# Hello! 👋

Pieter De Vis

Sam Hunt

Maddie Stedman

Enis Gerxhalija

**Workshop Materials**

https://www.comet-toolkit.org/user-guide/training/met4eo/

NPL National Physical Laboratory

# 📋 CoMet Agenda

1. **Presentation 1:** *CoMet Toolkit Introduction (15 min)*
2. **Exercise 1:** *“Punpy Uncertainty Propagation” (15 min)*
3. **Exercise 2:** “Punpy with *Error Correlation” (15 min)*

*Break (30 min)*

5. **Exercise 3:** *“Spectrometer example” (35 min)*
6. **Presentation 2:** *“UNC & obsarray” (20 min)*
7. **Exercise 4:** *“Multidimensional datasets” (35 min)*
8. **Exercise 5:** *“Handling error correlation in time series” (15 min – if time allows)*

*Tomorrow:*

5. **Exercise 6:** *“Satellite validation using RadCalNet and HYPERNETS” (45 min)*

NPL National Physical Laboratory

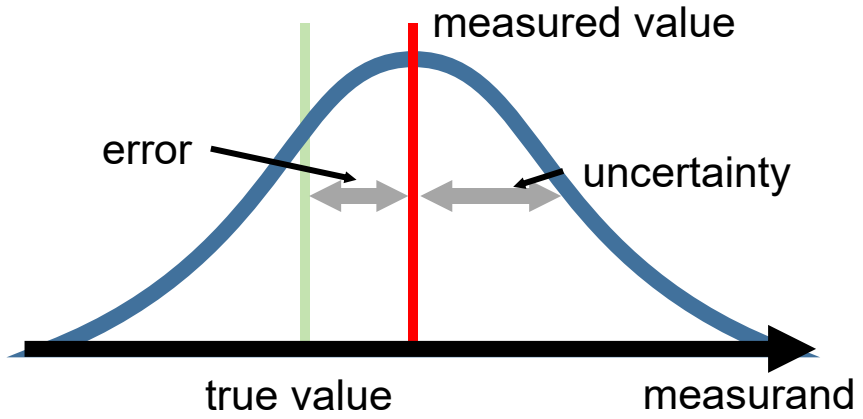
# The CoMet Toolkit – Uncertainties Made Easy

Pieter De Vis, Sam Hunt, Maddie Stedman,
Rasma Ormane, Enis Gerxhalija
***National Physical Laboratory***

Met4EO Uncertainty Training - 21/05/2026

# Uncertainties are important but can be complex

- EO data need uncertainties for credible and reliable interpretation
- Data is often affected by multiple uncertainty components. For some of these, the errors are correlated
- Understanding error-covariances in the data is key to combine observations
- The CoMet Toolkit (“Community Metrology Toolkit”) has been developed to enable easy handling and processing of dataset error-covariance information



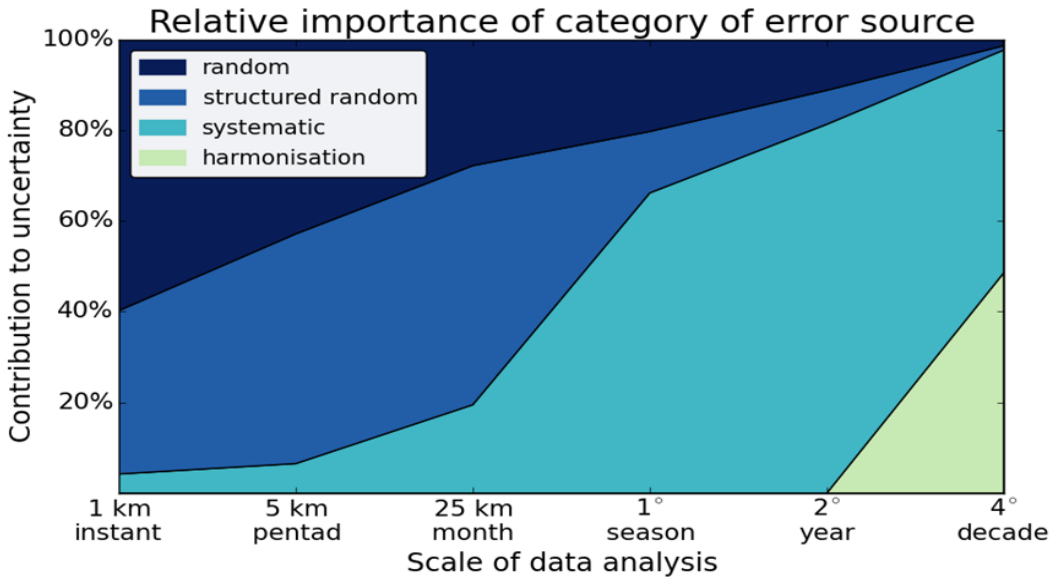




Image credit: http://dx.doi.org/10.6084/m9.figshare.1403409

# CoMet Toolkit

Python Tools

Uncertainty Handling

Open Source

Tested

HYPERNETS

truths

Applied

NPL National Physical Laboratory

# CoMet Toolkit

## punpy

Propagation UNcertainties in Python

## obsarray

Handling uncertainty and error-covariance in datasets

## UNC Specification

Uncertainty metadata naming conventions

## comet_maths

CoMet mathematical algorithms and interpolation tools

## curepy BETA

Comet Uncertainties for REtrievals in PYthon

NPL National Physical Laboratory

# A Metrological Approach

Uncertainties are evaluated and expressed following ***QA4EO*** *Five Steps*, a framework which employs the principles of metrology.


**Define the measurand and measurement function**

**Establish the traceability with a diagram**

**Evaluate each source of uncertainty and fill in an effects table**

**Calculate the data product and uncertainties**

**Record information for users**

punpy

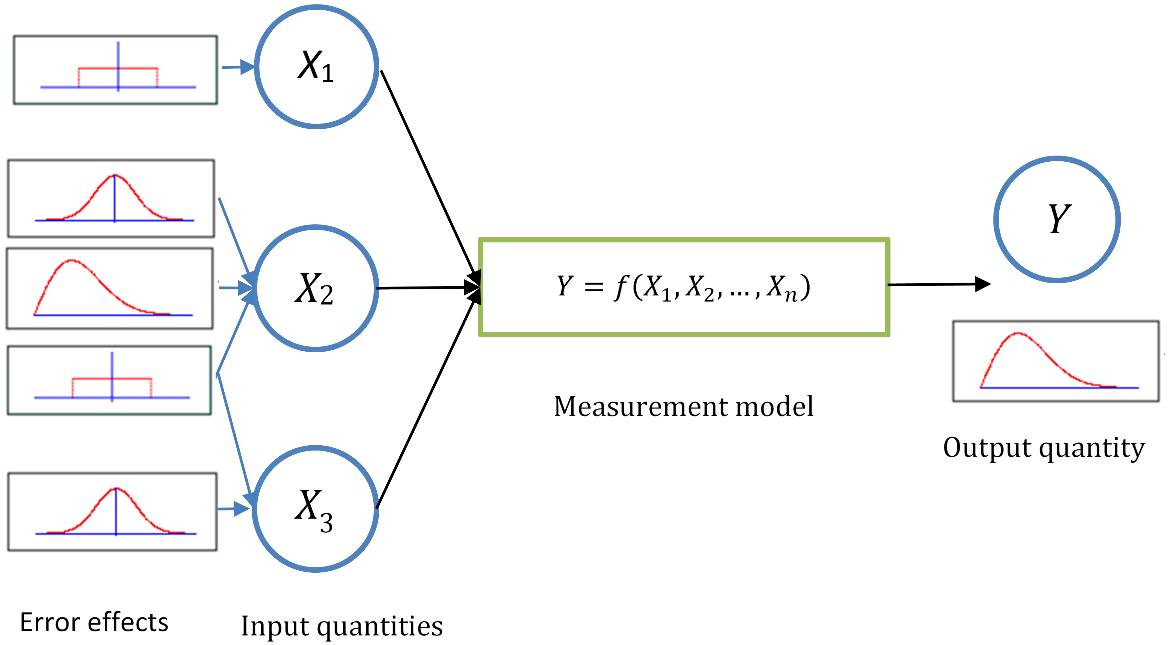
obsarray

QA4EO

NPL National Physical Laboratory

# Propagating Uncertainties with punpy

- Uncertainty and error correlation on input quantities are propagated to measurand
- Implements ***Monte Carlo*** and ***Law of Propagation of uncertainties***
- *Works for any measurement function in python that takes input quantities as arguments and returns measurand*



# Storing Uncertainties with obsarray

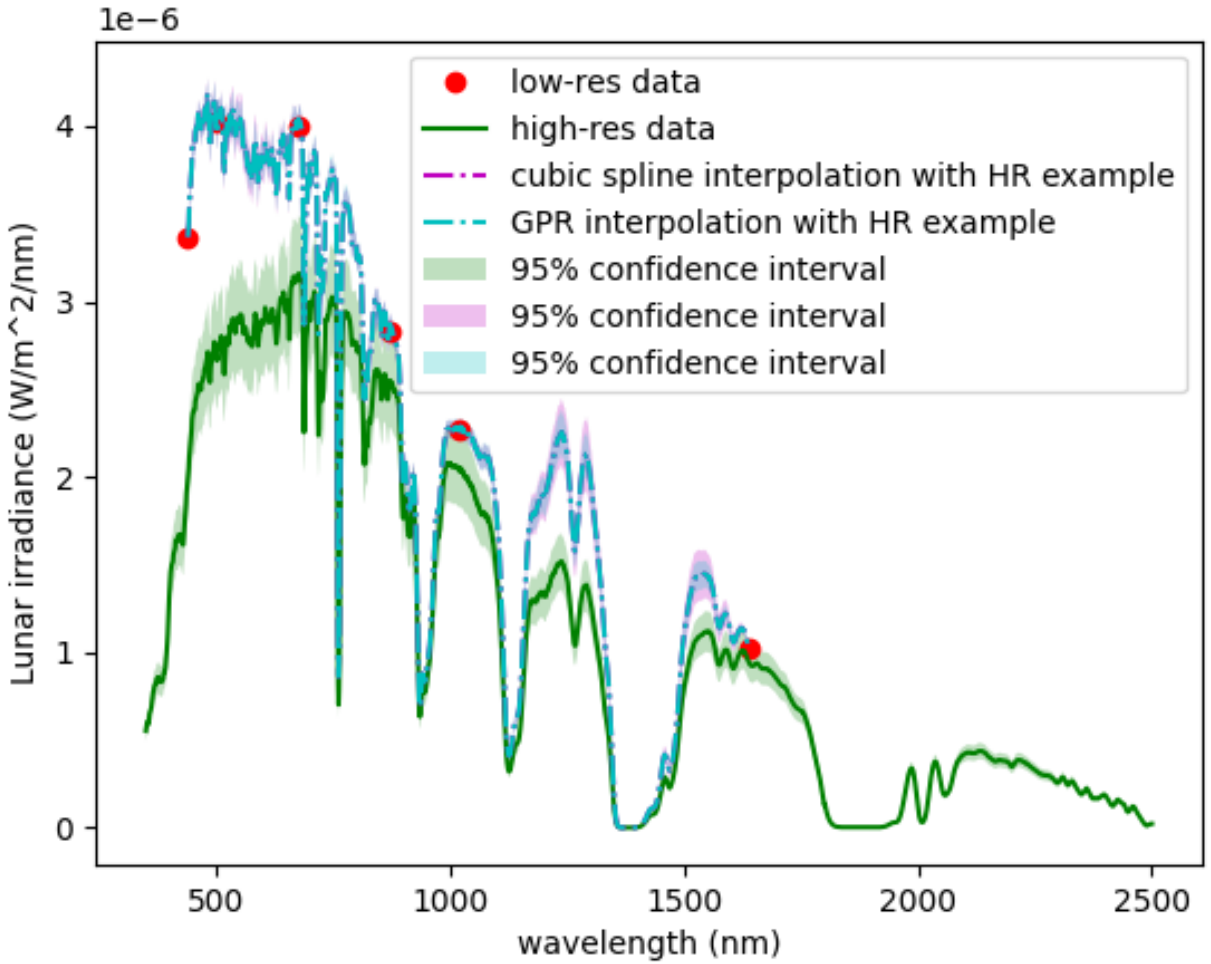
- **obsarray** is an extension to **xarray** to support defining, storing and interfacing with measurement data, uncertainties and error correlation.
- Builds on a standardised set of metadata – the **UNC specification**.
- Allows to interface with uncertainty information through .unc accessor
- It is designed to work well with **netCDF** files and for the **Earth Observation** community.

**Plugs straight into punpy for propagation through measurement functions!**

# Interpolation Uncertainties

comet_maths

- 1D interpolation with uncertainties
  - Input data uncertainties
  - Model uncertainties
- 1D Interpolate between low uncertainty anchor-points using higher resolution data (with poor absolute calibration)
- Comet_maths also handles all linear algebra and sample generation for other CoMet tools



# Retrieval Uncertainties with curepy

- *Beta version recently made open-source*

- ***y**=f(**x**), where **y** & u(**y**) are measured and x is retrieved*

- *Cost function:* $$\chi^2(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_a)^T \mathbf{S}_a^{-1}(\mathbf{x} - \mathbf{x}_a) + \frac{1}{2}(\mathbf{y} - \mathbf{f}(\mathbf{x}))^T \mathbf{S}_\varepsilon^{-1}(\mathbf{y} - \mathbf{f}(\mathbf{x})).$$

- *Inverse modelling implemented using:*
  - *Optimal Estimation (OE)*
    - *Jacobians are used to propagate the uncertainties once best fit is found*
  - *Markov Chain Monte Carlo (MCMC)*
    - *Generates random draws*
    - *Accept or reject sample based on probability from $\chi^2$*
    - *Final sample of state vector (x) has realistic PDF => Uncertainty and error correlation*

# CoMet Application Examples





NPL
National Physical Laboratory

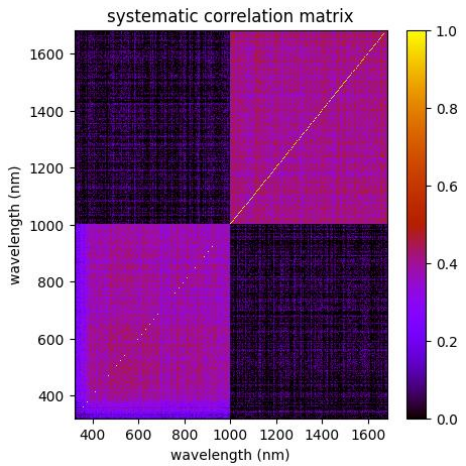
# CoMet Toolkit in Action

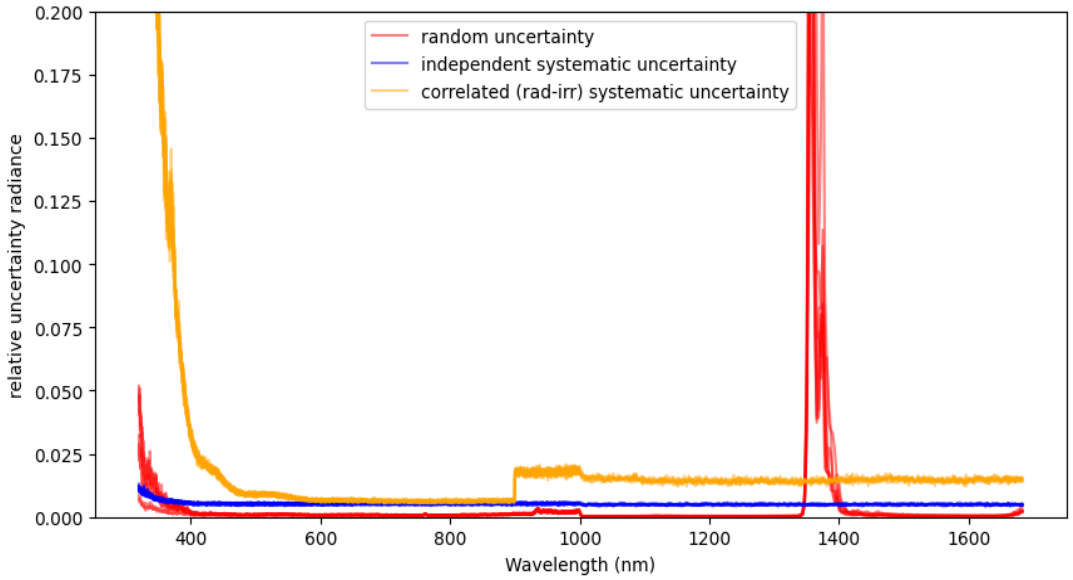
- Validated against **NIST** uncertainty engine:
  https://colab.research.google.com/github/comet-toolkit/comet_training/blob/main/NIST_example.ipynb

- *CoMet* is used in various other projects, such as **QA4EO**, **HYPERNETS**, **CHIME L2**, **FLEX** validation, **TRUTHS** science studies, **LIME**, **FRM4SOC**, **RPV4PICS**

- **Example** from *hypernets_processor*:

  *Hypernets is an automated network of in-situ instruments measuring reflectance for L2 satellite validation*



# CoMet Release

- V1.0 of Comet toolkit has been released as **open source** toolkit:
  - www.comet-toolkit.org
  - github.com/comet-toolkit

- Accompanied by training material (**Jupyter** notebooks hosted on google colab):
  - www.comet-toolkit.org/user-guide/examples

- Documentation & ATBD for individual tools:
  - obsarray.readthedocs.io/en/latest/
  - punpy.readthedocs.io/en/latest/
  - comet-maths.readthedocs.io/en/latest/
  - curepy.readthedocs.io/en/latest/

NPL National Physical Laboratory

# 📈 Outlook

- ❑ **Current release** will be presented De Vis & Hunt (in prep)

- ❑ Looking to continue to expand the use cases the developed tools
  - Aiming to enable uncertainty propagation through **any python measurement function**
  - 📧 Please get in touch if you are interested!

- ❑ This has been our first step into this way of working, **many more ideas in the roadmap/ under development**
  - CEOS PVP Cal/Val pipeline, BRDF tool, etc.

MetEOR

www.comet-toolkit.org/meteor/

NPL National Physical Laboratory

# ✅ Summary

- The **CoMet toolkit** is an open-source software project to develop Python tools for the handling of error-covariance information in the analysis of measurement data
- This toolkit is based on **robust metrology**, and makes dealing with complexities of uncertainties much easier
- Includes **obsarray**, **punpy** & **comet_maths** as initial offering, to be extended
- These tools are already being used operationally in various projects (e.g. Hypernets)

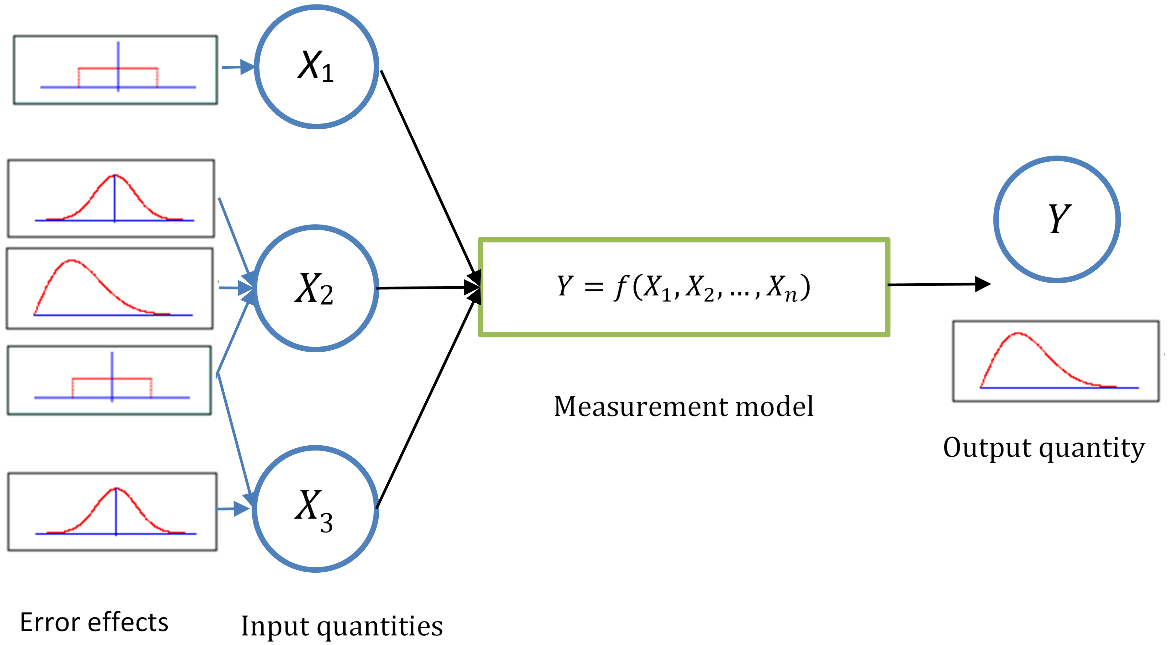
NPL National Physical Laboratory

# CoMet Exercises:


National Physical Laboratory

# Propagating Uncertainties with

- Python module for propagating **random**, **systematic** and **structured uncertainties** through any Python measurement function
- Flexible in terms of the **specified correlations** along given dimensions or between input quantities
- ***Monte Carlo*** and ***Law of Propagation of uncertainties*** methods available




NPL National Physical Laboratory

# Punpy as a Standalone Tool

- Simple user **interface**:
  1. Import punpy
  2. Define measurement function
  3. Create MC or LPU object
  4. Propagate uncertainties

```
import punpy
prop=punpy.MCPropagation(10000)
unc_measurand=prop.propagate_random(measurement_func,
      [input_qt1,input_qt2],[unc_qt1,unc_qt2])
```

- **Measurement function** are defined as python functions that take arrays as input quantities and return an array as measurand

- Many optional **keywords** for flexible functionality

  - *return_corr*
  - *Corr_between*
  - *Repeat_dims*
  - *Parallel_cores*
  - *Output_vars*
  - *…*

NPL National Physical Laboratory

# Exercise 1, 2 & 3

- Please go to https://www.comet-toolkit.org/user-guide/training/met4eo/

Mentimeter

**Code: 5335 9585**

https://www.menti.com/al7x3zbajvvk

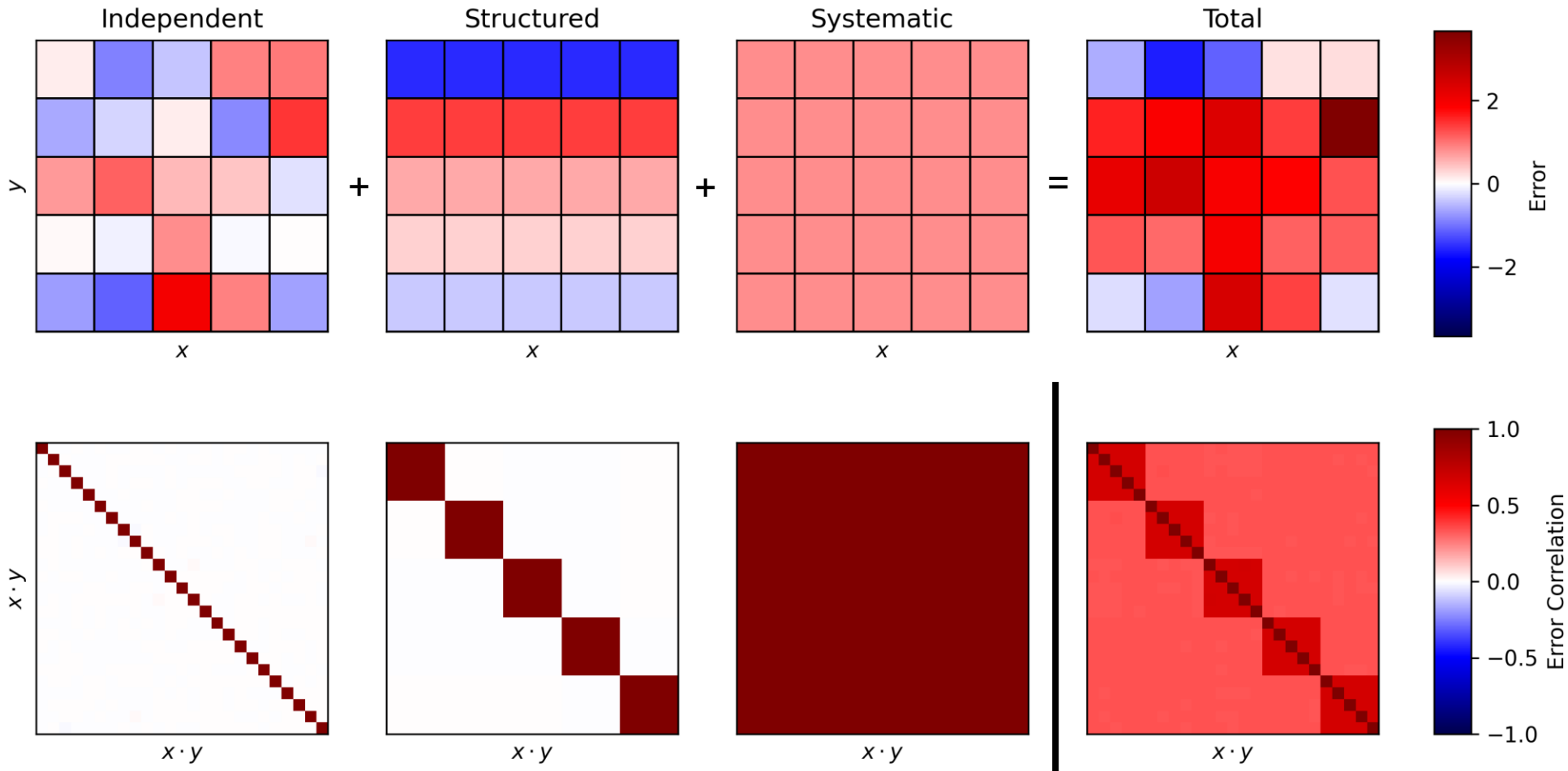
NPL National Physical Laboratory

# The CoMet Toolkit – UNC and obsarray

Sam Hunt, Pieter De Vis, Maddie Stedman,
Rasma Ormane, Enis Gerxhalija

***National Physical Laboratory***

# ND Datasets & Error-Correlation




- Simply parameterisable
- One term might dominate at different scales

- Parameterisable via other terms
- Same complexity at all scales

# Uncertainty metadata Naming Convention (UNC)

- Provide a standardised metadata format for uncertainty/error-covariance information.
- Enable compact representation of error-covariance matrices using parameterised metadata.

# Design Principles

### Scalability of Complexity

- The complexity of the metadata should align with the use case:
  - *Simple use cases* should be achievable with a *simple implementation*.
  - *Complex use cases* should be *possible without unnecessary restrictions*.

### Minimisation of Redundancy

- The metadata specification should *avoid duplication* of information to prevent potential inconsistencies.

### Human and Machine Readability

- Metadata must be c*omprehensible* for humans and *parsable* by machines.

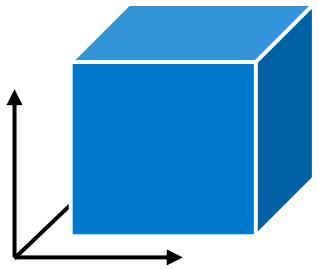
# Interaction with Other Standards

The specification is intended to contribute to and build upon an existing ecosystem of standards and best practices. In particular, the following are adhered to, to the extent possible:

- The understanding of uncertainty concepts defined in the JCGM GUM (Guide to the expression of uncertainty in measurement) suite of documents.
- The definition of uncertainty-related terminology defined in the JGCM VIM(International Vocabulary for Metrology).
- The NetCDF data model for creating self-describing, array-oriented scientific datasets.
- The Climate and Forecast (CF) conventions on metadata for weather and climate data.

# Uncertainty Variable Metadata

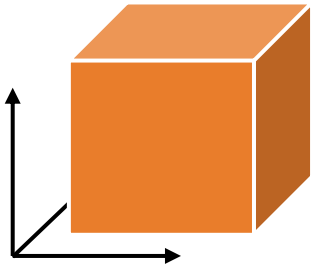
**Observation Variables**

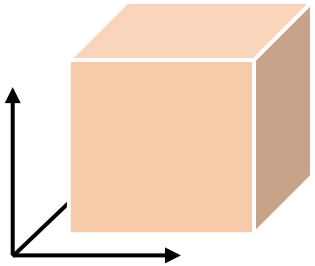


Metadata:
• Uncertainty Components

associated with

**Uncertainty Variables**





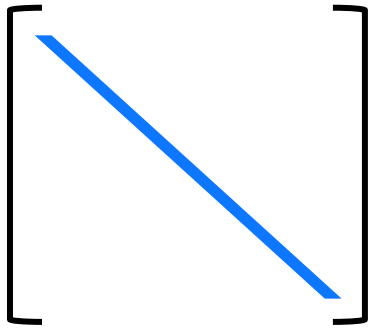
...

Metadata:
• PDF Shape (gaussian, …)
• Units (abs. or rel.)
• Error-Correlation…

# Error-Covariance Metadata

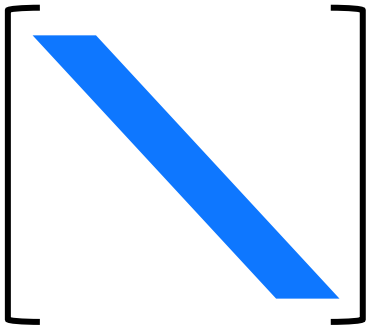
Error-covariance matrices grow quadratically with the size (N) of observation variables, making storage impractical for large datasets.

Instead, compact parameterisation of error-correlation matrices by defining specific structure, e.g.:



**Identity**

Effects:
• Random noise

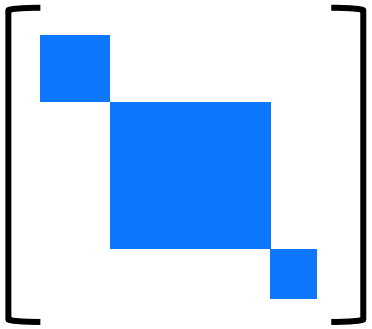


**Banded**

Effects:
• Averaging

**Banded**

Effects:
• Systematic



**Bloc Diagonal**

Effects:
• Local systematics

# UNC Progress – Towards CF Integration

```
variables:
  float temperature(time, lat, lon);
    temperature:unc_comps=["u_calibration", "u_noise"];
    temperature:units="K"
  float u_calibration(time, lat, lon);
    u_calibration:units="K";
    u_calibration:pdf_shape="rectangular";
    u_calibration:err_corr_dim1_name=["lat", "lon"];
    u_calibration:err_corr_dim1_form="systematic";
    u_calibration:err_corr_dim2_name="time";
    u_calibration:err_corr_dim2_form="err_corr_matrix";
    u_calibration:err_corr_dim2_params=["err_corr_calibration_time"];
  float u_noise(time, lat, lon);
    u_calibration:err_corr_dim1_name=["time", "lat", "lon"];
    u_calibration:err_corr_dim1_form="random";
  float err_corr_calibration_time(time, time);
```

```
variables:
    float sst(time, lat, lon) ;
        sst:standard_name = "sea_surface_temperature" ;
        sst:units = "K" ;
        sst:units_metadata = "temperature: on_scale" ;
        sst:uncertainty_variables = "unc" ;
    float unc(lat, lon) ;
        unc:error_correlation = "
                    lat: correlations
                    time: uncorrelated
                    lon: bell_shaped_relative (n: 15 sigma: sigma)" ;
    float correlations(lat, lat1) ;
    float sigma ;
        sigma:standard_name = "sea_surface_temperature" ;
        sigma:units = "K" ;
        sigma:units_metadata = "temperature: difference" ;
```

## UNC Spec v0.1

- Original proposal ~2023
- Implemented in e.g., Hypernets

## Towards CF-UNC Spec v1.0

- Update with support from Reading, NCAS
  - More idiomatic CF-style syntax

# UNC Roadmap

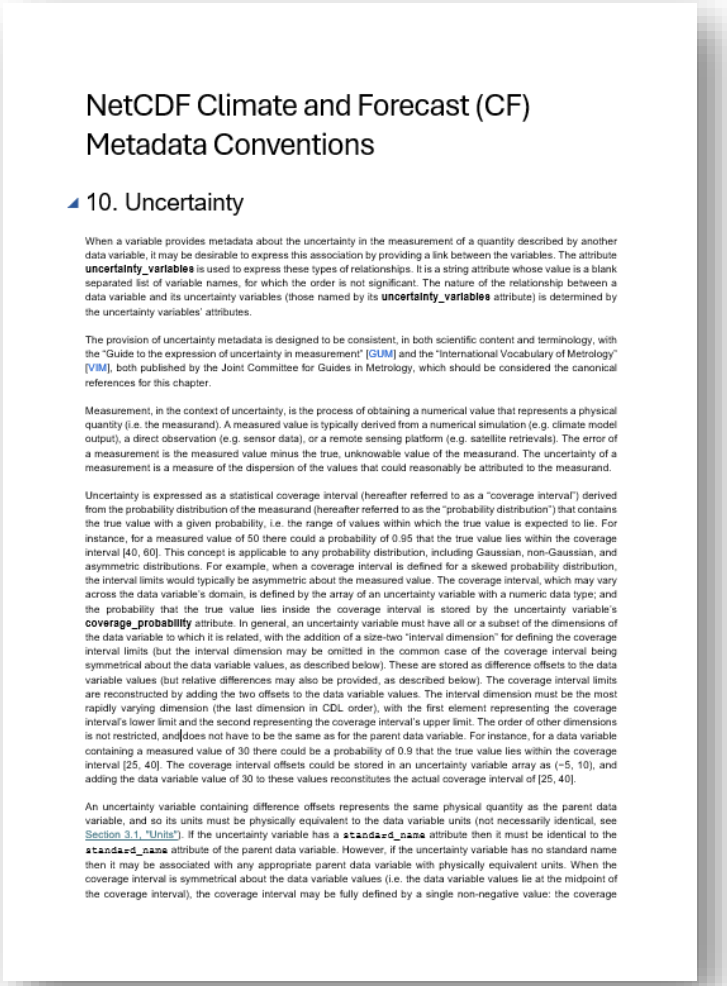
Alpha Testing – ends March 2026

- Prepared as a draft chapter to CF Conventions
- Test implementation by Hypernets (updated) and CCI SST

**Beta Testing – ends July 2026**

- Open testing
  - Do you want to participate?

Proposal to CF ~September 2026



NetCDF Climate and Forecast (CF) Metadata Conventions

10. Uncertainty

When a variable provides metadata about the uncertainty in the measurement of a quantity described by another data variable, it may be desirable to express this association by providing a link between the variables. The attribute **uncertainty_variables** is used to express these types of relationships. It is a string attribute whose value is a blank separated list of variable names, for which the order is not significant. The nature of the relationship between a data variable and its uncertainty variables (those named by its **uncertainty_variables** attribute) is determined by the uncertainty variables' attributes.

The provision of uncertainty metadata is designed to be consistent, in both scientific content and terminology, with the "Guide to the expression of uncertainty in measurement" [GUM] and the "International Vocabulary of Metrology" [VIM], both published by the Joint Committee for Guides in Metrology, which should be considered the canonical references for this chapter.

Measurement, in the context of uncertainty, is the process of obtaining a numerical value that represents a physical quantity (i.e. the measurand). A measured value is typically derived from a numerical simulation (e.g. climate model output), a direct observation (e.g. sensor data), or a remote sensing platform (e.g. satellite retrievals). The error of a measurement is the measured value minus the true, unknowable value of the measurand. The uncertainty of a measurement is a measure of the dispersion of the values that could reasonably be attributed to the measurand.

Uncertainty is expressed as a statistical coverage interval (hereafter referred to as a "coverage interval") derived from the probability distribution of the measurand (hereafter referred to as the "probability distribution") that contains the true value with a given probability, i.e. the range of values within which the true value is expected to lie. For instance, for a measured value of 50 there could a probability of 0.95 that the true value lies within the coverage interval [40, 60]. This concept is applicable to any probability distribution, including Gaussian, non-Gaussian, and asymmetric distributions. For example, when a coverage interval is defined for a skewed probability distribution, the interval limits would typically be asymmetric about the measured value. The coverage interval, which may vary across the data variable's domain, is defined by the array of an uncertainty variable with a numeric data type; and the probability that the true value lies inside the coverage interval is stored by the uncertainty variable's **coverage_probability** attribute. In general, an uncertainty variable must have all or a subset of the dimensions of the data variable to which it is related, with the addition of a size-two "interval dimension" for defining the coverage interval limits (but the interval dimension may be omitted in the common case of the coverage interval being symmetrical about the data variable values, as described below). These are stored as difference offsets to the data variable values (but relative differences may also be provided, as described below). The coverage interval limits are reconstructed by adding the two offsets to the data variable values. The interval dimension must be the most rapidly varying dimension (the last dimension in CDL order), with the first element representing the coverage interval's lower limit and the second representing the coverage interval's upper limit. The order of other dimensions is not restricted, and does not have to be the same as for the parent data variable. For instance, for a data variable containing a measured value of 30 there could be a probability of 0.9 that the true value lies within the coverage interval [25, 40]. The coverage interval offsets could be stored in an uncertainty variable array as (−5, 10), and adding the data variable value of 30 to these values reconstitutes the actual coverage interval of [25, 40].

An uncertainty variable containing difference offsets represents the same physical quantity as the parent data variable, and so its units must be physically equivalent to the data variable units (not necessarily identical, see Section 3.1, "Units"). If the uncertainty variable has a `standard_name` attribute then it must be identical to the `standard_name` attribute of the parent data variable. However, if the uncertainty variable has no standard name then it may be associated with any appropriate parent data variable with physically equivalent units. When the coverage interval is symmetrical about the data variable values (i.e. the data variable values lie at the midpoint of the coverage interval), the coverage interval may be fully defined by a single non-negative value: the coverage

# CoMet Packages:


NPL
National Physical Laboratory


# Measurement data handling in Python

- **obsarray** is an extension to **xarray** to support defining, storing and interfacing with measurement data – using the UNC specification.
- Also has functionality for defining flags following **CF Conventions**.
- It is designed to work well with **netCDF** files and for the **Earth Observation** community.

**Plugs straight into punpy for propagation through measurement functions!**

NPL National Physical Laboratory

obsarray

# Example Usage

First we build an example dataset that represents a time series of temperatures (for more on how do this see the xarray documentation).

```
In [1]: import numpy as np

In [2]: import xarray as xr

In [3]: import obsarray

# build an xarray to represents a time series of temperatures
In [4]: temps = np.array([20.2, 21.1, 20.8])

In [5]: times = np.array([0, 30, 60])

In [6]: ds = xr.Dataset(
   ...:     {"temperature": (["time"], temps, {"units": "degC"})},
   ...:     coords = {"time": (["time"], times, {"units": "s"})}
   ...: )
   ...:
```

# obsarray

## Example Usage

Uncertainty and error-covariance information for observation variables can be defined using the dataset's `unc` accessor, which is provided by **obsarray**.

```
# add random component uncertainty
In [7]: ds.unc["temperature"]["u_r_temperature"] = (
   ...:     ["time"],
   ...:     np.array([0.5, 0.5, 0.6]),
   ...:     {"err_corr": [{"dim": "time", "form": "random"}]}
   ...: )
   ...:

# add systematic component uncertainty
In [8]: ds.unc["temperature"]["u_s_temperature"] = (
   ...:     ["time"],
   ...:     np.array([0.3, 0.3, 0.3]),
   ...:     {"err_corr": [{"dim": "time", "form": "systematic"}]}
   ...: )
   ...:
```

NPL National Physical Laboratory

obsarray

# Example Usage

The defined uncertainty information then can be interfaced with, for example:

```
# get total combined uncertainty of all components
In [9]: ds.unc["temperature"].total_unc()
Out[9]:
<xarray.DataArray (time: 3)> Size: 24B
array([0.58309519, 0.58309519, 0.67082039])
Coordinates:
  * time     (time) int64 24B 0 30 60

# get total error-covariance matrix for all components
In [10]: ds.unc["temperature"].total_err_cov_matrix()
Out[10]:
<xarray.DataArray (time: 3)> Size: 72B
array([[0.34, 0.09, 0.09],
       [0.09, 0.34, 0.09],
       [0.09, 0.09, 0.45]])
Dimensions without coordinates: time, time
```

NPL National Physical Laboratory

# Punpy with digital effects tables

- **punpy** interfaces with **obsarray** to make uncertainty propagation as efficient and easy to use as possible
- ***propagate_ds()*** function returns an **obsarray** dataset with combined random, systematic and structured uncertainties on measurand

```python
from punpy import MeasurementFunction

# Define your measurement function inside a subclass of MeasurementFunction
class IdealGasLaw(MeasurementFunction):
    def meas_function(self, pres, temp, n):
        return (n *temp * 8.134)/pres

# create object of the measurement function class and specify the variable names
gl = IdealGasLaw(["pressure", "temperature", "n_moles"], "volume", yunit="m^3")

# propagate uncertainties on the input quantities in ds to measurand in ds_y
ds_y = gl.propagate_ds(ds)
```

NPL National Physical Laboratory

# Exercise 4

- Please go to https://www.comet-toolkit.org/user-guide/training/met4eo/

NPL National Physical Laboratory