - name: emissivityJacobian
        - name: sensorScanAngle
          order: 4
        - name: sensorScanAngle
          order: 3
```

Setting u

```yaml
obs bias:                                    # Optional: bias correction
  input file: Data/satbias_amsua_n19.nc4     # Prior coefficients from previous cycle
  output file: Data/satbias_amsua_n19_out.nc4 # Updated coefficients for next cycle
  variational bc:
    predictors:                              # Predictors estimated during minimization
    - name: constant
    - name: emissivityJacobian
    - name: sensorScanAngle
      order: 4
    - name: sensorScanAngle
      order: 3
    - name: sensorScanAngle
      order: 2
    - name: sensorScanAngle
      order: 1
    - name: lapseRate
      order: 2
  covariance:
    minimal required obs number: 20          # Min obs to estimate coefficients
    prior:
      input file: Data/satbias_cov_amsua_n19.nc4
      inflation:
        ratio: 1.1

obs filters:                                 # QC pipeline, applied in order listed
# Pre-filters: run before GeoVaLs, access obs + metadata only
- filter: PreQC
  maxvalue: 3

- filter: Bounds Check
  filter variables:
  - name: brightnessTemperature
    channels: 1-15
  minvalue: 100.0
  maxvalue: 500.0

# Post-filter: runs after H(x), access everything
- filter: Background Check
  filter variables:
  - name: brightnessTemperature
    channels: 1-15
  threshold: 3.0
```

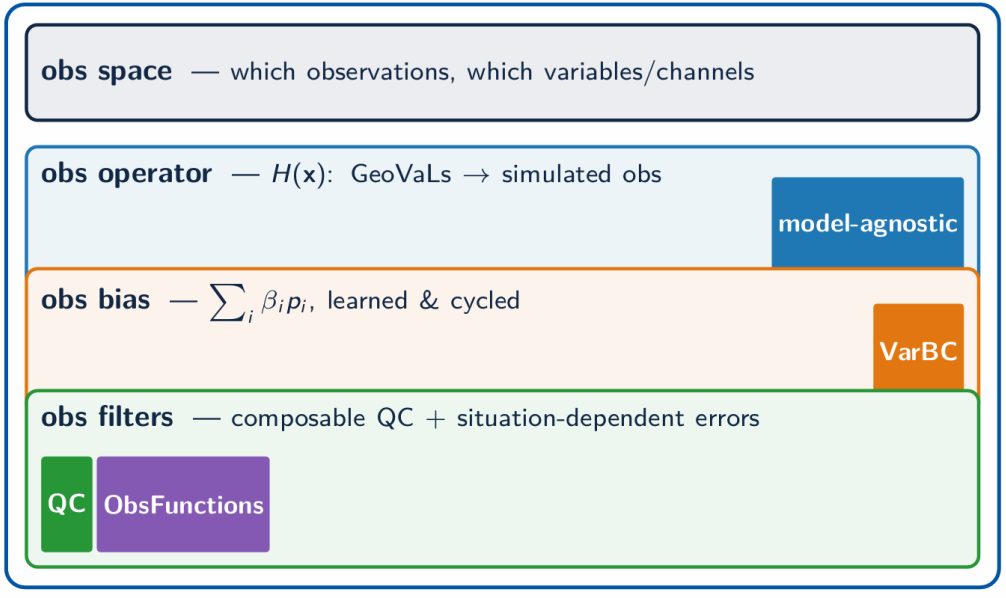
Setting u

```
      order: 1
    - name: lapseRate
      order: 2
  covariance:
    minimal required obs number: 20          # Min obs to estimate coefficients
    prior:
      input file: Data/satbias_cov_amsua_n19.nc4
      inflation:
        ratio: 1.1

obs filters:                                 # QC pipeline, applied in order listed
# Pre-filters: run before GeoVaLs, access obs + metadata only
- filter: PreQC
  maxvalue: 3

- filter: Bounds Check
  filter variables:
  - name: brightnessTemperature
    channels: 1-15
  minvalue: 100.0
  maxvalue: 500.0

# Post-filter: runs after H(x), access everything
- filter: Background Check
  filter variables:
  - name: brightnessTemperature
    channels: 1-15
  threshold: 3.0
  defer to post: true                        # Force to post stage if using auto ordering

# Conditional action with where clause
- filter: Perform Action
  filter variables:
  - name: brightnessTemperature
  where:
  - variable:
      name: MetaData/latitude
    minvalue: -30
    maxvalue: 30
  action:
    name: assign error
    error parameter: 2.0
```

Setting u

```
      order: 1
    - name: lapseRate
      order: 2
  covariance:
    minimal required obs number: 20          # Min obs to estimate coefficients
    prior:
      input file: Data/satbias_cov_amsua_n19.nc4
      inflation:
        ratio: 1.1
```

one `observers:` **entry**




```
      name: MetaData/latitude
    minvalue: -30
    maxvalue: 30
  action:
    name: assign error
    error parameter: 2.0
```

Conclusions

- UFO is one model-agnostic observation-space layer; GEOVals is the bridge that keeps the operator model-blind

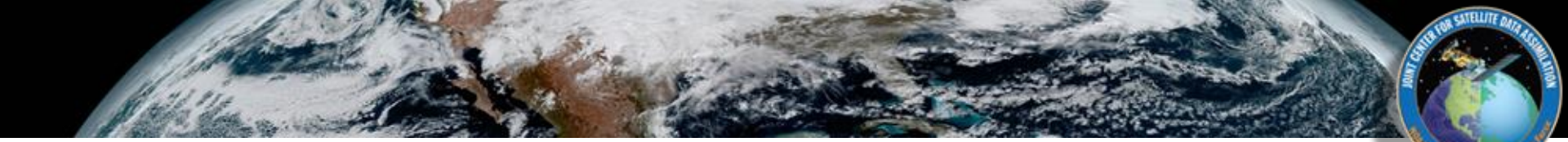
- VarBC learns systematic bias as $\sum_i \beta_i p_i$ inside the cost function. It adapts per channel , cycles between analyses.

- QC is a composable: filters + actions +where + ObsFunctions, ordered automatically.

- The interface is a declarative YAML→ Powerful, runtime-flexible, and the easier than custom code.

Hyperlinks

- Observation Operators in UFO:
  https://jointcenterforsatellitedataassimilation-jedi-docs.readthedocs-hosted.com/en/latest/inside/jedi-components/ufo/obsops.html
- satwind.yaml:
  https://github.com/JCSDA/ufo/blob/develop/test/testinput/unit_tests/operators/satwind.yaml
- airs_aqua_gfs_HofX.yaml:
  https://github.com/JCSDA/ufo/blob/develop/test/testinput/instrumentTests/airs/airs_aqua_gfs_HofX.yaml
- gnssrobndnbam.yaml:
  https://github.com/JCSDA/ufo/blob/master/test/testinput/unit_tests/operators/gnssrobndnbam.yaml
- omi_aura.yaml:
  https://github.com/JCSDA/ufo/blob/master/test/testinput/unit_tests/operators/omi_aura.yaml
- gnssrobndnbam.yaml:
  https://github.com/JCSDA/ufo/blob/master/test/testinput/unit_tests/operators/gnssrobndnbam.yaml
- sfcMarine_gfs_HofX.yaml:
  https://github.com/JCSDA/ufo/blob/develop/test/testinput/instrumentTests/Surface_marine_obs/sfcMarine_gfs_HofX.yaml
- amsua_crtm_bc.yaml:
  https://github.com/JCSDA/ufo/blob/develop/test/testinput/unit_tests/predictors/amsua_crtm_bc.yaml
- src/ufo/filters/QCflags.h:
  https://github.com/JCSDA/ufo/blob/develop/src/ufo/filters/QCflags.h
- sonde_gfs_qc.yaml:
  https://github.com/JCSDA/ufo/blob/develop/test/testinput/instrumentTests/Sonde/sonde_gfs_qc.yaml
- obserrorcrossvarcov.yaml:
  https://github.com/JCSDA/ufo/blob/develop/test/testinput/unit_tests/errors/obserrorcrossvarcov.yaml
- Conventional Tables:
  https://jointcenterforsatellitedataassimilation-jedi-docs.readthedocs-hosted.com/en/latest/inside/conventions/tbls.html#convention-tables
- jedi-bundle/CMakeLists.txt:
  https://github.com/JCSDA/jedi-bundle/blob/develop/CMakeLists.txt



Questions?

JEDI Documentation: https://jointcenterforsatellitedataassimilation-jedi-docs.readthedocs-hosted.com/en/latest/index.html
JEDI Forum: https://forums.jcsda.org/ (requires account to post/comment)
Github: https://github.com/JCSDA (public)