

|  |
| --- |
|  |
| Punpy  Functional Requirements Version 1.0 Draft1 | |
| Pieter De Vis |
| 01/12/20 |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Description** | **Author** |
| 1.0 | 01/12/20 | Draft functional requirements | Pieter De Vis |
|  |  |  |  |

**Version History**

# Introduction

## Purpose

The punpy module is a Python software package to propagate random, structured and systematic uncertainties through a given measurement function.

## Terminology

Within the GUM framework uncertainty analysis begins with understanding the measurement function. The measurement function establishes the mathematical relationship between all known input quantities (e.g. instrument counts) and the measurand itself (e.g. radiance). Generally, this may be written as:

where:

* is the measurand;
* are the input quantities.

Uncertainty analysis is then performed by considering in turn each of these different input quantities to the measurement function. This uncertainty can be done using Monte Carlo (MC) methods (Supplement 1 to GUM) or using the Law of Propagation of Uncertainties (LPU) method (GUM, 2008). There are different types of uncertainties to be propagated:

* **Random effects**: “those causing errors that cannot be corrected for in a single measured value, even in principle, because the effect is stochastic. Random effects for a particular measurement process vary unpredictably from (one set of) measurement(s) to (another set of) measurement(s). These produce random errors which are entirely uncorrelated between measurements (or sets of measurements) and generally are reduced by averaging.”
* **Structured random effects**: “means those that across many observations there is a deterministic pattern of errors whose amplitude is stochastically drawn from an underlying probability distribution; “structured random” therefore implies “unpredictable” and “correlated across measurements”…”
* **Systematic (or common) effects**: “those for a particular measurement process that do not vary (or vary coherently) from (one set of) measurement(s) to (another set of) measurement(s) and therefore produce systematic errors that cannot be reduced by averaging.”

## References

GUM (2008): JCGM 100:2008 Evaluation of measurement data – Guide to the expression of uncertainty in measurement, Report (http://www.bipm.org/en/publications/guides).

GUM Supplement 1 (2008): Supplement 1 to the “Guide to the expression of uncertainty in measurement” - Propagation of distributions using a Monte Carlo method. (https://www.bipm.org/utils/common/documents/jcgm/JCGM\_101\_2008\_E.pdf)

# Requirements

Requirements are graded as follows:

* *Critical:* Core to the software, must be met.
* *Major:* Improves the software, should be met.
* *Minor:* Useful, but not critical or major. If cannot be implemented in a first release perhaps can be implemented later.

## General requirements

### [Critical] Punpy needs to be able to propagate uncertainties through any python function that takes input quantities as function arguments and that returns the measurand.

### [Critical] Punpy should have functions for propagating random uncertainties, systematic uncertainties and structured uncertainties.

### [Critical] Punpy should be able to deal with covariance matrices as well as uncertainties.

### [Mayor] Punpy should allow to propagate uncertainties which are systematic/random along one dimension (associated with repeated measurements) and have a custom correlation structure along another dimension (e.g. between wavelengths).

### [Minor] Punpy should allow to specify a single correlation matrix, that is combined with the uncertainties for repeated measurements

### [Critical] Punpy needs to be able to propagate uncertainties using the MC method.

### [Mayor] Punpy needs to be able to propagate uncertainties using the LPU method.

### [Minor] It should be possible to calculate Look Up Tables (LUT), that might be expensive to calculate the first time. But once they have been calculated, they can be used to significantly speed up the propagation of uncertainties through the same measurement function.

## User interface

Definition of those user interface characteristics that allow to understand and learn the software easily so the user be able to perform his/her tasks efficiently including the interface exemplar description.

### [Critical] It should be easy to install punpy, preferably using pip.

### [Mayor] Punpy is run within python scripts. Examples of these scripts will be provided in the documentation.

### [Minor] User documentation will be provided including general guidelines and examples of use.

## External interface

Definition of interfaces with other software or hardware.

### [Critical] It needs to be very straightforward to import and use punpy in other python codes, as it is to be used as a building block for other codes.

## Input / Output File(s)

The contents of the files that the software will read in/save results to.

### [Critical] Punpy takes as input a measurement function as a python function which takes input quantities as function arguments and that returns the measurand.

### [Critical] Punpy takes as input the input quantities together with associated uncertainties.

### [Mayor] It should be possible to use scalars, 1D arrays, 2D arrays and 3D arrays as input quantities (and associated uncertainties).

### [Critical] Punpy returns uncertainties and correlation matrices as numpy arrays. It does not save this information into files. File saving can always be done outside punpy.

## Mathematical

Equations the software is to apply.

### [Critical] There need to be functions to covert from covariance matrices to correlation matrices, and to convert from correlation matrices and uncertainties to covariance matrices.

### [Critical] For correlating the MC samples, the code needs to be able to calculate the Cholesky decomposition of the correlation/covariance matrix. This is only possible for positive definitive matrices. Some correlation/covariance matrices are only positive semi-definite. It needs to be possible to change these positive semi-definite matrices into positive definite matrices, while changing the present covariance as little as possible.

### [Critical] For the LPU method, punpy needs to be able to efficiently calculate the Jacobian matrix.

## Operational\*

Hardware, operating system, memory requirements, performance, efficiency, portability etc.

### [Critical] It needs to be possible to run punpy using parallel processing on multiple CPUs at the same time to increase efficiency.

### [Mayor] The code should be able to run on linux, mac and windows machines. This includes documenting how to run all options in the code on different machines (this mostly applies to differences in parallel processing).

### [Minor] The MC method can take up loads of memory when processing lots of repeated measurements at the same time. There should be an option that allows to reduce the memory requirements by processing repeated measurements separately.

## Reliability\*

Specification of the software execution level concerning the maturity, fault tolerance and recovery.

### [Critical] Punpy results for the different methods should be tested against analytical calculations.

### [Minor] Punpy should reach at least 80% code coverage.

## Maintenance\*

Description of the elements facilitating the understanding and execution of the future *Software* modifications.

### [Minor] The code needs to be set up so that it is easy to add additional uncertainty propagation methods.

## Design and construction limitations/constraints\*

Needs, timelines imposed by the Customer.

### [Minor] A beta version for the MC method should be ready by April 2021 (in time for hypernets period 2 review).

## Legal and regulative\*

Needs imposed by laws, regulations, NPL security or IP regulations.

### [Critical] QA4EO tools are to be made open-source.