Mentimeter Survey

Code: 5322 5551

Today’s Exercises

comet-toolkit.org/user-guide/training/lps

Welcome!

Introductions

Pieter De Vis

Sam Hunt

Maddie Stedman

Astrid Zimmermann

Mentimeter Survey

Code: 5322 5551

Today's Exercises

comet-toolkit.org/user-guide/training/lps

The CoMet toolkit – Uncertainties made easy

Pieter De Vis, Sam Hunt, Astrid Zimmermann, Maddie Stedman

National Physical Laboratory

LPS hands-on tutorial - 22/06/2025

Outline

- **Motivation & overview**
- Theoretical background
- Defining digital effects tables with *obsarray*
- Propagating uncertainties with *punpy*
- Release and conclusions

What is the CoMet toolkit?

- Set of Python tools for handling, propagating and storing uncertainty and error-correlation information.
- Handle multiple uncertainty components and propagate these through any Python measurement function
- Use quality assured code, with most of the complexities of uncertainties handled behind the scenes
- Open-source software project

CoMet

www.comet-toolkit.org

Uncertainties are important but can be complex

- EO data need uncertainties
  - credible and reliable interpretation
  - Affects retrievals
  - Interoperability
  - Trend detection

- 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

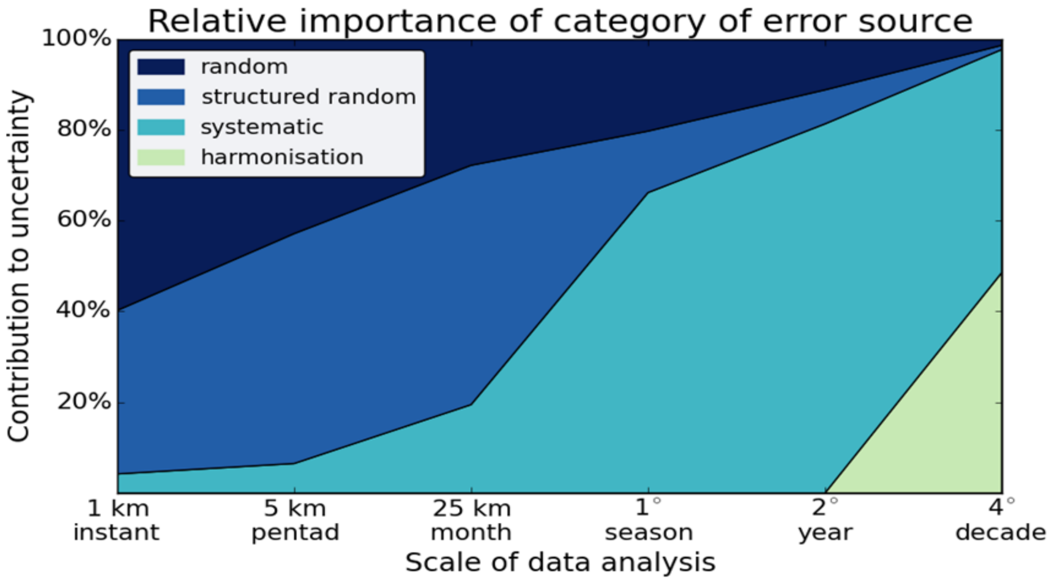


Image credit: http://dx.doi.org/10.6084/m9.figshare.1483409

IDEAS-QA4EO

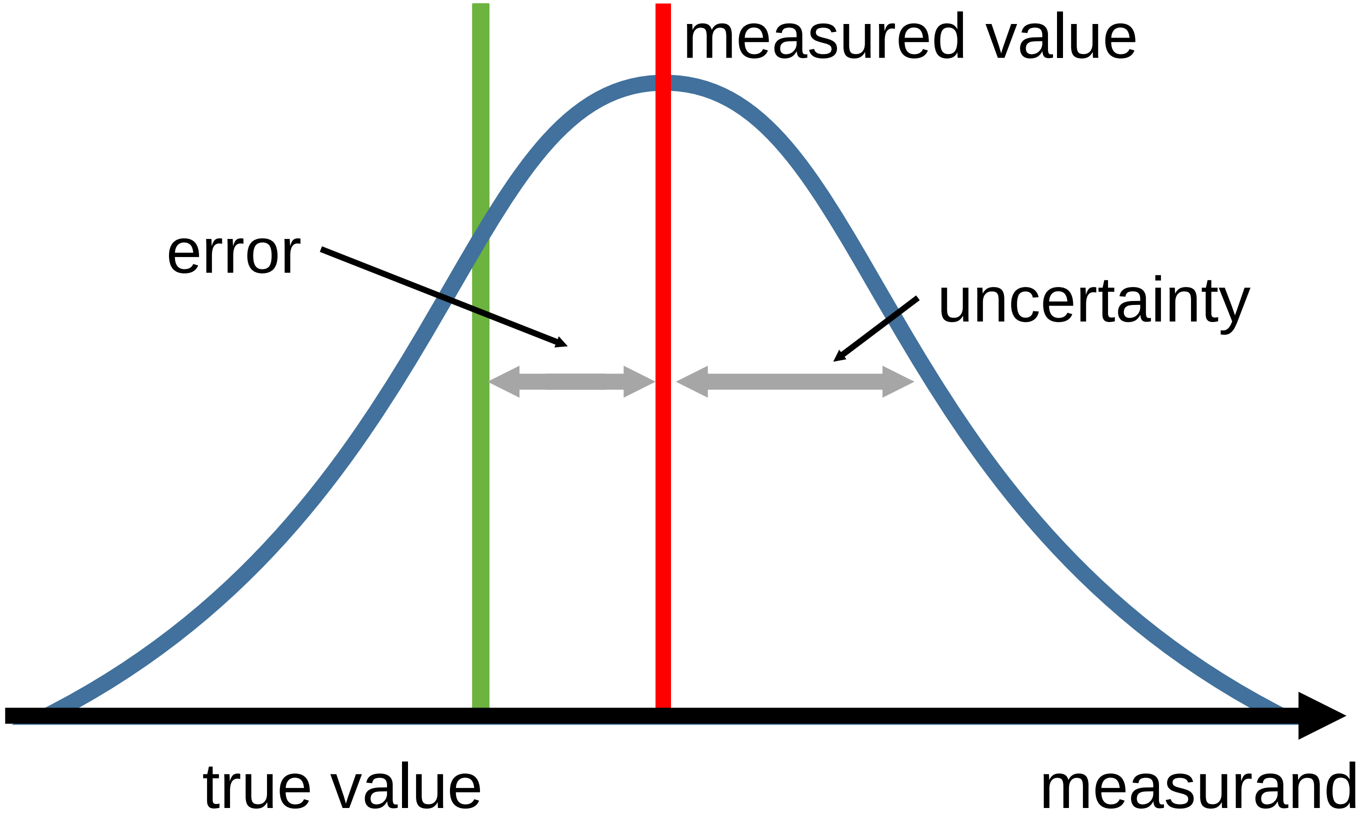
Tools overview

CoMet toolkit: Community Metrology toolkit

- **punpy**: Propagation UNcertainties in Python
- **obsarray**: Tool for storing and handling uncertainty and covariance in NetCDF files
- **comet_maths**: Comet mathematical algorithms and interpolation tools

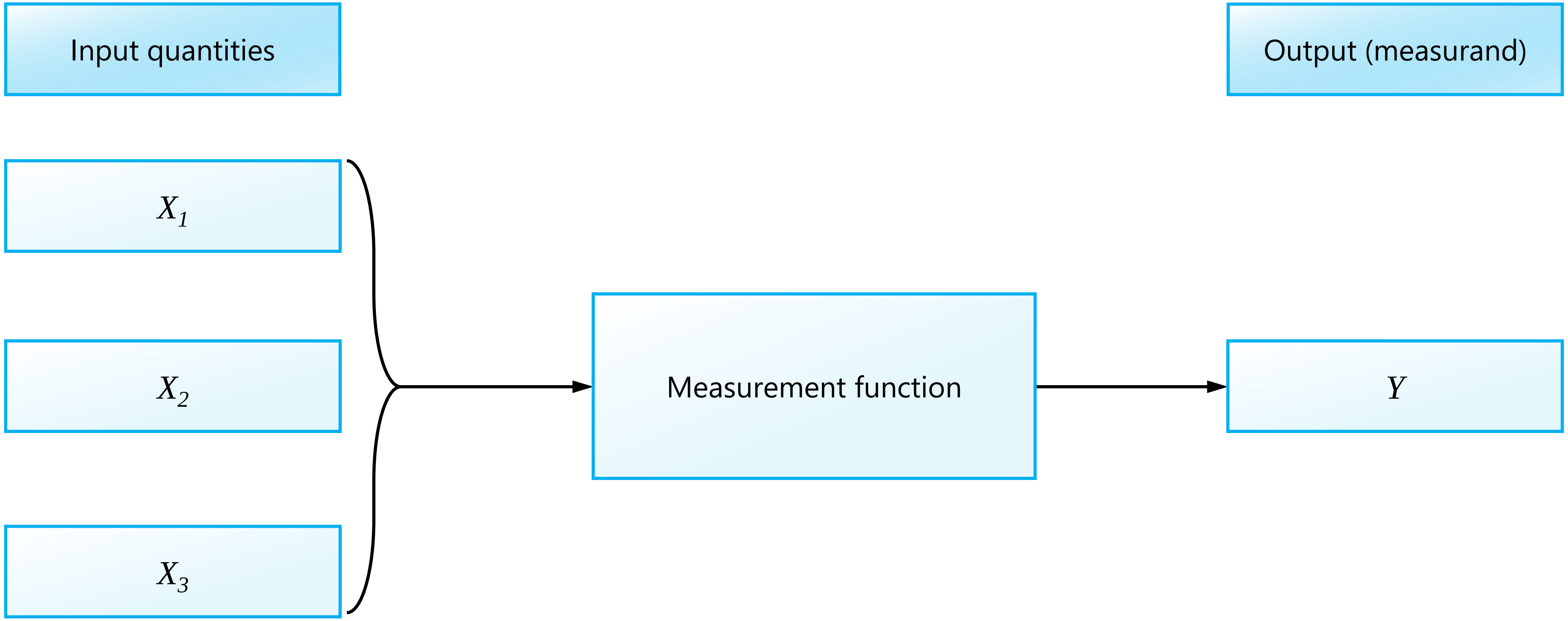
Outline

- Motivation & overview
- **Theoretical background**
- Defining digital effects tables with *obsarray*
- Propagating uncertainties with *punpy*
- Release and conclusions



IDEAS-QA4EO

I made a measurement of the measurand ‘radiance’ to get a measured value of 0.3 W m^{-2} sr^{-1} nm^{-1} with an associated standard uncertainty of 1 %.



Input quantities

X_1

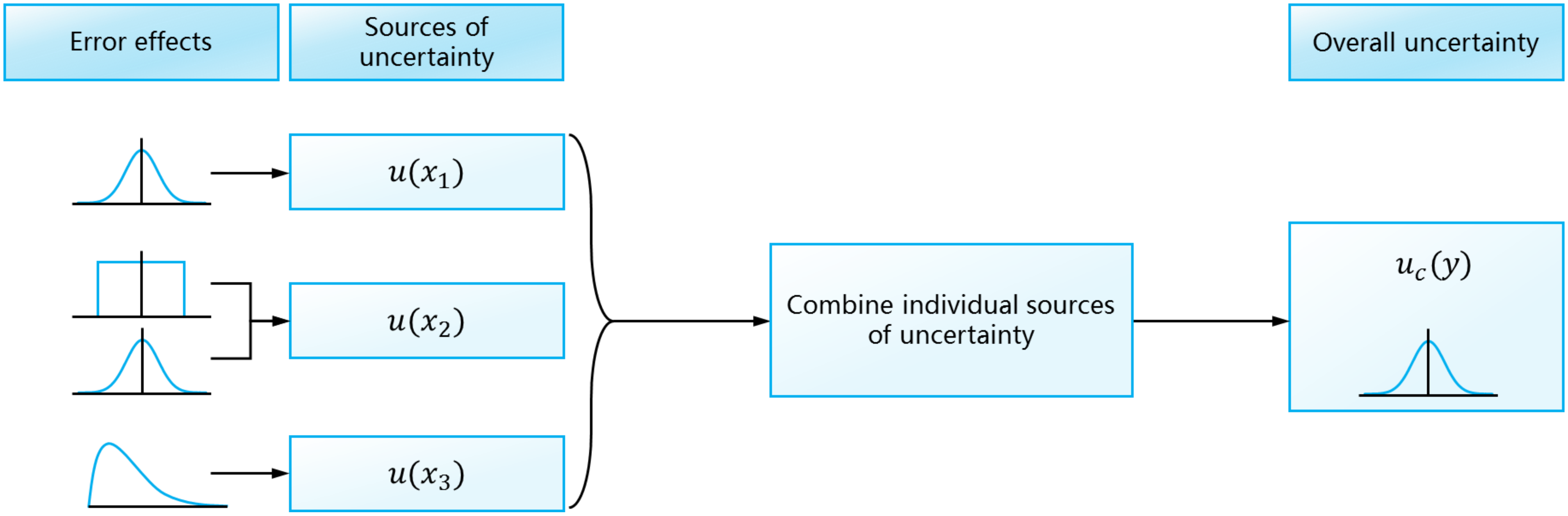
X_2

X_3

Measurement function

Output (measurand)

Y



Error effects

Sources of uncertainty

Overall uncertainty

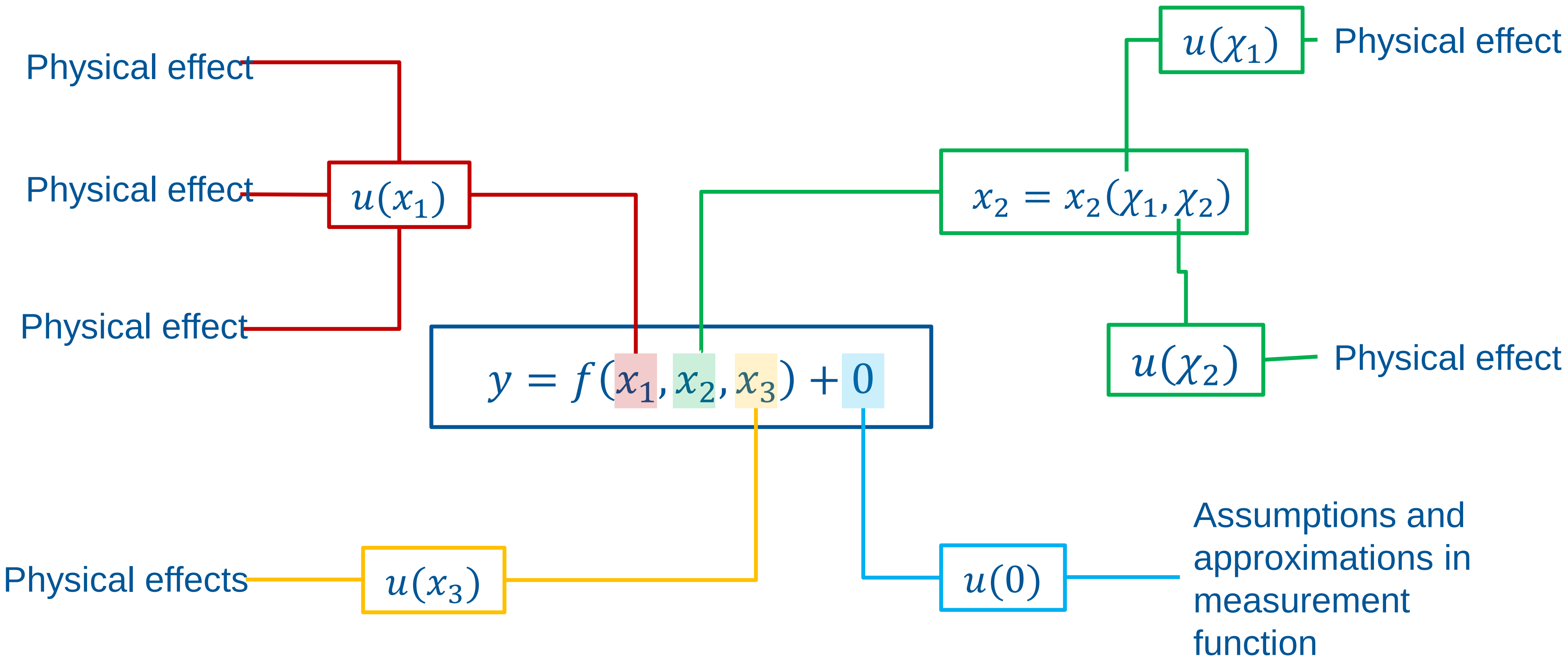
$u(x_1)$

$u(x_2)$

$u(x_3)$

Combine individual sources of uncertainty

$u_c(y)$



Physical effect

Physical effect

Physical effect

$u(x_1)$

$y = f(x_1, x_2, x_3) + 0$

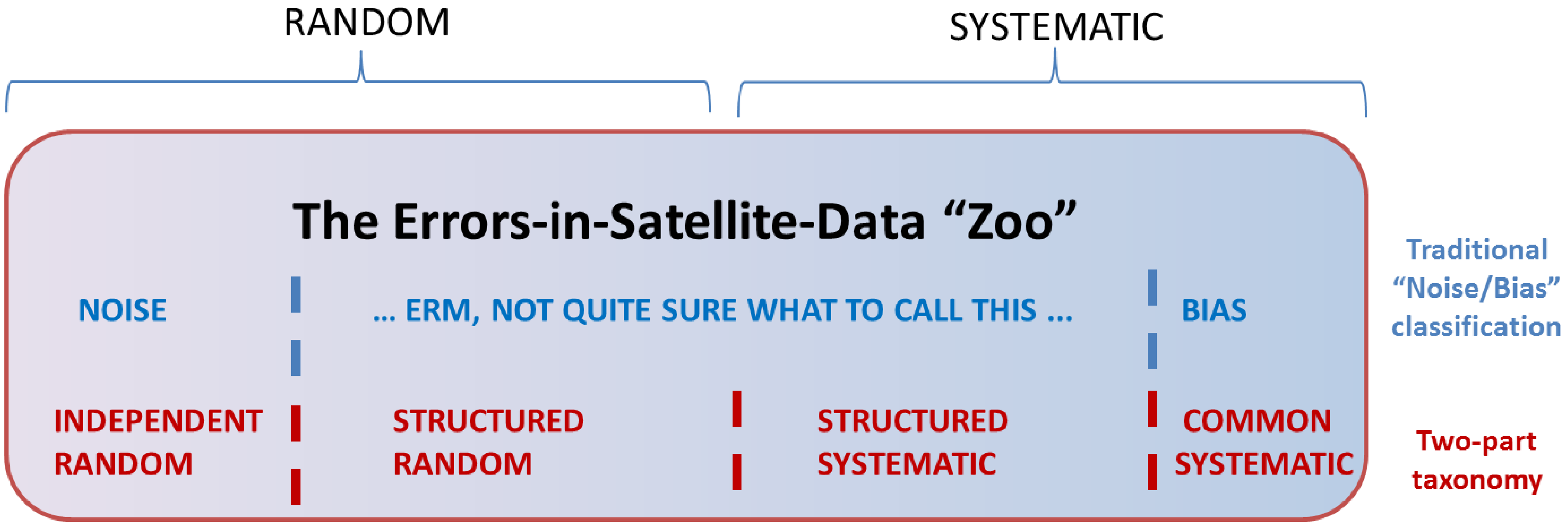
$x_2 = x_2(\chi_1, \chi_2)$

$u(\chi_1)$ Physical effect

$u(\chi_2)$ Physical effect

Physical effects $u(x_3)$

$u(0)$ Assumptions and approximations in measurement function



The Errors-in-Satellite-Data “Zoo”

RANDOM

SYSTEMATIC

NOISE

… ERM, NOT QUITE SURE WHAT TO CALL THIS …

BIAS

Traditional “Noise/Bias” classification

INDEPENDENT RANDOM

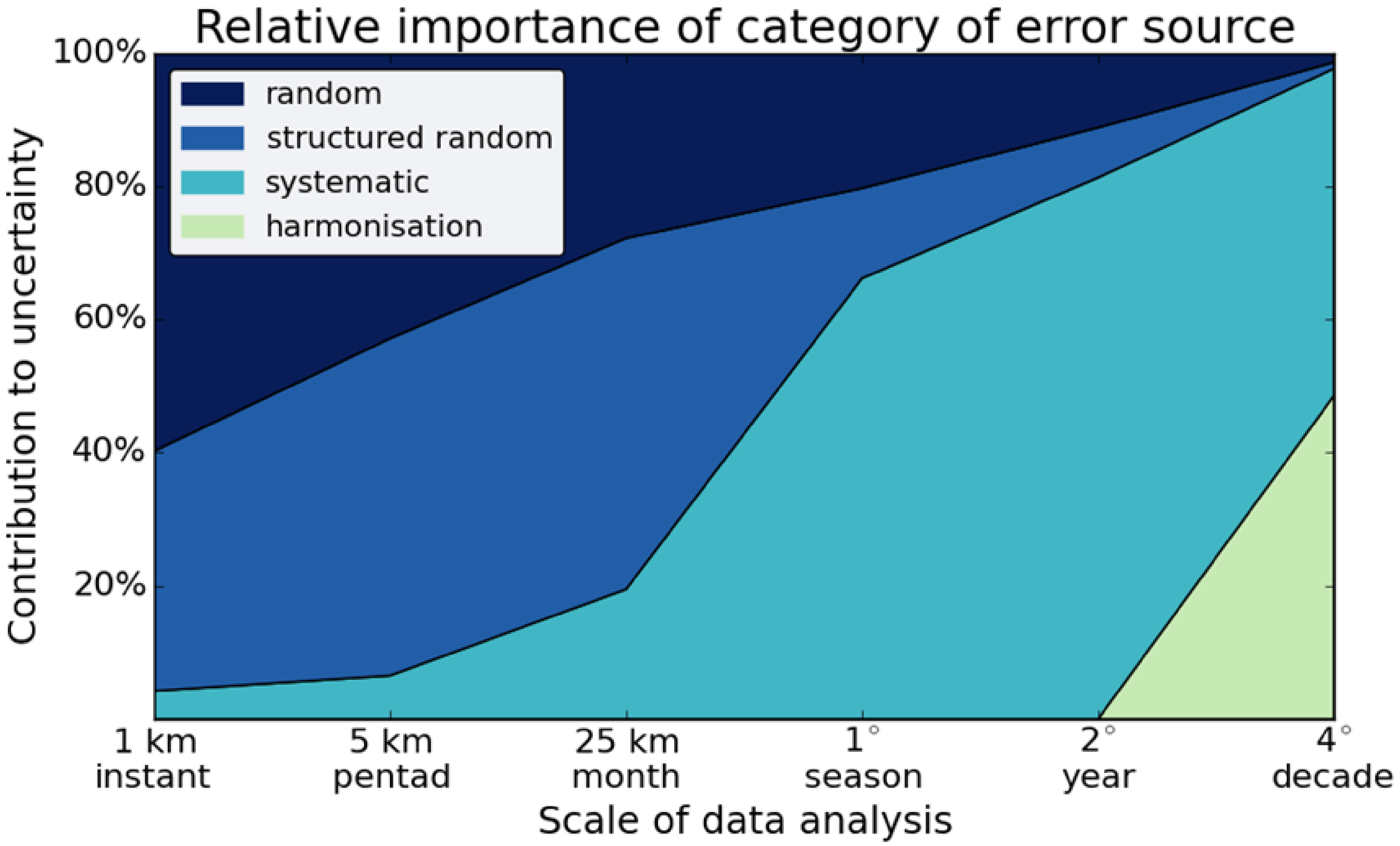
STRUCTURED RANDOM

STRUCTURED SYSTEMATIC

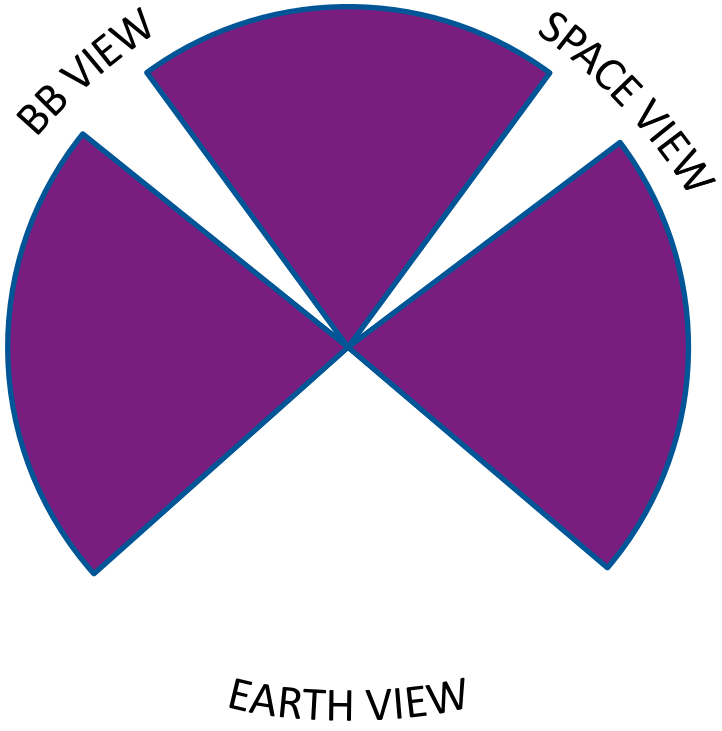
COMMON SYSTEMATIC

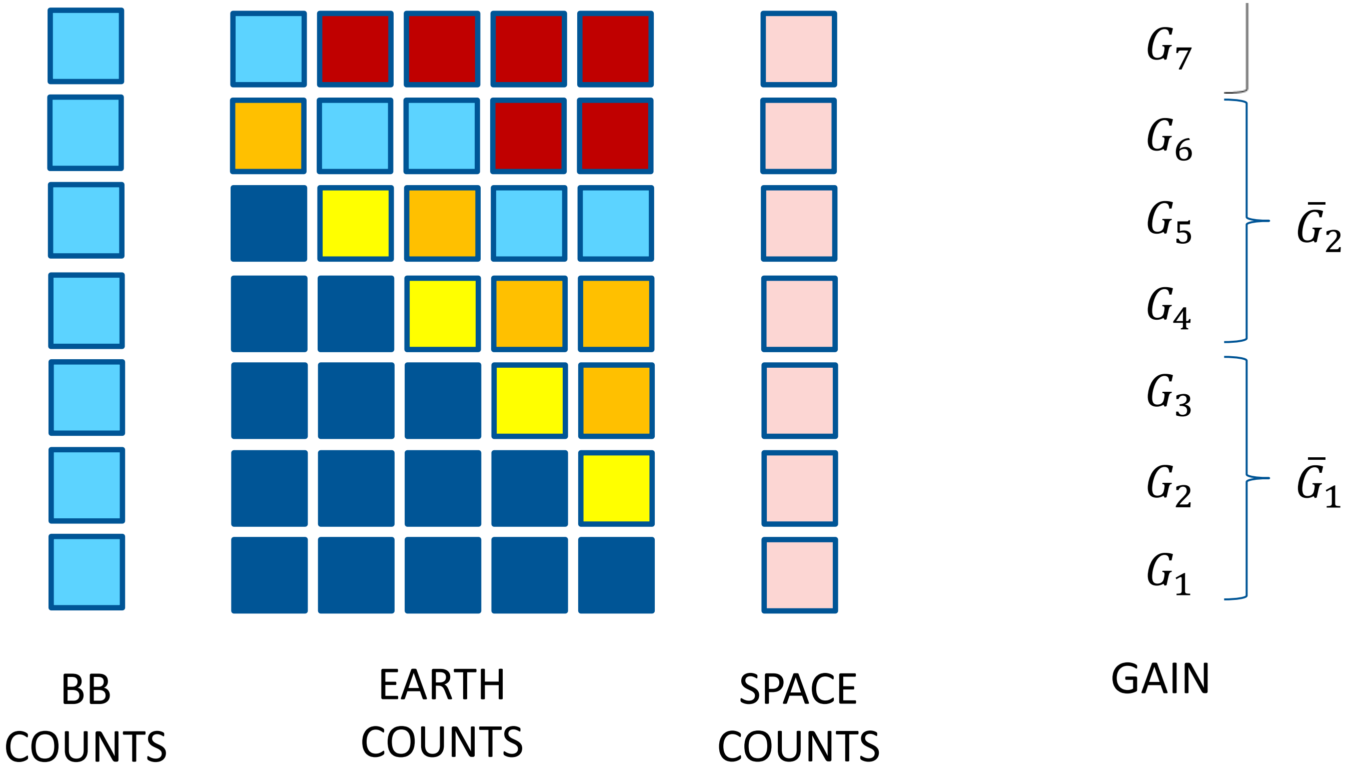
Two-part taxonomy

Error Correlation Structures: Why do we care?



Example: Structured Random Error Correlation in Cross Track Scanners





Outline

- Motivation & overview
- Theoretical background
- **Defining digital effects tables with *obsarray***
- Propagating uncertainties with *punpy*
- Release and conclusions

Encoded Observations

Geospatial data is encoded with complex metadata, though users typically never have to interact with it.

Example: Geocoding

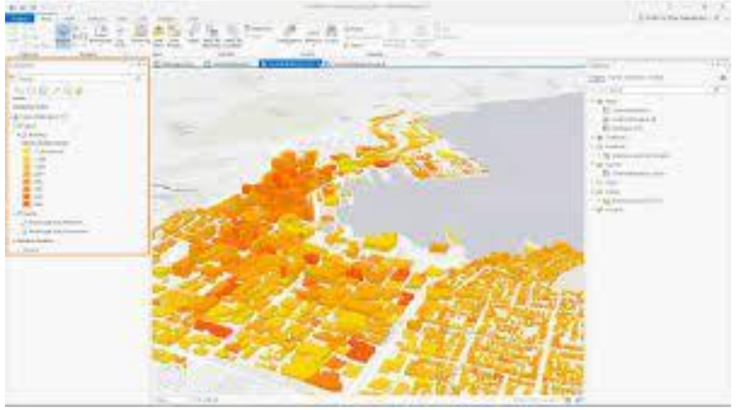
1. Data is accompanied with standardised metadata
2. Tools provide means to
   A. Interface with this information
   B. Interpret and make use of this information

OGC® Making location count.

WKT

ArcGIS

Cartopy



IDEAS-QA4EO

Encoded Observations

Geospatial data is encoded with complex metadata, though users typically never have to interact with it.

Why not take the same approach for error-covariance information for observations?

Encoded Observations

Geospatial data is encoded with complex metadata, though users typically never have to interact with it.

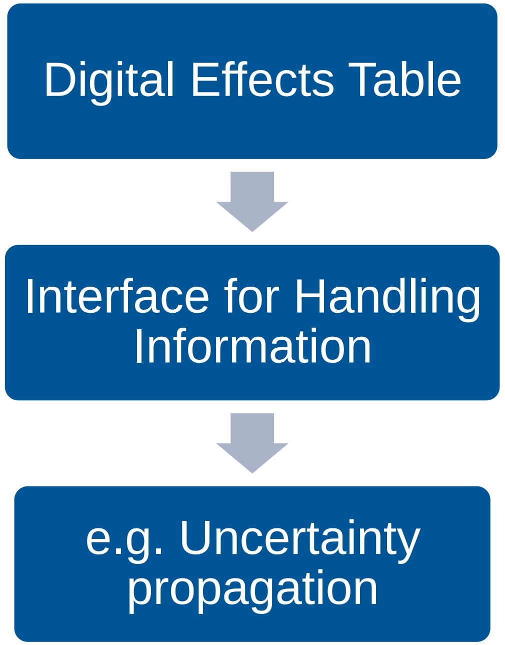
Parallel: Error-covariance encoding

1. Data is accompanied with standardised metadata
2. Tools provide means to
   A. Interface with this information
   B. Interpret and make use of this information

Fiduceo

obsarray

punpy


Digital Effects Table

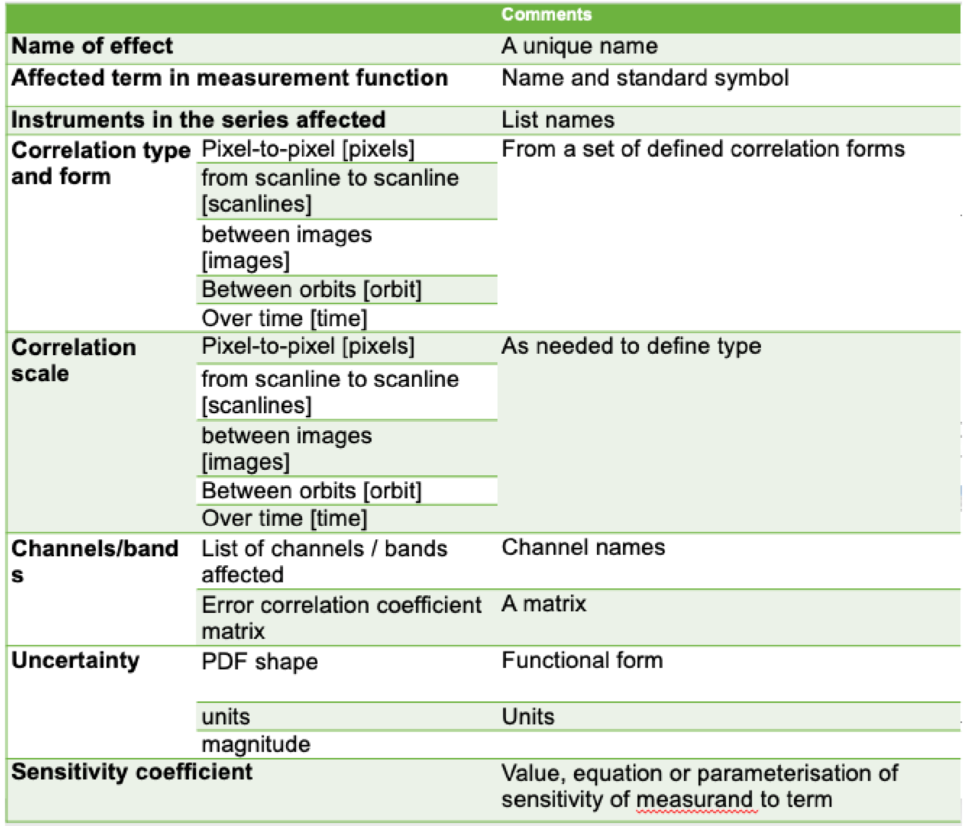
Interface for Handling Information

e.g. Uncertainty propagation

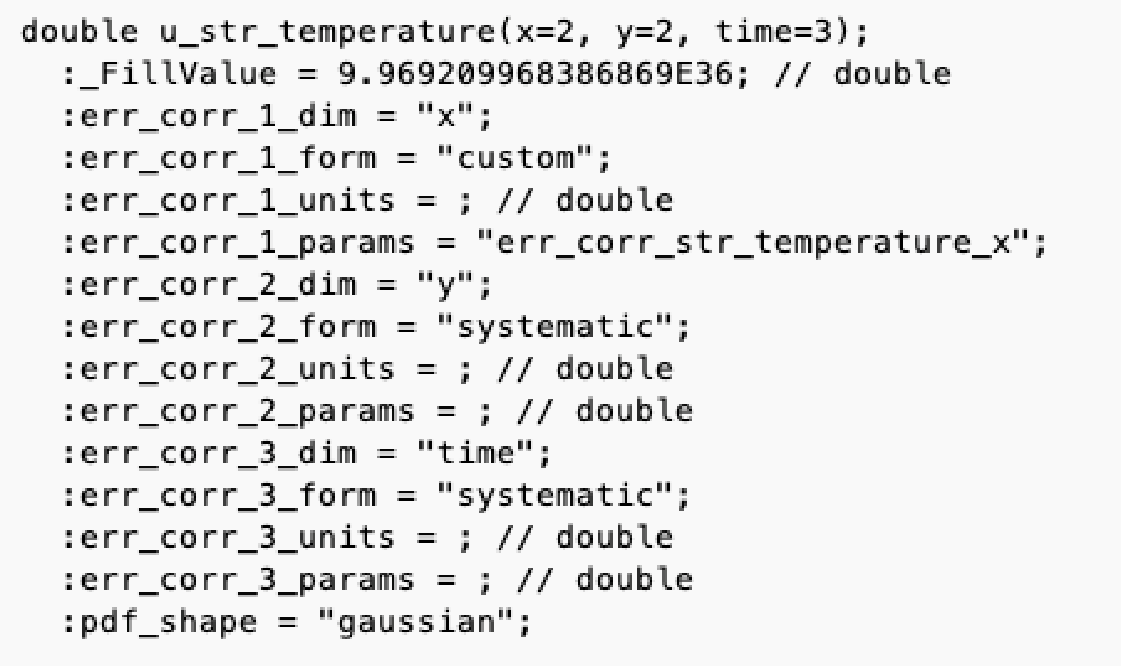
obsarray

- Python module that provides an extension to the widely used xarray package to interface with measurement error-covariance information encoded in datasets
- Includes templater that allows to build digital effects tables (xarray objects with uncertainties and covariance information), and save these as NetCDF files
- These digital effects tables can be passed to **punpy** to propagate uncertainties between products
- Once digital effects tables are defined, this effectively abstracts away the complexity of dealing with uncertainties

Standardised Error-Covariance Metadata: Digital Effects Tables



		Comments
Name of effect		A unique name
Affected term in measurement function		Name and standard symbol
Instruments in the series affected		List names
Correlation type and form	Pixel-to-pixel [pixels]	From a set of defined correlation forms
	from scanline to scanline [scanlines]	
	between images [images]	
	Between orbits [orbit]	
	Over time [time]	
Correlation scale	Pixel-to-pixel [pixels]	As needed to define type
	from scanline to scanline [scanlines]	
	between images [images]	
	Between orbits [orbit]	
	Over time [time]	
Channels/bands	List of channels / bands affected	Channel names
	Error correlation coefficient matrix	A matrix
Uncertainty	PDF shape	Functional form
	units	Units
	magnitude	
Sensitivity coefficient		Value, equation or parameterisation of sensitivity of measurand to term



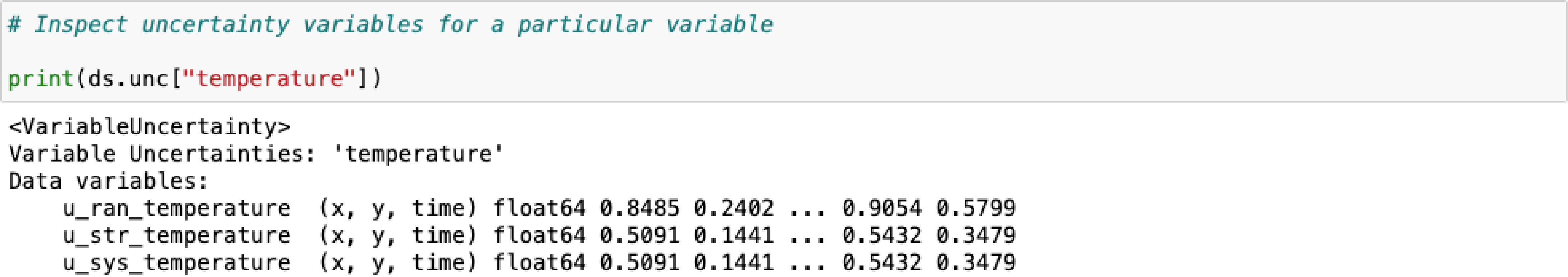
```
double u_str_temperature(x=2, y=2, time=3);
  :_FillValue = 9.969209968386869E36; // double
  :err_corr_1_dim = "x";
  :err_corr_1_form = "custom";
  :err_corr_1_units = ; // double
  :err_corr_1_params = "err_corr_str_temperature_x";
  :err_corr_2_dim = "y";
  :err_corr_2_form = "systematic";
  :err_corr_2_units = ; // double
  :err_corr_2_params = ; // double
  :err_corr_3_dim = "time";
  :err_corr_3_form = "systematic";
  :err_corr_3_units = ; // double
  :err_corr_3_params = ; // double
  :pdf_shape = "gaussian";
```

Effects Table

Digital Effects Table

Interface to Error-Covariance Metadata: obsarray

Once obsarray has been imported. Xarray datasets have a .unc property that allows to access the uncertainty information:



```
# Inspect uncertainty variables for a particular variable

print(ds.unc["temperature"])
```

```
<VariableUncertainty>
Variable Uncertainties: 'temperature'
Data variables:
    u_ran_temperature  (x, y, time) float64 0.8485 0.2402 ... 0.9054 0.5799
    u_str_temperature  (x, y, time) float64 0.5091 0.1441 ... 0.5432 0.3479
    u_sys_temperature  (x, y, time) float64 0.5091 0.1441 ... 0.5432 0.3479
```

It is also possible to return e.g. the combined total uncertainties and error correlation matrices for the relevant components.

Outline

- Motivation & overview

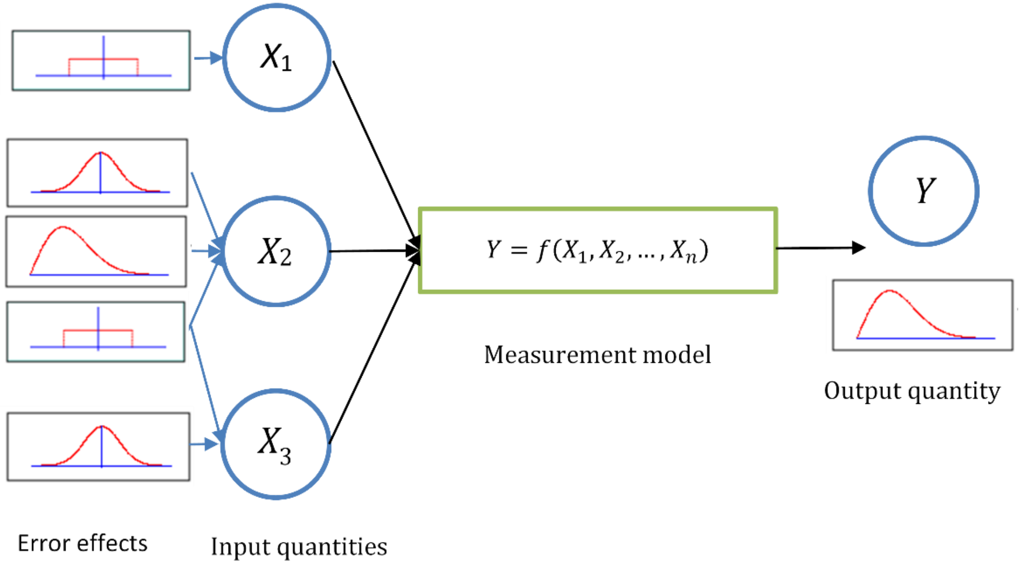
- Theoretical background

- Defining digital effects tables with *obsarray*

- **Propagating uncertainties with *punpy***

- Release and conclusions

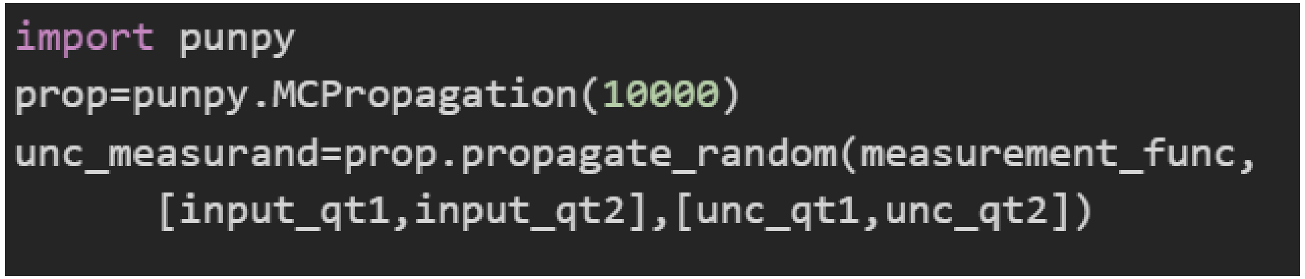
Propagating uncertainties with *punpy*



- 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

Punpy as standalone tool

- Simple user interface:
  - Import punpy
  - Define measurement function
  - Create MC or LPU object
  - Propagate uncertainties



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

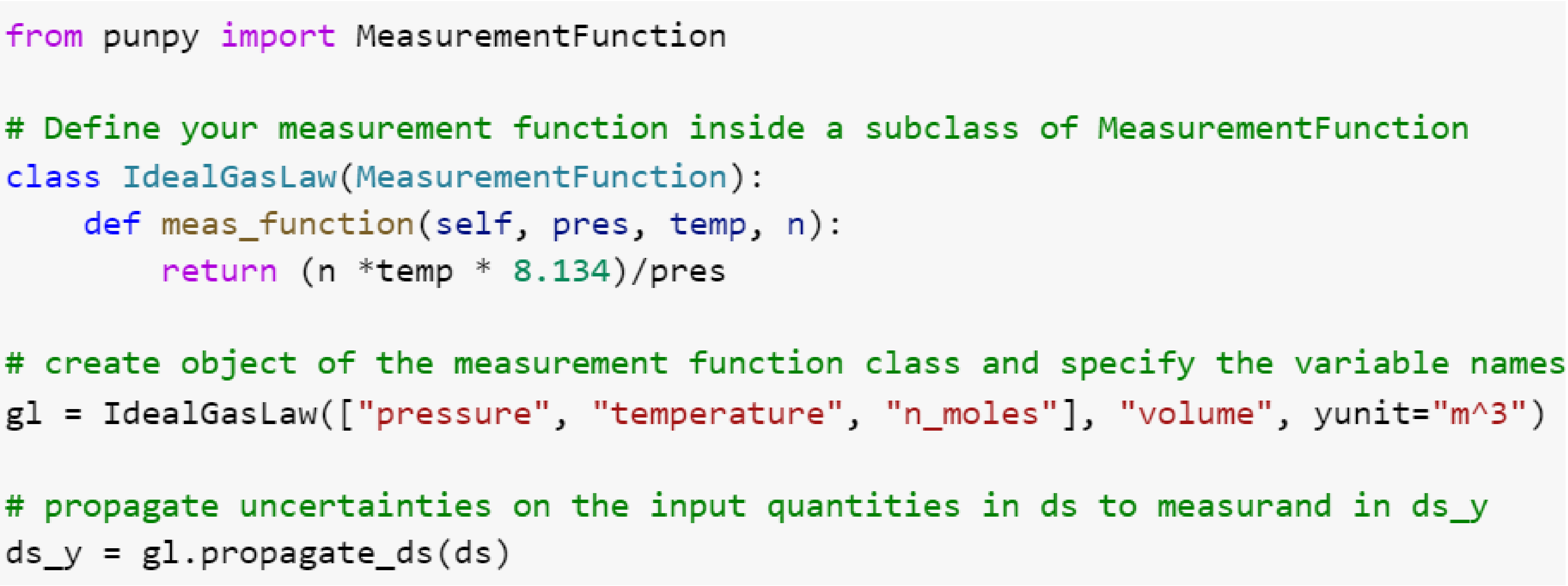
- Measurement function are defined as python functions that take arrays as input quantities and return an array as measurand

- Many optional keywords for flexible functionality
  - return_corr
  - Corr_between
  - Repeat_dims
  - Parallel_cores
  - Output_vars
  - …

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)
```

Outline

- Motivation & overview
- Theoretical background
- Defining digital effects tables with *obsarray*
- Propagating uncertainties with *punpy*
- **Release and conclusions**

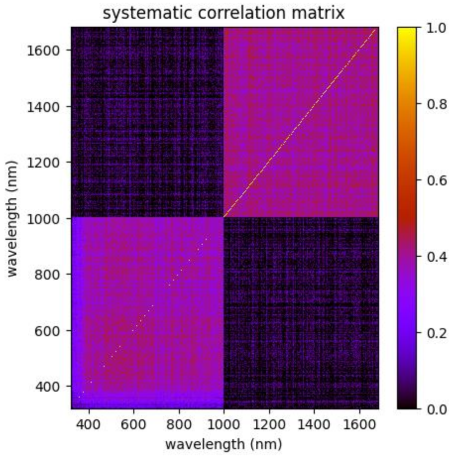
CoMet toolkit in action

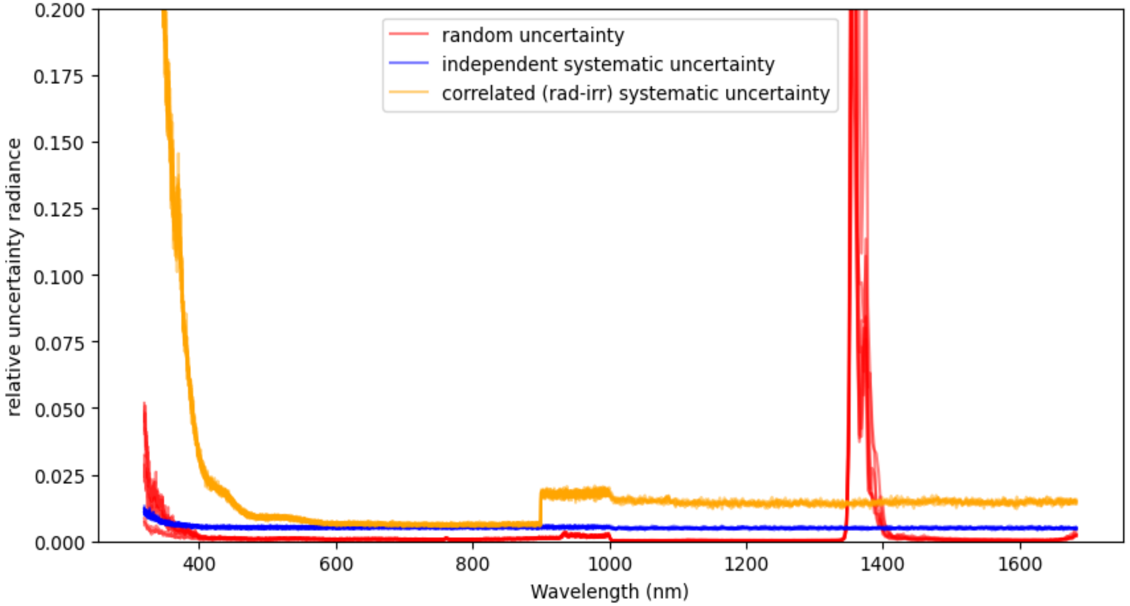
- Validated against NIST uncertainty engine:

  https://colab.research.google.com/github/comet-toolkit/comet_training/blob/main/NIST_example.ipynb

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

- Example from *hypernets_processor*:





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/examples

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

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 a roadmap to building up a comprehensive set of tools
  - e.g. retrieval tool/optimisation, BRDF tool, Look-up tables for faster processing, etc.

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)

Exercises

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