CIMR RGB – Re-Gridding Toolbox

D9 - Software Installation and User Manual

Signature Page

Prepared by	Joseph Heywood, Maksym Brilenkov and Davide Decataldo	Date: 2025-03-19
Issued by	Carolina Barrack - Project Manager	Date: 2025-03-19
Approved by	ESA Technical Officer	Date:

Table of contents

Changelog

Issue	Author	Affected Section	Reason	Status	Date
1.0	S&T AS	All	Document creation	Issued	2024-11-20
1.1	S&T AS	References 4 3.5.1 4.2.4 & 4.2.5 4.2.15 4.2.17 3.5.1 3.5, 4 All sections 3.3 3.1	RID01: MS3-MSC-1 RID03: MS3-MSC-3 RID04: MS3-MSC-4 RID05: MS3-MSC-5 RID06: MS3-MSC-6 RID13: MS3-MSC-13 RID14: MS3-MSC-14, added new parameters RID19: MS3-MSC-19 RID03: MS3-FCA-3 RID07: MS3-FCA-7 RID27: MS3-MSC-27	Modified	2024-12-17
1.2	S&T AS	3.2.1 3.5.1 3.5.5 4.1 4.2	Updating links, changing names of configuration parameters, fixing typos. Added processing_flag variable	Modified	2025-03-19
1.4	S&T AS	3.4	Remove direct link to Google Drive for antenna pattern (downlaod dpr)	Issued	2025-11-25

Acronyms

ATBD	Algorithm Theoretical Baseline Document
AMSR2	Advanced Microwave Scanning Radiometer
BG	Backus-Gilbert
BTs	Brightness Temperatures
CEA	Cylindrical Equal Area
CETB	Calibrated Passive Microwave Daily Equal-Area Scalable Earth Grid 2.0 Brightness Temperature
CG	Conjugate Gradients
CGNE	Conjugate Gradients Normal Equations
CIMR	Copernicus Imaging Microwave Radiometer
DIB	Drop-in-the-Bucket
EASE2	Equal-Area Scalable Earth 2.0
GRASP	General Reflector Antenna Software Package
HDF	Hierarchical Data Format
ICI	Ice Cloud Imager
IDS	Inverse Distance Squared
IODD	Input/Output Data Definition
JAXA	Japanese Space Agency
L1b, L2	Level 1b, Level 2
LW	Landweber
LAEA	Lamberts Equal Area
LTS	Long Term Support
MACRAD	Metrological Analysis of CIMR RADiometry
MRF	Measurement Response Function
MWI	Microwave Imager
NaN	Not a Number
NETCDF	Network Common Data Form
NN	Nearest Neighbour

PAD	Processing and Algorithm Development
RGB	ReGridding toolBox
RMSE	Root Mean Square Error
rSIR	Radiometer Scatterometer Image Reconstruction
SMAP	Soil Moisture Active Passive
SoW	Statement of Work
UML	Unified Modelling Language
XML	eXtensible Markup Language

1. Introduction

1.1. Purpose and Scope

This document contains the *Software Installation and User Manual* for the CIMR RGB project in accordance with the contract and *Statement of Work* (SoW). It is designed to provide comprehensive guidance on the installation, configuration and operation of the software.

It is intended that this guide, along with the *Algorithm Theoretical Baseline Document* [RD8], provides the user with the required knowledge to use the tool and perform a successful regrid, as well as test the various provided configurations.

It is expected that the readers are familiar with the CIMR project and with microwave radiometer data in general. Moreover, this document assumes a user's familiarity with their operating system and basic to intermediate knowledge of Python language.

1.2. Document Structure

The document is structured as follows:

- Section 1 provides an introduction and defines the scope of this document.
- Section 2 provides references.
- Section 3 provides step-by-step instructions to obtain and install the software and perform a regrid.
- Section 4 provides a detailed description of the tool's configuration parameters.

2. References

2.1. Applicable Documents

AD	Title	Reference
[LI]	Request for Proposal for CIMR RGB RE-GRIDDING TOOL BOX - EXPRO - ESA RFP/3-17792/22/I-AG	Invitation letter: Request for Proposal for CIMR RGB RE-GRIDDING TOOL BOX - EXPRO Activity No. 1000035090 in the esa-star system
[SOW]	Statement of Work for “Statement of Work ESA Express Procurement - EXPRO CIMR RGB – Re-Gridding tool Box”	ESA-EOPG-EOPGMQ-SOW-48
[CC]	Draft Contract for “CIMR RGB Re-Gridding Tool Box - EXPRO”	ESA Contract No. 4000xxxxxx/22/I-AG
[TC]	Conditions of Tender for “CIMR RGB Re-Gridding Tool Box - EXPRO”	Appendix 3 EXPRO Tendering Conditions (EXPRO/TC)
[MRD]	Copernicus Imaging Microwave Radiometer (CIMR) Requirements Document Version 4.0	ESA-EOPSM-CIMR-MRD-3236

2.2. Reference Documents

RD	Title	Reference
[RD1]	Soil Moisture Active Passive (SMAP) L1-L3 Ancillary Static Data V001	Peng, J., Mohammed, P., Chaubell, J., Chan, S., Kim, S., Das, N., Dunbar, S., Bindlish, R. & Xu, X. (2019). Soil Moisture Active Passive (SMAP) L1-L3 Ancillary Static Data. (SMAP_L1_L3_ANC_STATIC, Version 1). [Data Set]. Boulder, Colorado USA. NASA National Snow and Ice Data Center Distributed Active Archive Center. https://doi.org/10.5067/HB8BPJ13TDQJ. [describe subset used if applicable]. Date Accessed 11-20-2024.
[RD2]	Generic IPF Interface Specifications	MMFI-GSEG-EOPG-TN-07-0003, v1.8, 2009-08-03
[RD3]	Copernicus Imaging Microwave Radiometer (CIMR) Requirements Document Version 4.0	ESA-EOPSM-CIMR-MRD-3236
[RD4]	CIMR RGB Design Document	CIMR-DD-ST-RGB-001 v1.3
[RD5]	CIMR RGB Level 1c Product Data Format Specifications	CIMR--ICD-ST-RGB-002 v1.3

[RD6]	CIMR RGB Performance Assessment Plan	CIMR-TP-ST-RGB-003 v1.2
[RD7]	CIMR RGB Input/Output Data Definition	CIMR-IODD-ST-RGB-004
[RD8]	CIMR RGB Algorithm Theoretical Baseline Document	CIMR-ATBD-ST-RGB-005
[RD9]	CIMR RGB Detailed Processing Model	CIMR-COM-ST-RGB-006

3. Quick Start Guide

This Section provides all the necessary steps to install, configure and perform a regrid with the RGB software. It begins with setting up a suitable environment and retrieving the software from GitHub. Followed by instructions on running the software, either from the command line or by importing the toolbox as a python library.

3.1. Prerequisites

The software was developed with the intention to be deployed on an `Ubuntu Linux 22.04 Long Term Support` (LTS) (its `amd64` architecture) operating system on a computer with at least 8 CPU cores, 2.6GHz clock frequency, 16GiB memory, and 500GiB storage. The proposed platform requirements can be considered mid-high end and will be achievable by most business/professional hardware. Ubuntu is a popular Linux distribution known for its stability, security, and user-friendliness. The LTS version ensures regular security and maintenance updates for an extended period, making it a good choice for long-term deployment and scalability.

Before installing the software, first create `python` virtual environment with minimum required version (i.e., `>=3.10`) as[1]

```
$ python3.10 -m venv  $PATH/pyenv310-cimr
```

or `conda` virtual environment

```
$ conda create --prefix $PATH/pyenv310-cimr python=3.10
```

where `pyenv310-cimr` is the name of the virtual environment (a different name can be chosen by the user) and `$PATH` can be any path, preferably a folder dedicated to the project.

[**Note**]: The $ in the name indicates that this is a placeholder variable whose actual value should be provided by the user. In the case above $PATH should be substituted for the actual path where the user wants to create a virtual environment.

Once the environment is created, it can be activated with:

```
$ source $PATH/pyenv310-cimr/bin/activate
```

or, in the case of `fish` shell:

```
$ source $HOME/pyenv310-cimr/bin/activate.fish
```

or, in the case of `conda` virtual environment:

[1] The “$” sign indicates the terminal command.

```
$ conda activate $PATH/pyenv310-cimr
```

Finally, verify the python installation with the command

```
$ python3 --version
```

If the `python` installation was successful, then proceed to obtaining and installing the software.

3.2. Obtaining the Software

The software repository is hosted on GitHub. It is organised into multiple branches, each serving a specific purpose. The branches represent different stages of development and stability. In addition, the repository includes **tags** that signify specific versions of the software. These tags generally mark release versions and are the recommended versions for users to download and use.

For users seeking the latest stable version, the **tagged releases** are the preferred option. However, if the user is interested in accessing new development functionality, the `devel` branch is where active development takes place, incorporating the latest features and updates.

When the software in the `devel` branch reaches a stable state and is ready for release, it is merged into the `main` branch. The `main` branch indicates that the software is feature-complete and stable, signalling an imminent release. Shortly after, the version in `main` will be tagged as an official release.

To obtain the software, you can use various methods depending on your needs, as detailed below.

3.2.1. Downloading an Archive of the Repository

A tagged release can be downloaded as a compressed archive from the repository.

- Visit the repository's GitHub page, for which access is required.
- Navigate to the **Releases** section.
- Select the desired version and download the source code archive (`.zip` or `.tar.gz`).

3.2.2. Downloading a Specific Tag (Release Version)

Alternatively, the code can be retrieved from the command line using the `git` command[2]:

```
$ git clone git@github.com:S-T-Norway/CIMR-RGB.git
$ cd CIMR-RGB
$ git checkout tags/<tag_name>
```

Replace `<tag_name>` with the desired release tag (e.g., `v1.0.0`).

---

[2] This command requires the setup of `ssh` access to GitHub. As an alternative, clone the software via `https`, using the command
`$ git clone https://github.com/jmheywood/CIMR-RGB.git`
or use the approach described in Section 3.2.1.

3.2.3. Cloning a Specific Branch

To work with a specific branch, a repository can be cloned by checking out directly to the desired branch. For example, to clone `devel`:

```
$ git clone --branch devel git@github.com:S-T-Norway/CIMR-RGB.git
```

3.3. Installation

Since CIMR RGB is a Python-based software, it is installable and importable as any other Python package. It follows PEP518 and PEP621 and thus uses `pyproject.toml` file to set the dependencies. Section 5.3 of the *Detailed Processing Model* [RD9] lists all packages the RGB relies on and which will be installed together with the toolbox.

Once the code has been obtained from GitHub and the python virtual environment has been set up and activated (see Section 3.1), it can be installed (from within the root folder of the repository) with the command:

```
$ python3 -m pip install .
```

`pip` will install the `cimr-rgb` package together with all its dependencies into the python virtual environment.

Another way to install the package would be in **editable mode**. This mode is primarily used by package developers, contributors, and researchers to enable quick iterations on code without needing to reinstall a package after changes. By using

```
$ pip install -e .
```

the package is linked to its source directory, so updates to the codebase are immediately reflected.

Overall, this is particularly useful for developing Python libraries, contributing to open-source projects, or experimenting with modifications in existing packages. Editable mode works by creating a symbolic link in the Python environment and is ideal for debugging, testing, or collaborative development, though it is not recommended for **production environments**.

[**Note**]: Together with `cimr-rgb` executable, `cimr-rgb` package also instals `cimr-grasp`, which is a stand-alone module created specifically for preprocessing of antenna patterns and whose modus operandi is briefly described in *Input/Output Data Definition* [RD7].

3.4. Downloading Antenna Patterns

Certain regridding algorithms require full information of the antenna patterns of the instrument feedhorns. These files are large and cannot be included directly in the GitHub repository, so they should be downloaded by the user and placed in the correct directory.

The `root` path (labelled `$AP`) for antenna patterns can be specified through the `<antenna_patterns_path>` configuration parameter, while the rest of the folder structures should be

```
$AP/$INSTRUMENT/$BAND
```

where

- `$INSTRUMENT` is the name of the instrument (currently, available instruments are `SMAP`, and `CIMR`)
- `$BAND` is the name of the band (for each instrument, see Section 4.2.4 for the supported bands)

For the available instruments, more information on how to download the antenna patterns is provided below.

3.4.1. SMAP

The antenna pattern for band L can be found at the link [RD1], named `RadiometerAntPattern_170830_v011.h5`. The file is also included in the test dataset folder (`dpr/antenna_patterns/SMAP`), which can be downloaded as described in Section 3.5.5.1.

3.4.2. AMSR2

The antenna patterns are not publicly available, and they were not provided by JAXA at this time.

3.4.3. CIMR

Antenna patterns in GRASP format have been provided by Deimos. A conversion to a standard format (defined in the IODD [RD7]) is required before the antenna patterns can be used by the RGB. The code to perform this conversion is provided as part of this software and described in detail in [RD7]. However, the result of this conversion can be found directly inside the test dataset folder (`dpr/antenna_patterns/CIMR`), which can be downloaded as described in Section 3.5.5.1.

Alternatively, users can run the conversion again to change some of the parameters (such as the resolution or the maximum polar angle of the antenna pattern grid) if they have access to the original GRASP files.

3.5. Running the Software

3.5.1. Using Parameter File

Once the installation is complete, `cimr-rgb` executable will be available in the active virtual environment. The standard way to run the software is by using an XML configuration file. You can execute the software (from anywhere on the system) via the command line with the following syntax:

```
$ cimr-rgb /path/to/config.xml
```

Replace `/path/to/config.xml` with the path to the desired configuration file.

Here is an example of a configuration file:

```
<?xml version='1.0' encoding='utf-8'?>
<config>
  <InputData>
    <type>SMAP</type>
<path>./dpr/L1B/SMAP/SMAP_L1B_TB_47185_D_20231201T212120_R18290_001.h5</pat
h>
    <antenna_patterns_path>./dpr/antenna_patterns</antenna_patterns_path>
    <split_fore_aft>True</split_fore_aft>
    <source_band>L</source_band>
    <target_band>L</target_band>
    <quality_control>True</quality_control>
  </InputData>
  <GridParams>
    <grid_type>L1C</grid_type>
    <grid_definition>EASE2_G9km</grid_definition>
    <projection_definition>G</projection_definition>
    <reduced_grid_inds></reduced_grid_inds>
  </GridParams>
  <ReGridderParams>
    <regridding_algorithm>IDS</regridding_algorithm>
    <search_radius>18</search_radius>
    <max_neighbours>100</max_neighbours>
    <variables_to_regrid>
      bt_h bt_v bt_3 bt_4
      processing_scan_angle
      longitude latitude
      faraday_rot_angle
      nedt_h nedt_v nedt_3 nedt_4
      regridding_n_samples
      regridding_l1b_orphans
      acq_time_utc azimuth
    </variables_to_regrid>
    <source_antenna_method>instrument</source_antenna_method>
    <target_antenna_method>instrument</target_antenna_method>
    <polarisation_method>scalar</polarisation_method>
    <source_antenna_threshold>0.5</source_antenna_threshold>
    <target_antenna_threshold>0.5</target_antenna_threshold>
    <MRF_grid_definition>EASE2_G3km</MRF_grid_definition>
    <MRF_projection_definition>G</MRF_projection_definition>
```

```
    <source_gaussian_params>100000 100000</source_gaussian_params>
    <target_gaussian_params>100000 100000</target_gaussian_params>
    <boresight_shift>True</boresight_shift>
    <rsir_iteration>15</rsir_iteration>
    <bg_smoothing>1</bg_smoothing>
    <antenna_pattern_uncertainty>0<antenna_pattern_uncertainty>
    <cimr_L_nedt>0.3<cimr_l_nedt>
    <cimr_C_nedt>0.2<cimr_c_nedt>
    <cimr_X_nedt>0.3<cimr_x_nedt>
    <cimr_K_nedt>0.7<cimr_k_nedt>
    <cimr_KA_nedt>0.4<cimr_ka_nedt>
    <relative_tolerance>0.001<relative_tolerance>
    <max_iterations>1000<max_iterations>
    <regularisation_parameter>0.0035<regularisation_parameter>
    <max_chunk_size>100<max_chunk_size>
    <chunk_buffer>1.2<chunk_buffer>
  </ReGridderParams>
  <OutputData>
    <save_to_disk>True</save_to_disk>
    <output_path>./output/cimr_rgb</output_path>
    <version>1.0.0</version>
    <creator_name>insert your name</creator_name>
    <creator_email>test@email.com</creator_email>
    <creator_url>insert your url</creator_url>
    <creator_institution>insert your institution</creator_institution>
    <suffix></suffix>
    <timestamp_fmt>%Y-%m-%d_%H-%M-%S</timestamp_fmt>
  </OutputData>
  <LoggingParams>
    <config_path>./src/cimr_rgb/logger_config.json</config_path>
    <decorate>True</decorate>
  </LoggingParams>
</config>
```

[**Note**]: All parameters are discussed in Section 4. In addition, the software can process only 1 L1b file at a time. To process multiple files, users can create a shell script that loops through the files, updating the file path for each iteration.

3.5.2. Via Command Line

The configuration file is an essential component of the software, as it defines the parameters required for proper operation. **The software cannot be run without a valid configuration file**. However, there is an option to override specific (or even all) parameters defined in the configuration file directly via the command line:

```
$ cimr-rgb --param1 value1 --param2 value2 /path/to/config.xml
```

For instance, the values assigned to the parameters <output_path> and <grid_type> can be specified from the command line with the following:

```
$ cimr-rgb --output-path=/home/exampleusername/cim_rgb -gd EASE2_G36km
./configs/rgb_config.xml
```

To see the full list of available options, run:

```
$ cimr-rgb --help
```

3.5.3. Software Outputs and Logging

When the software is run, the generated output will be stored in the directory specified by the `<OutputData/output_path>` variable. This directory serves as the central location for all outputs generated during the execution. The content of the `<OutputData/output_path>` directory includes: a `logs` subdirectory, an output netCDF file containing the data product, and a file with the parameter list used for the last run.

The Python standard library[3] includes an embedded module for handling logging functionality: `dictConfig` from `logging.config` is used to manage logging for critical parts of the RGB system. To configure the logging, two key parameters are introduced in the configuration file: `<LoggingParams/config_path>` and `<LoggingParams/decorate>`.

The `<LoggingParams/config_path>` parameter specifies the path to a file (e.g., `logger_config.json`) that defines the RGB logging configuration. This file contains the necessary setup for loggers, formatters, and handlers, all of which are essential components of the logging process. If `<LoggingParams/config_path>` is left blank, logging is disabled.

During execution, the application will generate logs to monitor progress and capture errors. These log files will be stored in the logs directory, which is specified via the `<OutputData/output_path>` in the parameter file. As an example, here is the part of the log file using a very simple `formatter`:

```
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `DataIngestion`
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `DataIngestion` -- Started Execution
[rgb-logger - INFO]: `DataIngestion` -- Executed in: 0.00s
[rgb-logger - INFO]: `DataIngestion` -- CPU User Time (Change): 0.00s
[rgb-logger - INFO]: `DataIngestion` -- CPU System Time: 0.00s
[rgb-logger - INFO]: `DataIngestion` -- CPU Total Time: 0.00s
[rgb-logger - INFO]: `DataIngestion` -- Process-Specific CPU Usage
(Before): 0.00%
[rgb-logger - INFO]: `DataIngestion` -- Process-Specific CPU Usage (After):
0.00%
[rgb-logger - INFO]: `DataIngestion` -- Memory Usage Change: 0.000000 MB
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `ingest_data`
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `ingest_data` -- Started Execution
[rgb-logger - INFO]: `L` Band: Calculating max altitude for `ap_radius`
[rgb-logger - INFO]: `combine_cimr_feeds`
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `combine_cimr_feeds` -- Started Execution
```

[3] To learn more, please consult the official documentation at:
https://docs.python.org/3.12/library/logging.html

```
[rgb-logger - INFO]: `combine_cimr_feeds` -- Executed in: 0.01s
[rgb-logger - INFO]: `combine_cimr_feeds` -- CPU User Time (Change): 0.01s
[rgb-logger - INFO]: `combine_cimr_feeds` -- CPU System Time: 0.00s
[rgb-logger - INFO]: `combine_cimr_feeds` -- CPU Total Time: 0.01s
[rgb-logger - INFO]: `combine_cimr_feeds` -- Process-Specific CPU Usage
(Before): 0.00%
[rgb-logger - INFO]: `combine_cimr_feeds` -- Process-Specific CPU Usage
(After): 80.30%
[rgb-logger - INFO]: `combine_cimr_feeds` -- Memory Usage Change: 4.218750
MB
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `remove_out_of_bounds`
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `remove_out_of_bounds` -- Started Execution
[rgb-logger - INFO]: `GridGenerator`
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `GridGenerator` -- Started Execution
[rgb-logger - INFO]: `GridGenerator` -- Executed in: 0.00s
[rgb-logger - INFO]: `GridGenerator` -- CPU User Time (Change): 0.00s
[rgb-logger - INFO]: `GridGenerator` -- CPU System Time: 0.00s
[rgb-logger - INFO]: `GridGenerator` -- CPU Total Time: 0.00s
[rgb-logger - INFO]: `GridGenerator` -- Process-Specific CPU Usage
(Before): 0.00%
[rgb-logger - INFO]: `GridGenerator` -- Process-Specific CPU Usage (After):
0.00%
[rgb-logger - INFO]: `GridGenerator` -- Memory Usage Change: 0.000000 MB
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `lonlat_to_xy_cea`
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `lonlat_to_xy_cea` -- Started Execution
[rgb-logger - INFO]: `lonlat_to_xy_cea` -- Executed in: 0.00s
[rgb-logger - INFO]: `lonlat_to_xy_cea` -- CPU User Time (Change): 0.00s
[rgb-logger - INFO]: `lonlat_to_xy_cea` -- CPU System Time: 0.00s
[rgb-logger - INFO]: `lonlat_to_xy_cea` -- CPU Total Time: 0.00s
[rgb-logger - INFO]: `lonlat_to_xy_cea` -- Process-Specific CPU Usage
(Before): 0.00%
[rgb-logger - INFO]: `lonlat_to_xy_cea` -- Process-Specific CPU Usage
(After): 0.00%
[rgb-logger - INFO]: `lonlat_to_xy_cea` -- Memory Usage Change: 0.937500 MB
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `remove_out_of_bounds` -- Executed in: 0.00s
[rgb-logger - INFO]: `remove_out_of_bounds` -- CPU User Time (Change):
0.00s
[rgb-logger - INFO]: `remove_out_of_bounds` -- CPU System Time: 0.00s
[rgb-logger - INFO]: `remove_out_of_bounds` -- CPU Total Time: 0.00s
[rgb-logger - INFO]: `remove_out_of_bounds` -- Process-Specific CPU Usage
(Before): 0.00%
[rgb-logger - INFO]: `remove_out_of_bounds` -- Process-Specific CPU Usage
(After): 0.00%
[rgb-logger - INFO]: `remove_out_of_bounds` -- Memory Usage Change:
0.937500 MB
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `clean_data`
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `clean_data` -- Started Execution
[rgb-logger - INFO]: `clean_data` -- Executed in: 0.00s
[rgb-logger - INFO]: `clean_data` -- CPU User Time (Change): 0.00s
[rgb-logger - INFO]: `clean_data` -- CPU System Time: 0.00s
[rgb-logger - INFO]: `clean_data` -- CPU Total Time: 0.00s
[rgb-logger - INFO]: `clean_data` -- Process-Specific CPU Usage (Before):
```

```
0.00%
[rgb-logger - INFO]: `clean_data` -- Process-Specific CPU Usage (After):
0.00%
[rgb-logger - INFO]: `clean_data` -- Memory Usage Change: 0.312500 MB
[rgb-logger - INFO]: ---------------------
[rgb-logger - INFO]: `ingest_data` -- Executed in: 0.09s
[rgb-logger - INFO]: `ingest_data` -- CPU User Time (Change): 0.08s
[rgb-logger - INFO]: `ingest_data` -- CPU System Time: 0.01s
[rgb-logger - INFO]: `ingest_data` -- CPU Total Time: 0.09s
[rgb-logger - INFO]: `ingest_data` -- Process-Specific CPU Usage (Before):
0.00%
[rgb-logger - INFO]: `ingest_data` -- Process-Specific CPU Usage (After):
97.10%
[rgb-logger - INFO]: `ingest_data` -- Memory Usage Change: 40.179688 MB
```

where `INFO` is the logging level, `rgb` is the name of the custom `logger` object, `bt_h_fore` is the name for the variable to be regridded.

3.5.4. Importing the Software

This section describes how to import the RGB module (or parts of it) into another `python` application (or from an `ipython` notebook or interactive session).

Generally speaking, any python-based library is imported with an `import` statement. The same goes for the RGB library, with the following line:

```
import cimr_rgb.grid_generator as grid_generator
```

“Simple is better than complex” is one of the guidelines from the Zen of Python and the Python code’s design philosophy. This principle is incorporated into the RGB, to make it as simple as possible to perform regridding operations. As an example, to regrid the L1c data, once can simply do:

```
import pathlib

from cimr_rgb.regridder import ReGridder
from cimr_rgb.config_file import ConfigFile
from cimr_rgb.data_ingestion import DataIngestion


configpath = pathlib.Path(".").joinpath('rgb_config.xml')

config = ConfigFile(configpath)

data_dict = DataIngestion(config).ingest_data()
data_dict_out = ReGridder(config).regrid_data(data_dict)
```

Just remember that configuration file should be present and have correct formatting.

3.5.5. Running the Tests

The software is designed with quality and reliability in mind, and for this purpose, the **pytest** framework is used to develop and run tests. The tests ensure the correctness of functionalities, performance, and robustness of the software.

To enable testing, the required dependencies can be installed alongside the software. Use one of the following commands depending on your installation mode:

- **Editable Mode (Development Environment)**:

  ```
  $ pip install -e .[tests]
  ```

- **Production Environment**:

  ```
  $ pip install .[tests]
  ```

3.5.5.1. Test Dataset

A prepared test dataset is available for validating software functionality. The dataset is hosted on Google Drive and can be downloaded via the following link: Download Test Dataset

Details of the Dataset

- Format: `tar.gz`
- Size: ~13.65 GB
- Integrity Check: SHA hash provided for consistency verification.

Before using the dataset, it is strongly recommended to verify its integrity. Follow the steps below to ensure the dataset is not corrupted.

3.5.5.2. Verifying Dataset Consistency

1. **Download the Dataset:**
   Download the dataset from the link provided above.
2. **Obtain the SHA Hash:**
   The SHA hash for the dataset is provided alongside the download link and it is called `dpr_v1.0.3.tar.gz.sha256`.
3. **Verify the Integrity:**
   Use the following commands to compute and verify the SHA-256 hash of the dataset file:

   Compute the SHA-256 hash of the downloaded file

   ```
   $ sha256sum dpr_v1.0.3.tar.gz
   ```

Use the prepared hash file for automated verification

```
$ sha256sum -c dpr_v1.0.3.tar.gz.sha256
```

The output should read like this:

```
dpr_v1.0.3.tar.gz: OK
```

If the output indicates that the hash does not match, re-download the dataset, as it may have been corrupted during transfer.

4. **Extract the Dataset:** Once the hash is verified, extract the dataset into the root directory of the repository. This ensures that the test suite can locate the data as expected. Use the following commands:

   Navigate to the root directory of the repository

   ```
   $ cd /path/to/your/repository/root
   ```

   Extract the dataset into the root directory

   ```
   $ tar -xvzf /path/to/dataset.tar.gz
   ```

3.5.5.3. Running Tests

With the test dataset in place, you can proceed to run the test suite using the `pytest` framework. Below are instructions for running individual tests, multiple tests, or the entire test suite.

To run one test (e.g., T12) do:

```
$ pytest tests/system/test_T.12.py
```

To run several tests, do:

```
$ pytest tests/system/test_T.{12,14,19}.py
```

To run the entire test suite, do the following:

```
$ pytest tests/system/
```

[**Note**]: Once the tests run, the terminal output will contain PASS/FAILED statements, but no terminal logging will be performed. To enable it, add the `-s` flag to the test like so:

```
$ pytest tests/system/test_T.12.py -s
```

4. RGB Configurations

This Section provides a detailed description of how to apply the various configuration parameters in order to perform a successful regridding. Moreover, the configuration decisions/syntax should be the first thing for a user to check when the software crashes or gives errors.

4.1. Configuration Parameters Table

The table below provides a summary of RGB configuration parameters. More detailed information can be found in Section 4.2. For each parameter, it is also specified if the parameter is optional, required, or conditionally required (i.e. required only if a specific algorithm is used). The corresponding XML tag for each parameter must be included in the input file, even when the parameter is not required. If the parameter is not required, the tag may be left empty.

Table 1 - RGB Configuration Parameters.

Parameter	Description	Valid Input	Required /Optional	Command Line Key	Section
<InputData>					
<type>	Specifies the input data type	[SMAP, CIMR, AMSR2]	Required	--input-data-type, -t	4.2.1
<path>	Path to/name of input L1B file	File path string	Required	--input-data-path, -p	4.2.2
<antenna_patterns_path>	Specifies the directory that contains the antenna patterns for a specific instrument.	File path string	Required for BG, RSIR, LW, CG	--antenna-patterns-path, -app	4.2.30
<split_fore_aft>	Separate the measurements into forward and backward looking scans.	[True, False]	Required	--split-fore-aft, -sfa	4.2.3
<source_band>	Specifies the input source band (options depend on <type>)	See conditional options below.	Required	--source-band, -sb	4.2.4
<target_band>	Specifies the target source band (options depend on <type>)	See conditional options below.	Required	--target-band, -tb	4.2.5
<quality_control>	Whether to filter input samples based on associated input quality information.	[True, False]	Required	--quality-control, -qc	4.2.6
<GridParams>					

<grid_type>	Specifies the type of output grid for <target_band> measurements to be remapped onto.	[L1C, L1R]	Required	--grid-type, -gt	4.2.7
<grid_definition>	Specifies the grid parameters for the output grid.	A set of predefined identifying strings. See details below.	Required	--grid-definition, -gd	4.2.8
<projection_definition>	Specifies the projection for the output grid.	A set of predefined identifying strings. See details below.	Required	--projection-definition, -pd	4.2.9
<reduced_grid_inds>	Indices on the output grid for which to perform a regrid.	[row_min, row_max, col_min, col_max] or leave empty to remap the entire input dataset.	Optional	--reduced-grid-inds, -rg	4.2.10
<ReGridderParams>					
<regridding_algorithm>	Specifies the regridding algorithm to apply to the source data.	[NN, DIB, IDS, BG, RSIR]	Required	--regridding-algorithm, -ra	4.2.11
<search_radius>	The radius for which to consider neighbouring samples in <regridding_algorithm>	A float in km OR leave empty to only consider samples within the boundaries of an output grid cell for L1c and 30km for L1r.	Optional	--search-radius, -sr	4.2.12
<max_neighbours>	The maximum number of neighbours to consider in <regridding_algorithm>	An integer OR leave empty to use default value of 200.	Optional	--max-neighbours, -mn	4.2.13
<variables_to_regrid>	Specifies the variables to regrid, to be included in the output product.	See conditional options below, or leave empty to include all default variables.	Optional	--variables-to-regrid, -vtr	4.2.14
<source_antenna_method>	Specifies which form of source antenna pattern to apply to applicable <regridding_algorithm>	[instrument, gaussian_projected, gaussian]	Required for BG, RSIR, LW, CG	--source-antenna-method, -sam	4.2.15
<source_antenna_thr	Specifies the	Float in the range	Required	--source-antenn	4.2.16

eshold>	percentage of maximum power to retain. It sets the range for considered power values as a fraction of the maximum power output.	0 < x ≤ 1	for BG, RSIR, LW, CG	a-threshold, -sat	
<source_gaussian_params>	Specifies parameters for a simulated source antenna pattern. Units dependant on <source_antenna_method>	A list of 2 float numbers (standard deviation in two perpendicular directions) See details below on how these are defined.	Required for BG, RSIR, LW, CG	--source-gaussian-params, -sgp	4.2.17
<target_pattern_method>	Specifies which form of target antenna pattern to apply to applicable <regridding_algorithm>	[instrument, gaussian_projected, gaussian]	Required for BG, RSIR, LW, CG	--target-pattern-method, -tpm	4.2.15
<target_antenna_threshold>	Specifies the percentage of maximum power to retain. It sets the range for considered power values as a fraction of the maximum power output.	Float in the range 0 < x ≤ 1	Required for BG, RSIR, LW, CG	--target-antenna-threshold, -tat	4.2.16
<max_theta_antenna_patterns>	Specifies the maximum polar angle for the antenna patterns, cropping the gain for larger angles.	Angle in degrees	Required for BG, RSIR, LW, CG	--max-theta-antenna-patterns, -mtap	4.2.17
<target_gaussian_params>	Specifies parameters for a simulated source antenna pattern. Units dependant on <target_antenna_method>	A list of 2 float numbers (standard deviation in two perpendicular directions) See details below on how these are defined.	Required for BG, RSIR, LW, CG	--target-gaussian-params, -tgp	4.2.18
<polarisation_method>	Specifies the method by which to apply the antenna polarisation components.	[scalar]	Required for BG, RSIR, LW, CG	--polarisation-method, -pm	4.2.19

<MRF_grid_definition>	Specifies the grid parameters for the image/integration/MRF grid .	A set of predefined identifying strings. See details below. Only EASE grids are valid.	Required for BG, RSIR, LW, CG	--mrf-grid-definition, -mgd	4.2.20
<MRF_projection_definition>	Specifies the projection for the image/integration/MRF grid.	A set of predefined identifying strings. See details below. Only EASE projections are valid.	Required for BG, RSIR, LW, CG	--mrf-projection-definition, -mpd	4.2.21
<boresight_shift>	Whether to shift the SMAP antenna projection to the provided L1b boresight location. (Currently used internally for developmental purposes).	[True, False]	Required for BG, RSIR, LW, CG and SMAP input data	--boresight-shift, -bs	4.2.22
<rsir_iteration>	The number of iterations to apply to the RSIR algorithm.	An integer value n > 0	Required for RSIR	--rsir-iteration, -rsir	4.2.23
<bg_smoothing>	Regularisation factor in the BG algorithm	Float value	Required for BG	--bg-smoothing, -bgs	4.2.24
<regularisation_parameter>	Regularisation parameter for iterative methods (regridding_algorithm = [LW, CG])	Float > 0	Required for LW, CG	--regularisation_parameter, -rp	4.2.31
<max_iterations>	Max iterations for iterative methods (regridding_algorithm = [LW, CG])	Integer > 0	Required for LW, CG	--max-iterations, -mi	4.2.32
<relative_tolerance>	Relative Tolerance for iterative methods (regridding_algorithm = [LW, CG])	Float > 0	Required for LW, CG	--relative-tolerance, -rt	4.2.33
<max_chunk_size>	Chunk size for iterative methods (regridding_algorithm = [LW, CG])	Integer > 0	Required for LW, CG	--chunk-size, -cs	4.2.34

<chunk_buffer>	Chunk buffer for iterative methods (regridding_algorithm = [LW, CG])	Float > 1	Required for LW, CG	--chunk-buffer, -cb	4.2.35
<antenna_pattern_uncertainty>	Antenna pattern uncertainty contribution	Float > 0	Optional	--antenna-pattern-uncertainty, -aps	4.2.36
<cimr_L_nedt>	NEDT for CIMR L-band samples	Float > 0	Optional	--cimr-l-nedt, -nedtl	4.2.37
<cimr_C_nedt>	NEDT for CIMR C-band samples	Float > 0	Optional	--cimr-c-nedt, -nedtc	4.2.38
<cimr_X_nedt>	NEDT for CIMR X-band samples	Float > 0	Optional	--cimr-x-nedt, -nedtx	4.2.39
<cimr_K_nedt>	NEDT for CIMR K-band samples	Float > 0	Optional	--cimr-k-nedt, -nedtk	4.2.40
<cimr_KA_nedt>	NEDT for CIMR KA-band samples	Float > 0	Optional	--cimr-ka-nedt, -nedtka	4.2.41
<OutputData>					
<save_to_disk>	Save the output data locally	[True, False]	Required	--save-to-disk, -std	4.2.25
<output_path>	The output path. Dependant on <save_to_disk>	File path string, or leave empty (see details)	Optional	--output-path, -op	4.2.26
<version>	The version number for the output product.	Version string	Optional	--version, -V	4.2.29
<creator_email>	Email of the user who created the end L1C/L1R product using RGB software. Part of end product metadata.	String	Required	--creator-email, -ce	4.2.42
<creator_url>	URL of the user who created the end L1C/L1R product using RGB software. Part of end product metadata.	String	Optional	--creator-url, -cu	4.2.43
<creator_institution>	Institution of the user who created the end	String	Optional	--creator-institution, -ci	4.2.44

	L1C/L1R product using RGB software. Part of end product metadata.				
<suffix>	The suffix of the end product name.	String	Conditionally required	--suffix, -sf	4.2.45
<timestamp_fmt>	The time stamp format to act as a suffix if <suffix> variable is not used.	String	Required	--timestamp-fmt, -tfmt	4.2.46
<LoggingParams>					
<config_path>	The path to the logging configuration in json format.	File path string, or leave empty.	Conditionally required	--logging-params-config, -cp	4.2.27
<decorate>	Enables custom RGB decorator to track performance (time, CPU and Memory usage) of RGB functionality.	[True, False]	Conditionally required	--logging-params-decorate, -d	4.2.28
<logger_name>	Name of the logger to be used to log in RGBs output. [**Note**]: it should be the same as the one defined inside cimr_rgb_logger_config.json if the user wants to use the one defined in that configuration file.	String	Conditionally required	--logging-params-logger-name, -lm	4.2.47

4.2. Configuration Parameters Details

4.2.1. <type>

This Section is intentionally left blank.

4.2.2. <path>

Currently, the RGB tool processes one L1B input file at a time.

[**Note**]: To process multiple files, users can create a shell script that loops through the files, updating the file path for each iteration.

4.2.3. <split_fore_aft>

When this parameter is enabled, the Earth samples are separated into forward and backward-looking scans in the `data_ingestion` module. Two independent regrids are then performed on the fore and aft datasets independently. AMSR2 has only forward looking scans and therefore this parameter will be automatically set to `False` for this data type.

4.2.4. <source_band>

Any combination of bands can be used together, or use the keyword `All` to regrid data from all bands. This would regrid all source bands to the single output target band.

For L1C, this would regrid the chosen bands to the chosen output grid. Moreover, for L1C, `<source_band>` must be the same as `<target_band>`.

For L1R, a single or multiple source bands can be chosen, but only a single target band can be chosen. In the case multiple source bands are chosen, they will all be regridded to the single target band.

If `<type>` = `AMSR2`:
Valid input: [`6, 7, 10, 18, 23, 36, 89a, 89b, All`]

If `<type>` = `CIMR`
Valid input: [`L, C, X, K, KA, All`]

If `<type>` = `SMAP`
Valid input: [`L`]

4.2.5 <target_band>

Any combination of bands can be used together, or use the keyword `All` for all bands. For L1C, all target bands will be regridded to the chosen grid. In the case of L1R, only a single output grid can be chosen, and all `<source_band>` will be regridded to it.

For L1C, <source_band> must be the same as <target_band>.

If `<grid_type>` = `L1C`:

`<type>` = `AMSR2`:

Valid input: [`6, 7, 10, 18, 23, 36, 89a, 89b, All`]

`<type>` = `CIMR`:

Valid input: [`L, C, X, K, KA, All`]

`<type>` = `SMAP`:

Valid input: `[L]`

If `<grid_type>` = `L1R`:

`<type>` = `AMSR2`:

Valid input: `[6, 7, 10, 18, 23, 36, 89a, 89b]`

`<type>` = `CIMR`:

Valid input: `[L, C, X, K, KA]`

While no L1R regridding is available for SMAP data.

4.2.6. <quality_control>

- When quality control is enabled, it implements the quality control provided with the input L1b data. Only the strictest quality control can be applied, that is, users cannot choose from the range of quality indicators; they are either all applied with `True` or none for `False`.

- This parameter is currently only available for `<type>` = `AMSR2` or `SMAP`.

4.2.7. <grid_type>

This parameter specifies if the RGB should perform an L1C or an L1R regridding from L1b data:

- `L1C`: the variables are regridded into a grid on the Earth's surface, specified by the `grid_definition` and `projection_definition` parameters.

- `L1R`: the variables are regridded to the coordinate position (scan geometry) of a set of measurements obtained from another band, taking into account the corresponding antenna pattern where applicable.

- `L1R2L1C`: It is also possible to perform a double regrid, from L1b->L1r, followed by a regrid from L1r->L1c. **This type of regrid can not be initiated by a command line parameter**. The user will need to perform the first regrid (L1b->L1r) and feed the results back into the ReGridder class to perform the second regrid (L1r->L1c). A tutorial of how to do this is provided in a Jupyter Notebook in the RGB Github repository.

4.2.8. <grid_definition>

Users will be able to select from a selection of pre-defined grids, listed in Table 2.

Table 2 - <grid_definition> valid input and associated grid parameters.

Valid Input	EPSG	X_min [m]	Y_max [m]	Res [m]	N_cols [-]	N_rows [-]
EASE2_G1km	6933	-17367530.44	7314540.83	1000.9	34704	14616
EASE2_G3km	6933	-17367530.44	7314540.83	3002.69	11568	4872
EASE2_G9km	6933	-17367530.44	7314540.83	9008.05	3856	1624
EASE2_N3km	6931	-9000000	9000000	3000	6000	6000
EASE2_N9km	6931	-9000000	9000000	9000	2000	2000
EASE2_S3km	6932	-9000000	9000000	3000	6000	6000
EASE2_S9km	6932	-9000000	9000000	9000	2000	2000
EASE2_G36km	6933	-17367530.44	7314540.83	36032.22	964	406
EASE2_N36km	6931	-9000000	9000000	36000	500	500
EASE2_S36km	6932	-9000000	9000000	36000	500	500
STEREO_N6.25km	3413	-3850000	5850000	6250	1216	1792
STEREO_N12.5km	3413	-3850000	5850000	12500	608	896
STEREO_N25km	3413	-3850000	5850000	25000	304	448
STEREO_S6.25km	3976	-3950000	4350000	6250	1264	1328
STEREO_S12.5km	3976	-3950000	4350000	12500	632	664
STEREO_S25km	3976	-3950000	4350000	25000	316	332
MERC_G25km	3395	-20037508.34	19929239.11	25000	1604	1595
MERC_G12.5km	3395	-20037508.34	19929239.11	12500	3207	3189
MERC_G6.25km	3395	-20037508.34	19929239.11	6250	6413	6378

4.2.9. <projection_definition>

Users will be able to select from a selection of pre-defined projections, listed in Table 3.

Table 3 - <projection_definition> valid input and associated PROJ strings.

Valid Input	PROJ String
G	+proj=cea +lat_ts=30 +lon_0=0 +lat_0=0 +x_0=0 +y_0=0 +datum=WGS84 +ellps=WGS84 +units=m +no_defs +type=crs
N	+proj=laea +lat_0=90 +lon_0=0 +x_0=0 +y_0=0 +datum=WGS84 +ellps=WGS84 +units=m +no_defs +type=crs

S	+proj=laea +lat_0=-90 +lon_0=0 +x_0=0 +y_0=0 +datum=WGS84 +ellps=WGS84 +units=m +no_defs +type=crs
PS_N	+proj=stere +lat_0=90 +lon_0=-45 +lat_ts=70 +k=1 +x_0=0 +y_0=0 +datum=WGS84 +ellps=WGS84 +units=m +no_defs
PS_S	+proj=stere +lat_0=-90 +lat_ts=-70 +lon_0=0 +k=1 +x_0=0 +y_0=0 +datum=WGS84 +ellps=WGS84 +units=m +no_defs
MERC_G	+proj=merc +k=1 +lon_0=0 +x_0=0 +y_0=0 +datum=WGS84 +units=m +no_defs +type=crs

4.2.10. <reduced_grid_inds>

At the current stage of development, the BG and rSIR algorithms have significant processing times on a standard laptop. To address this, users can opt to process only a portion of the output grid, allowing for faster results and more efficient testing.

For L1C, the configuration parameters include `[row_min, row_max, col_min, col_max]`, which specify a selected portion of the output grids as outlined in Table 2. The index `[0,0]` represents the upper-left corner of the grid, while the grid's maximum boundaries are defined by the `N_rows` and `N_cols` columns in Table 2. To use this parameter effectively, users need to know the satellite's path for a given granule, such that they don't attempt to regrid to output cells with no measurements falling within or near them. A helpful tip is to first perform a NN regrid and plot the results on a grid. By visually inspecting the area of interest, users can then determine where to apply the BG/rSIR regrid and rerun the toolbox accordingly on the selected `<reduced_grid_inds>`.

For L1R, the same applies, except that the scan geometry grid of `<target_band>` is indexed. For example, a `<reduced_grid_inds>0 10 0 10<reduced_grid_inds>` will regrid to the footprints of the first 10 samples of the first 10 scans for the `<target_band>`.

4.2.11. <regridding_algorithm>

This Section is intentionally left blank.

4.2.12. <search_radius>

Specifies the radius in km for which to search for neighbouring samples for the target output grid cells.

For L1C, if this parameter is left empty, the search radius will automatically be set to half the target grid cell's resolution, as defined in Table 2. This will create a circle centred on the output grid cell, with the circumference passing through the four corners of the grid cell.

For L1R, if this parameter is left empty, the search radius will be set to 30 km.

4.2.13. <max_neighbours>

If this parameter is left empty, it will default to a value of 200.

4.2.14. <variables_to_regrid>

This parameter defines the variables to be regridded and included in the output product. Variables should be named following the RGB naming convention, as outlined in Table 4. The listed variables represent the standard set for the L1C product (defined in *Level 1c Product Data Format Specifications* [RD5]), but they do not encompass all potential variables. As the project evolves, additional variables may be incorporated. The variables in Table 4 form the core of the standard L1C product at this stage. Notice that not all variables are available for all products.

[**Note**]: The rSIR algorithm is an iterative algorithm that uses antenna patterns, and is therefore only appropriate for the regridding brightness temperatures. However, the user may include additional variables that will be regridded using the IDS algorithm.

Table 4 - Valid <variables_to_regrid> for each data type. An **X** indicates the given variable is available for inclusion. A number refers to an added note below the table.

	Variables		
Valid Input (RGB convention)	**SMAP**	**AMSR2**	**CIMR**
`longitude`	**X**	**X**	**X**
`latitude`	**X**	**X**	**X**
`bt_h`	**X**	**X**	**X**
`bt_v`	**X**	**X**	**X**
`bt_3`	**X**		**X**
`bt_4`	**X**		**X**
`processing_scan_angle`	**X**		**X**
`x_position`	**X**	**X**	**X**
`y_position`	**X**	**X**	**X**
`z_position`	**X**	**X**	**X**
`x_velocity`	**X**	**X**	**X**
`y_velocity`	**X**	**X**	**X**
`z_velocity`	**X**	**X**	**X**
`sub_satellite_lon`	**X**		**X**

sub_satellite_lat	**X**		**X**
faraday_rot_angle	**X**		**1**
geometric_rot_angle			**1**
instrument_status			**2**
nedt_h	**X**		**1, 3**
nedt_v	**X**		**1, 3**
nedt_3	**X**		**1, 3**
nedt_4	**X**		**1, 3**
regridding_n_samples	**X**	**X**	**X**
regridding_quality_measure	**X**		**X**
regridding_l1b_orphans	**X**	**X**	**X**
azimuth	**X**	**X**	**X**
oza			**X**
solar_azimuth		**X**	**1**
solar_zenith			**1**
land_sea_content			**1**
acq_time_utc	**X**	**X**	**X**
calibration_flag			**1**
data_quality_flag	**4**	**4**	**1**
navigation_status_flag			**1**
scan_quality_flag	**4**	**4**	**1**
temperatures_flag			**1**
processing_flag			**X**

1 The simulated data used for the initial development of the RGB **does not** contain this variable. This variable is included in more recent simulated data, which will be integrated into a future version of the RGB.

2 The simulated data used for the initial development of the RGB **does** contain this variable. However, in the absence of an accompanying product specification document for this product, this variable cannot be interpreted.

3 This variable is synthetically created.
4 This variable is not regridded, but it can be applied to the input data with the use of the <quality_control> parameter.

4.2.15. <source_antenna_method>, <target_antenna_method>

Users have 3 options for the application of antenna patterns in the <regridding_algorithm>'s that require them. The valid inputs are [`instrument`, `gaussian_projected`, `gaussian`]. Options were provided here for two main reasons, the first is the potential removal of sensitivities/uncertainties in the assessment of regridding algorithms. The second is to allow for a faster regridding, under the assumption that the application of a gaussian will likely yield reasonable results.

`instrument`: This opens and processes the set of actual antenna patterns accompanying the instrument. The patterns are loaded from file, pre-processed and then projected from the antenna reference frame to the surface of the Earth.

`gaussian_projected`: A gaussian resembling an antenna pattern is created in the antenna reference frame (specifically, a 2D gaussian in the director cosine frame) with user-defined parameters (see `<[source|target]_gaussian_params>` below). The antenna pattern is subsequently projected to the surface of the Earth.

`gaussian`: A 2D gaussian resembling an antenna pattern is created directly on the surface of the Earth, with user-defined parameters (see `<[source|target]_gaussian_params>` below). The particularly computationally expensive operation of projecting the antenna patterns to Earth is, in this case, not performed, rendering this option significantly quicker.

4.2.16. <source_antenna_threshold>, <target_antenna_threshold>

The antenna threshold parameter allows the user to control how much of the pattern is used in the applicable <regridding_algortithm>. In reality, the patterns provide gain/radiance from a large area on Earth, as well as from Space. In practice, it is, for example, common to consider an antenna footprint as the 3-dB (half-gain) beamwidth (which would correspond to a `<source_antenna_threshold>` of 0.5.

This parameter specifies the values of the gain in the antenna pattern to retain relative to the maximum gain. For example, a value of 0.9 would keep all gain values within 10% of the maximum gain.

4.2.17. <max_theta_antenna_patterns>

Value of the polar angle at which the gain of the antenna patterns is cropped. Since the value of the gain drops rapidly at a certain polar angle, this parameter allows to speed up the projection of antenna patterns on the Earth surface, by considering a smaller beamwidth. Since the threshold parameters already restrict the values of the gain effectively used, this parameter is meant as a

further control to limit the size of the antenna patterns and avoid projecting over very large areas on the Earth, where the value of the gain would be anyway negligible. A value of 30-40 degrees is recommended and safe for all the feedhorns.

4.2.18. <source_gaussian_params>, <target_gaussian_params>

If a simulated gaussian is used (i.e., `<[source|target]_antenna_method> = ['gaussian_projected', 'gaussian']`, this parameter defines the shape of the gaussian. A list of two values [`sx`, `sy`] is passed via this parameter, with a meaning which depends on the `<[source|target]_antenna_method>`.

In particular, for `<source[target]_antenna_method> = 'gaussian_projected'`:

- `sx` is the standard deviation in the u direction of the director cosine reference frame of the antenna, with a value in the range $0 < sx \leq 1$
- `sy` is the standard deviation in the v direction of the director cosine reference frame of the antenna, with a value in the range $0 < sy \leq 1$

For `<[source|target]_antenna_method> = 'gaussian'`:

- `sx` is the standard deviation along parallels on the Earth's surface, expressed in metres
- `sy` is the standard deviation along meridians on the Earth's surface, expressed in metres

4.2.19. <polarisation_method>

This parameter defines how to process the various components (cross-pol, co-pol, real, imaginary) in the creation of the projected antenna patterns. At the time of writing, the only valid option is 'scalar'. A second option, 'mueller', which has been partially implemented considers a more elaborate and computationally expensive application of the mueller matrix. The configuration is kept in the case of potential future implementation.

A more detailed description of this parameter can be found in the *Algorithms Theoretical Baseline Document Description* [RD8].

4.2.20. <MRF_grid_defintion>

This parameter defines the *grid* used to construct the image/integration grid upon which the antenna patterns are projected, this is referred to as the *Measurement Response Function* (MRF).

The main function of this for the user is to be able to test and apply different resolutions of MRF's. In the future, it is intended to change this parameter to `<MRF_grid_resolution>`, in which users only have the ability to specify a resolution.

At the time of writing, it is required to define both an MRF grid and projection. This is to allow ease of development and testing. As of now, there are some rules in order to apply this parameter correctly.

- All MRF's are defined on EASE grids, you cannot use stereographic grids for this parameter.

- The grid orientation must be the same as the `<grid_definition>`, that is, if using '`EASE_G9km`', then `<MRF_grid_defintion>` must be '`EASE_G9km`', '`EASE_G3km`', '`EASE_G1km`' etc.

- Similarly if using `<grid_definition>` of '`EASE_N9km`' or '`STEREO_N6.25km`', then `<MRF_grid_defintion>` must be '`EASE_N9km`', '`EASE_N3km`' or '`EASE_N1km`' etc.

4.2.21. <MRF_projection_defintion>

This parameter defines the *projection* used to construct the image/integration grid upon which the antenna patterns are projected, this is referred to as the Measurement Response Function.

It is intended to remove the users ability to choose this parameter in future versions. For now, it is required that this parameter is set to the orientation of `<MRF_grid_defintion>`, that is, if `<MRF_grid_defintion>` is '`EASE_N3km`', this parameters must be '`N`'.

4.2.22. <boresight_shift>

This parameter is only applicable to `<data_type> = 'SMAP'`, otherwise it will be automatically ignored. In order to project the antenna patterns to Earth, the satellite position is required. SMAP L1B data only provides an interpolated value for satellite position once per scan. Consequently, when the antenna pattern is projected to Earth, the boresight of the pattern does not align with the longitude, latitude of the L1B measurement.

Enabling the <boresight_shift> parameter shifts the boresight of the projected pattern, to the boresight provided in the L1B file. It is advised to enable this parameter when using SMAP data.

This parameter might be removed in a future version, if a better strategy for projecting SMAP antenna patterns is developed.

4.2.23. <rsir_iteration>

This parameter defines, k, the number of iterations to apply in the RSIR algorithm. See *Algorithms Theoretical Baseline Document Description* [RD8] for more details.

4.2.24. <bg_smoothing>

This parameter defines, β, the BG smoothing parameter. See *Algorithms Theoretical Baseline Document Description* [RD8] for more details.

4.2.25. <save_to_disk>

This Section is intentionally left blank.

4.2.26. <output_path>

This Section is intentionally left blank.

4.2.27. <config_path>

Depending on RGB's usage, it may be preferable to enable, modify or entirely disable logging functionality. To provide more flexibility, RGB relies on an external file (in `JSON` format, see Annex C of *Input/Output Data Definition* [RD7]) to configure logging, instead of embedding the complete configuration into one of the modules. This variable provides a path to the logging configuration file, and if left blank, no logging will be configured.

The default logging configuration file is provided in the toolbox, named `logger_config.json`.

4.2.28. <decorate>

Enables custom RGB decorator to track performance (time, CPU and Memory usage) of RGB functionality.

4.2.29. <version>

This Section is intentionally left blank.

4.2.30. <antenna_patterns_path>

This Section is intentionally left blank.

4.2.31. <regularisation_parameter>

Regularization parameter λ to be used for iterative methods (`<regridding_algorithm> = [LW, CG]`), as described in the Algorithms Theoretical Baseline Document Description [RD8].

4.2.32. <max_iteration>

Maximum number of iterations for matrix inversion in iterative methods (`<regridding_algorithm> = [LW, CG]`). The algorithm stops at `<max_iteration>` or when the desired relative tolerance (see following parameter) is reached, whichever occurs first.

4.2.33. <relative_tolerance>

Desired tolerance for matrix inversion in iterative methods (`<regridding_algorithm>= [LW, CG]`). When solving iteratively the problem Y = AX, for given A and Y, the relative tolerance at iteration k is computed as

$$\epsilon_k = \frac{\|Y - AX_k\|}{\|Y\|} \tag{4.1}$$

where X_k is the estimate of the solution at iteration k, and $\|\cdot\|$ denotes the Euclidean norm. The algorithm stops when the desired tolerance is reached, or at <max_iteration>, whichever occurs first.

4.2.34. <max_chunk_size>

This parameter is used for iterative methods (<regridding_algorithm> = [LW, CG]). While it is theoretically possible to perform a matrix inversion on the full satellite data granule, given sufficient RAM, this is not feasible on most machines. The <max_chunk_size> parameter enables the user to divide the calculation into smaller chunks, which are processed sequentially.

The <max_chunk_size> defines the maximum size of each chunk to be processed at a time, measured in pixels on the output grid.

For example, with <max_chunk_size> = 100 on an EASE2_9km grid, the regridding algorithm would process a maximum chunk size of 100x100 pixels, corresponding to a surface area of 900 km².

4.2.35. <chunk_buffer>

When performing chunking on matrix inversions, it is important to be careful at the boundaries of chunks to avoid discrete discontinuities at the borders. In order to avoid this, a buffer is created around the chunks to ensure a smooth transition.

A minimum buffer is enforced equal to the radius of an antenna pattern of the band under consideration, this is to ensure that any antenna patterns at the border of the chunk are included in the calculation.

The <chunk_buffer> is defined relative to the size of the antenna pattern radius and it is advised to use a value of at least <chunk_buffer> = 1.2. This parameter must be considered in relation to performance, as a bigger buffer leads to a larger matrix to be inverted and may cause RAM issues.

4.2.36. <antenna_pattern_uncertainty>

This parameter allows the user to conduct a sensitivity analysis on total uncertainty in the absence of detailed information regarding instrument antenna patterns. The uncertainty contribution from the antenna pattern is added to the total uncertainty of the brightness temperatures output from the regridding algorithm, according to the following formula:

$$NEDT_{out} = \sqrt{NEDT_{BT}^2 + NEDT_{AP}^2} \tag{4.2}$$

Where $NEDT_{BT}$ is the uncertainty calculated from the regridding algorithm and $NEDT_{AP}$ is the <antenna_pattern_uncertainty> configuration parameter.

4.2.37. <cimr_L_nedt>

In the absence of a L1b NEDT field for simulated CIMR data, this parameter allows the user to create a field to be input to the regridding algorithm. The uncertainty entered here will be applied to all input samples.

If left empty, the design uncertainty stated in [MRD] is used, equal to 0.3K for L-band.

4.2.38. <cimr_C_nedt>

In the absence of a L1b NEDT field for simulated CIMR data, this parameter allows the user to create a field to be input to the regridding algorithm. The uncertainty entered here will be applied to all input samples.

If left empty, the design uncertainty stated in [MRD] is used, equal to 0.2K for C-band.

4.2.39. <cimr_X_nedt>

In the absence of a L1b NEDT field for simulated CIMR data, this parameter allows the user to create a field to be input to the regridding algorithm. The uncertainty entered here will be applied to all input samples.

If left empty, the design uncertainty stated in [MRD] is used, equal to 0.3K for X-band.

4.2.40. <cimr_K_nedt>

In the absence of a L1b NEDT field for simulated CIMR data, this parameter allows the user to create a field to be input to the regridding algorithm. The uncertainty entered here will be applied to all input samples.

If left empty, the design uncertainty stated in [MRD] is used, equal to 0.7K for K-band.

4.2.41. <cimr_KA_nedt>

In the absence of a L1b NEDT field for simulated CIMR data, this parameter allows the user to create a field to be input to the regridding algorithm. The uncertainty entered here will be applied to all input samples.

If left empty, the design uncertainty stated in [MRD] is used, equal to 0.4K for Ka-band.

4.2.42. <creator_email>

Email address of the product creator. Required parameter.

4.2.43. <creator_url>

URL address of the person / institution responsible for creating the data product. Optional.

4.2.44. <creator_institution>

Name of the institution responsible for creating the data product. Optional.

4.2.45 <suffix>

A string which is going to be added to the end of the product name.

4.2.46. <timestamp_fmt>

A variable representing the timestamp format to act as a suffix for the data product name (if `<suffix>` variable is not used). This variable is also propagated into the end netCDF data product as part of the metadata for fields such as Date Created, Dat Metadata Created, etc. However, the format here will be changed to ISO 8601.

4.2.47. <logger_name>

This section is intentionally left blank.