PNNL-30067

# MHKiT QC Vision

June 2020

Carina Lansing
Chitra Sivaraman
Brian Ermold
Tim Shippert
Eddie Schuman

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

PACIFIC NORTHWEST NATIONAL LABORATORY
*operated by*
BATTELLE
*for the*
UNITED STATES DEPARTMENT OF ENERGY
*under Contract DE-AC05-76RL01830*

PNNL-30067

# MHKiT QC Vision

June 2020

Carina Lansing
Chitra Sivaraman
Brian Ermold
Tim Shippert
Eddie Schuman

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

## Acknowledgments

The material in this report is based upon work conducted in support of the Atmospheric Radiation Measurement User Facility, a U.S. Department of Energy Office of Science user facility managed by the Biological and Environmental Research program.

# Acronyms and Abbreviations

| | |
|---|---|
| ADI | ARM Data Integrator |
| API | application programming interface |
| ARM | Atmospheric Radiation Measurement |
| CSV | comma separated values |
| I/O | input/output |
| MHKiT | Marine and Hydro Kinetic Tools |
| MRE | marine renewable energy |
| QA | quality assurance |
| QC | quality control |
| UTC | Coordinated Universal Time |

## Contents

## Figures

## Tables

# 1.0 Introduction

Marine and Hydro Kinetic Tools (MHKiT) is providing the marine renewable energy (MRE) community open source tools for data processing, visualization, quality control, resource assessment, and device performance. As part of MHKiT's Task 1, the team will contribute modules to perform quality control (QC) checks on data. Robust and automatic QC checks, assessment, and filtering are critical for quickly identifying and removing inaccurate data. In addition, automated QC methods are becoming increasingly necessary as the volume of data being collected by instrumentation networks grows. Providing standard data quality modules in MHKiT will result in:

- A common set of QC tools that can be used consistently across the MRE community
- Support for automated data processing for streaming instrument data
- An increase in data quality and data relevance for design, evaluation, and certification of MRE systems
- An increase in the probability of success and impact of laboratory and field testing
- A reduction in costs and timelines of laboratory and field validation and certification.

This document provides a description of the approach that will be used to provide QC capabilities for MHKiT and a summary of the data quality checks and tools that will be developed.

# 2.0 Background

Several documents have been published regarding the need for data quality assurance (QA) for scientific data [1, 2, 3, 4, 5, 6, 7, 8, 9]. QA refers to preventative measures and activities used to minimize inaccuracies in the data. A key component in data QA is automated QC checks, which are especially important for domains that capture large amounts of streaming sensor data. QC of data streams involves automated or semi-automated processes whereby values and associated timestamps are cross-checked against predetermined standards and optionally separate, concurrently collected data streams. Many of the reports previously referenced detail specific approaches for implementing QC checks in observational data. One program in particular has developed and used a standardized, automated QC process for more than 20 years. The Atmospheric Radiation Measurement (ARM) Climate Research facility has provided the world's atmospheric scientists with continuous observations of cloud and aerosol properties streamed from hundreds of instruments, which are automatically monitored for data QC. The ARM QC system is based upon data quality standards and the ARM Data Integrator (ADI) suite of tools for automatically processing and integrating heterogeneous data streams [4,10,11]. The ARM QC system detects the majority of common time series data quality problems, but is also easily adaptable to individual problems specific to a particular dataset or instrument. In addition, it produces actionable advice embedded in the data that leads to quickly identifying and resolving problems on a per-case basis.

Since the general nature and type of ARM data and QC checks are directly applicable to MRE instrument measurements, the team will leverage the ARM system when designing QC modules for MHKiT. The following sections detail the approach and conventions that will be used and the specific functions that will be provided.

# 3.0 Approach

This section details the conventions, checks, and suite of capabilities that that will be provided by the MHKiT QC module to support a full QA program for the MRE community.

## 3.1 General Assumptions

MHKiT QC will adhere to the following guiding principles, which will be discussed in more detail throughout this section.

1. QC modules will be written for Python 3 and a binding layer will be provided for MatLAB.
2. Data will be represented as an XArray [12] dataset that can support both 1 and N-dimensional data as well as embedded metadata describing QC flags and assessments.
3. MHKiT QC will adhere to specified data standards. Datasets to be used with QC functions must first be converted to standard format prior to invoking any QC.
4. MHKiT QC functions will support the production of data with two different quality levels:
    a. Level 1. Raw data with data standards and QC checks applied. Contains embedded data quality flags that have been applied by automated QC checks.
    b. Level 2. Derived or calculated data that may have been smoothed or gap-filled via aggregation, interpolation, or averaging.
5. MHKiT QC will provide a low-level set of QC test functions that apply to most time series instrument data.
6. MHKiT QC will provide a specification format for declaring a complete suite of QC checks on a data stream.
    a. Users have complete control over which tests are run and how failures are reported.
    b. Users can define custom QC checks and error handlers that can be modularized and reused by others.
    c. Provides complete transparency of the QC process.
    d. Specifications can be checked into a code repository and versioned to support provenance and reproducibility.
    e. Can easily be included as part of a processing pipeline for near-real-time streaming data.
7. MHKiT QC will support reading from and writing to multiple file formats.
8. MHKiT QC will embed the results of QC checks in the data via flags and metadata attributes.
9. MHKiT QC will provide easy-to-use high-level functions for quickly applying and filtering QC on datasets.
10. MHKiT QC will provide not only functions to perform QC checks, but also functions to use, analyze, and visualize QC data.

## 3.2 QC Data Standards

The team highly recommends that the MRE community design a full data standards document such as the one defined by ARM for climate observations [4, 13]. Standards are critical for the development of automated analysis and discovery tools as they enable:

- Consistent analysis across multiple heterogeneous data streams and time periods
- Development of open source libraries that support the creation, access, processing, sharing, and analysis of MRE data
- Code reuse by using consistent formats
- Creation of standardized files that are both human- and machine-readable
- Enhanced data integrity and reliability
- Reduced delays in releasing data products.

These capabilities will only be possible if the data follow a minimum set of standards. While these standards are required for the complete implementation of MHKiT transformation and analysis functions, for the purposes of QC, MHKiT requires only the following subset of conditions. Before calling MHKiT QC functions, users must first standardize raw data per these standards. MHKiT will provide additional data utilities for standardizing field values.

### 3.2.1 Time Variable

Since potential instruments could be deployed anywhere in the world, standardized data streams will ensure that the time variable is stored as Coordinated Universal Time (UTC) and is represented as “seconds since January 1, 1970 00:00:00”, also known as epoch time. For example, an epoch time of 1 means “Thursday January 1, 1970 00:00:01 UTC”; an epoch time of 992794875 is “Sunday June 17, 2001 16:21:15 UTC”.

### 3.2.2 Missing Value

Each standardized data stream must define a specific value to represent the case where no data are available for a given measurement. The selected value should be outside the valid minimum and maximum for that variable. In general, a missing value of -9999.0 is recommended as it will support most measurements. If a different value is used, it must be explicitly stated in attribute metadata for the dataset. For example, to change the missing value of a variable to -100, the attribute metadata would contain the field ***missing_value*** = -***10***.

## 3.3 Data Quality Levels

MHKiT QC functions will support generating QC flags for Level 1 and Level 2 data quality as described in the following sections. The specific QC checks for each data type will be described later in the Section 3.7 on QC Checks.

### 3.3.1 Level 0 (Raw)

Level 0 data are unfiltered, raw data, with no QC tests applied and no data qualifiers (flags) applied. Typically, these are original data streams that are not published but should be preserved. Data quality flags are not assigned.

### 3.3.2 Level 1 (Processed)

Level 1 data are provisional data released in near real time with initial QC testing applied, such as range checking, missing values, and date-time checks. Well-defined data qualifiers are assigned and may be used to guide further review of the data. Level 1 data may also include conversion of property names to published standard names and measurement values to standard units.

### 3.3.3 Level 2 (Derived)

Derived or calculated data that may have been smoothed or gap-filled via aggregation, interpolation, or averaging. Level 2 data should include clear descriptions of the transformation methods applied and flag the quality of the derived data points based upon the quality of input data. For example, one could assign a quality flag if a certain percentage of the data points averaged together are missing.

## 3.4 QC Representation

The MHKiT QC system will allow users to assign mutually exclusive coded values (i.e., flags) to each data point to indicate the potential problems and abnormal states in the data. QC flags represent the results of data quality tests performed on a particular data point, and QC metadata describe each flag, how it was computed, and what the value means. The following sections describe how QC flags and metadata will be represented by MHKiT. Note that MHKit QC will flag potential data inaccuracies but will not alter the data.

### 3.4.1 QC Flags

It is critical to store the QC flags with the data so they are not lost, and as such QC flags for each variable (e.g., temperature) will be embedded in the dataset as a new companion QC variable. A companion QC field will have the same dimensionality and name as the original variable field with the addition of a “qc_” prepended to the field name. For example, the QC flags for a temperature variable would be stored in a new field called ***qc_temperature***.

To store multiple QC flags for a given data point in a single field, QC flags will be stored as a bit-packed, 32-bit integer. As digital information is stored as a series of bits (ones and zeroes), a 32-bit integer for the number 41 would look like this in digital form:

```
0000 0000 0000 0000 0000 0000 0010 1001
```

Bit packing involves using each individual bit comprising the integer value to represent a QC flag, where:

- Each bit represents a single QC test.
- A bit value of 0 means pass, 1 means fail.
- Integer values of 0 mean all QC tests passed for the given field.

So for example the integer above (41) would mean that QC failed tests 1, 4, and 6. A more in-depth discussion of the bit packing technique can be found at: https://engineering.arm.gov/~shippert/ARM_bits.pdf.

### 3.4.2 QC Metadata

Each QC test must be assigned a set of metadata attributes to clarify details about the test:

1. The unique bit associated with the test
2. A unique label for the test (so it can be easily referenced via API methods)
3. A description of the test
4. The variable associated with the test or if it applies to all variables
5. The assessment of the test—whether the flag means the data are “bad” or “indeterminate”:
    a. A “bad” assessment means the value should be rejected.
    b. An “indeterminate” assessment indicates an unusual but not fatal condition that could still be usable.

An example of metadata attributes for three standard QC tests (missing data, or data below or above acceptable range) is shown in Table 1. A discussion of how QC flags and metadata can be easily applied and filtered at the dataset level can be found in Section 3.8.

Table 1. Example of QC Metadata

| Bit | Label | Description | Associated Variables | Assessment |
|---|---|---|---|---|
| 1 | missing | Value is equal to missing_value | All | Bad |
| 2 | below_valid_range | Value is less than the valid_min | All | Bad |
| 3 | above_valid_range | Value is greater than the valid_max | All | Bad |

QC metadata will be stored as variable attributes inside the QC-enhanced XArray Dataset, so each variable will contain a set of metadata defining all the QC checks that were performed on that variable and their disposition. For more information on how QC-enhanced datasets can be saved to file, see Section 3.8.4.

## 3.5 Level 1 QC

Level 1 QC tests are performed against raw data that have been converted to a standard format. The purpose of Level 1 QC is to flag any unusual or erroneous data, but to otherwise leave the data in their original state. Although there are no universal checks that are applicable in all circumstances, some common types of checks apply to most time series instrument data and can be modularized for general use. Table 2 lists the Level 1 QC functions and this section describes the common, low-level QC functions that will be provided by MHKiT to support Level 1 quality checks. These functions can then be combined as part of a full QC specification, as described in Section 3.7.

Table 2. Level 1 QC Functions

| Function | Description |
|---|---|
| check_missing | Check for missing values are present (as specified by the missing_value parameter) |
| check_empty | Check for empty field values - every value should be filled in or have missing_value assigned |
| check_type | Check that the field has the correct data type (e.g., char, string, double, utf timestamp, etc.) (as specified by the data_type parameter) |
| check_below_range | Check if data values are below the valid minimum range (as specified by the valid_min parameter) |
| check_above_range | Check if data values are above the valid maximum range (as specified by the valid_max parameter) |
| check_inf | Check for infinite values in numeric data |
| check_nan | Check for NaNs in numeric data |
| check_sequential | Check if data are sequential, constantly increasing or decreasing, and by an optional fixed interval |
| check_duplicate | Check if the data have repeated constant values |
| check_spike | Check for abrupt changes in the data using the difference between max and min values (delta) within a rolling window |
| check_outlier_std | Check if the value is an outlier based upon the standard deviation from the dataset mean. (Alternate outlier detection algorithms may be added at a future date.) |

### 3.5.1 Correcting Inaccurate Data (i.e., Data Cleaning)

The referenced data quality documents and years of experience with the ARM program have indicated that automated data corrections and gap filling can lead to misinterpretation and inappropriate data use as decisions are subjective and require an in-depth knowledge of the instrument. Therefore, MHKiT Level 1 QC will not provide automated data-cleaning functions. Users, however, may design a Level 2 data stream that makes use of MHKiT transformation functions to interpolate or smooth data, thereby providing a cleaning effect. Level 2 data streams must be designed on a case-by-case basis, taking careful consideration of the instrument, physics, and observations being measured. QC applied to Level 2 data streams is discussed further in the Section 3.6.

## 3.6 Level 2 QC

Higher-order, derived data products may be desired to ease scientists' use of data in MRE research, as they provide an important translation between the instrumental measurements and the geophysical quantities needed for scientific analysis. These Level 2 data products may be derived by combining measurements from heterogeneous data streams, smoothing data points, filling gaps, modifying the coordinate grid, applying algorithms to derive new properties, or any combination thereof. MHKiT will facilitate Level 2 data stream development by providing a transformation API based upon the ADI toolkit [11]. The MHKiT transformation API will be detailed in a separate document, but it will allow users to declaratively specify and automate combining and transforming data streams using algorithms such as averaging, interpolation, and nearest neighbor.

Whenever data are transformed from their original format, the QC flags from the original data may become obscured:

- No one-to-one correspondence – array[n] → array'[m]
- Cannot associate QC for input samples directly with output samples (except for when using nearest neighbor).

In addition, it may be desirable to take QC flags from input samples into consideration when deciding when data points should be used in a transformation. To address these issues, the MHKiT transformation API will:

- Allow users to mask out specified input samples based upon their QC flags, and
- Calculate statistics on the used data points and use those statistics to provide new QC flags for the transformed data, allowing users to define the thresholds used in those statistics.

In addition to providing new QC flags for transformed data, the MHKiT transformation API will also record detailed attributes describing the input variables and transformation parameters used to ensure the process used to generate the new data is well documented. Table 3 provides examples of the computed QC flags that would be recorded for different transformation methods.

Table 3. Level 2 QC Flags

| QC Flag | Description | Transformation Method | Assessment |
|---|---|---|---|
| qc_indeterminate | Some or all of the input values used to create this output value had a QC assessment of indeterminate. | All | Indeterminate |
| qc_all_bad_inputs | All of the input values used to create this output value had a QC assessment of bad. Value is set to missing_value. | All | Bad |
| qc_outside_range | Could not find any good data points within the transformation region. Transformation region is the bin width when averaging or the range parameter used when computing nearest neighbor or interpolating. Value is set to missing_value. | All | Bad |
| qc_interpolate | Interpolation was performed with points other than the two nearest. This could happen if both the nearest points were flagged as bad. | Interpolate | Indeterminate |
| qc_extrapolate | Extrapolation was performed out from two points on the same side of the target index. This occurs if the input grid does not span the output grid or if all points within range on one side of the target were flagged as bad. | Interpolate | Indeterminate |
| qc_not_using_closest | The nearest good point is not the nearest actual point. | Nearest Neighbor | Indeterminate |
| qc_some_bad_inputs | Some, but not all of the inputs in the averaging window were flagged bad and excluded from the transform. | Average | Indeterminate |
| qc_zero_weight | The weights for all of the input points to be averaged were set to zero. | Average | Indeterminate |

Table 3. (contd.)

| QC Flag | Description | Transformation Method | Assessment |
|---|---|---|---|
| qc_bad_std | Standard deviation of all the points to be averaged is greater than the limit set by the std_bad_max transform parameter. | Average | Bad |
| qc_ind_std | Standard deviation of all the points to be averaged is greater than the limit set by the std_ind_max transform parameter. | Average | Indeterminate |
| qc_bad_goodfrac | The fraction of good and indeterminate data points over the averaging interval is less than the limit set by the goodfrac_bad_min transform parameter. | Average | Bad |
| qc_ind_goodfrac | The fraction of good and indeterminate data points over the averaging interval is less than the limit set by the goodfrac_ind_min transform parameter. | Average | Indeterminate |

## 3.7 Specifying a Suite of QC Checks

In addition to the low-level QC functions described previously, to provide support for reproducible processing pipelines, MHKiT QC must allow running a suite of QC checks that can be declaratively configured via a single configuration file. Declarative configuration offers several benefits:

- Improved readability
- Full transparency
- Consistency
- Reduced coding and quicker implementation
- Versioning for configuration definitions
- Full control of the QC process
- Ease of modification and reuse.

With full control of the QC configuration, users can specify:

- A label for the test
- The bit to flag for the test
- A description of the test
- If a test failure should result in the value being replaced with missing_value
- Specific data streams or locations for which the test applies
- Specific variables for which the test applies
- Test-specific parameters
- Variable-specific parameters
- Multiple, aggregate operators for a QC test

- Custom operators
- Multiple, aggregate error handlers for a test
- Custom error handlers
- New tests as composites of other tests.

The QC configuration template can easily be extended to support additional pipeline parameters as more features are added to MHKiT such as the transformation API. In addition, custom QC operators, error handlers, and configuration templates can easily be published as separate modules for easy reuse by the MRE community. Future syntax enhancements could include features such as being able to express simple comparison operators via text expressions, further reducing the amount of code a user was required to write.

Figure 1 shows an example of a declarative QC configuration.

## 3.8 Applying and Utilizing QC Data

To enable scientists in implementing a complete quality assurance program for their data, it is critical that MHKiT QC provide not only functions to perform QC checks, but also easy-to-use, high-level functions to quickly apply QC checks on their entire dataset, as well as functions to quickly filter, visualize, and store QC data. This section highlights additional capabilities that will be provided with the MHKiT QC module to support these features.

### 3.8.1 Data Object Interoperability

For ease of use, the MHKiT QC library will provide utility methods for working with common scientific data objects. In particular, MHKiT will provide methods to convert a Pandas Dataframe to an XArray Dataset and conversely to convert a 1D Xarray Dataset to a Pandas Dataframe to enable users to work with their object of choice when performing analysis operations. MHKiT process-level QC methods will wrap any lower-level functions so users will not need to worry about data representation and conversion between formats.

### 3.8.2 Applying QC

The MHKiT QC library will provide a top-level function for quickly applying QC checks to their entire dataset via the configuration file described in Section 3.7. When applying QC checks to a dataset, users can specify if they would like to replace the corresponding data value with ***missing_value*** in cases of bad and/or indeterminate QC. If the corresponding value is removed due to one of the QC flags, the bit description will clearly state that the value has been replaced by ***missing_value***.

For reproducibility and to support further manual review, it is recommended that the complete set of QC results be stored to a Level 1 data file using input/output (I/O) functions described in Section 3.8.4. However, when analyzing QC-enhanced datasets, users will be allowed to quickly filter values with bad or indeterminate QC, as described in Section 3.8.3.

```yaml
qc_tests:
    time_valid:  # label that can be used to reference check in API methods
      description: "Check that time is valid.  If not, results in critical failure."
      variables: # Variables for which to apply this test
        - time
      assessment: Bad
      operators:  # Can list multiple operators - executed in sequence
        - mhkit.qc.operators.check_missing
        - mhkit.qc.operators.check_type
        - mhkit.qc.operators.check_sequential
      error_handlers:
        - mhkit.qc.handlers.fail  # fail the pipeline
        - mhkit.qc.handlers.email:  # send alert email
            address: pipeline-team@gmail.com  # function-specific parameters

    missing:
        qc_bit: 1
        description: "Value is equal to missing_value"
        assessment: Bad
        variables:
            - ALL  # keyword to apply check to all variables
        operators:
          - mhkit.qc.operators.check_missing
        error_handlers:
          - mhkit.qc.handlers.flag  # flag the failure (default)

variables:  # could be defined in a separate file
  time:
    data_type: long
    units: utf timestamp
    description: "Time offset from midnight"
    missing_value: -9999

  temp_mean:
    data_type: float
    units: degC
    description: "Mean temperature"
    valid_delta: 20
    missing_value: -9999

  vapor_pressure_mean:
    data_type: float
    units: kPa
    description: "Mean vapor pressure"
    valid_min: 0
    valid_max: 10
    valid_delta: 10
    missing_value: -9999
```

Figure 1.  Example QC Configuration

### 3.8.3 Accessing/Filtering QC

To analyze data, it is critical that the MHKiT QC library provide functions for quickly filtering out undesirable values. To provide full flexibility, MHKiT QC will allow users to filter based upon the following criteria:

- Remove all values with bad or indeterminate QC.
- Remove only values with bad QC.
- Remove values with bad and/or indeterminate QC for a specified set of flags.

For manual QC review, the MHKiT QC library will also provide methods to support easy access and use of QC flags from a QC-enhanced dataset. Methods will include:

- Get QC flags by label.
- Set QC flags by label.
- Roll up all QC into a single field.
- Roll up QC into a new field based upon specified flags.

### 3.8.4 QC I/O

The MHKiT QC library will provide extensible reader/writer classes that can be used to read and write a QC-enhanced XArray Dataset to file. Initially the team will provide readers/writers for comma separated values (CSV) as it is a common format that also supports embedded metadata. However, new formats can easily be added and contributed by the community as separate modules. In the case where a file format does not support embedded metadata, the team will provide a default reader/writer that can store variable and QC attributes in a separate, companion file.

### 3.8.5 Visualizing Data Quality Results

The MHKiT QC module will leverage the ARM Data Quality Inspector tool to provide functions for visualizing data quality for a given data stream and variable over a specified time period. For example, the Figure 2 shows the data quality for an irradiance measurement over a 24-hour period. The image indicates that the data quality was good for almost the entire period with only a few bad values due to a failure from QC check #4.

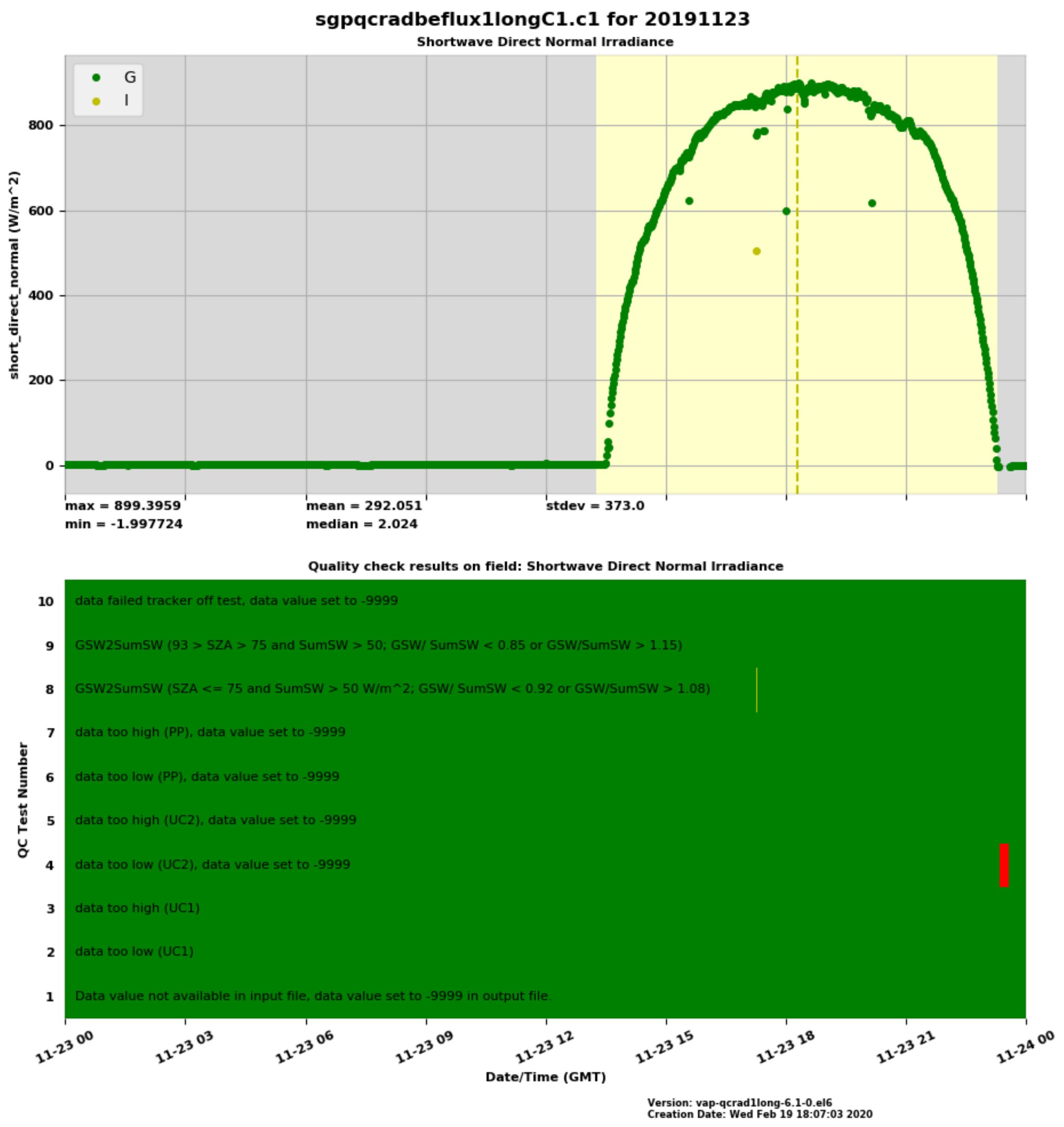

Figure 2. QC Visualization Example #1

Conversely, Figure 3 shows a different example where the data were predominantly bad or indeterminate. The bad data were mainly due to missing data flagged by QC check #4 and the indeterminate data were mainly due to failed QC checks #1 and #3.

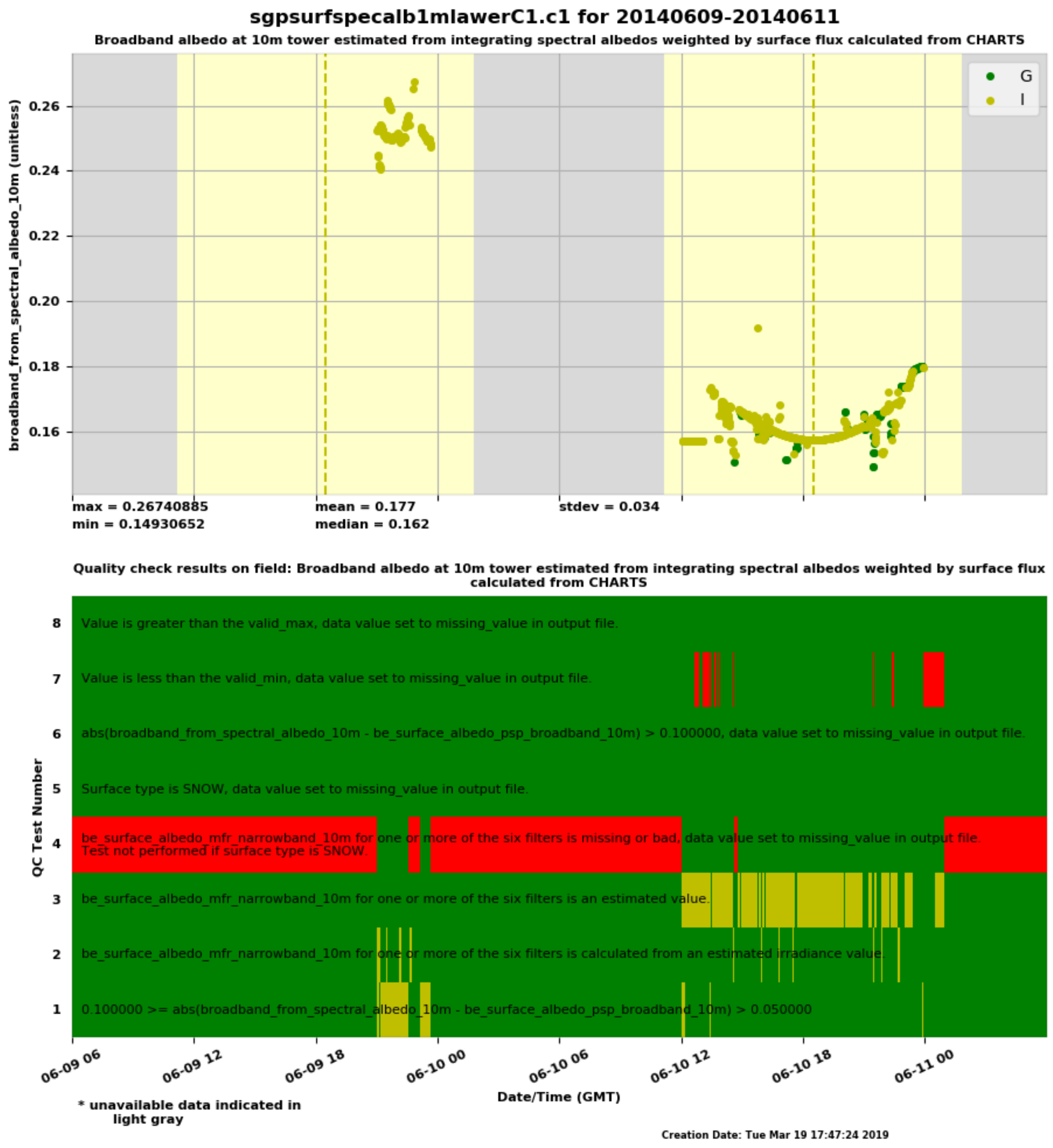

Figure 3. QC Visualization Example #2

## 4.0 References

[1] U.S. Department of Commerce. 2009. *Handbook of Automated Data Quality Control Checks and Procedures.* NDBC Technical Document 09-02, National Data Buoy Center, Mississippi. https://www.ndbc.noaa.gov/NDBCHandbookofAutomatedDataQualityControl2009.pdf.

[2] Integrated Ocean Observing System. 2020. "Quality Assurance/Quality Control of Real Time Oceanographic Data." Accessed June 25, 2020 at https://ioos.noaa.gov/project/qartod/.

[3]  Driscoll F, E Mauer, and J Rieks. 2018. 2017 *Marine Hydrokinetic Instrument Workshop Report*. NREL/TP-5000-70591, National Renewable Energy Laboratory, Golden Colorado. https://www.nrel.gov/docs/fy18osti/70591.pdf.

[4]  ARM Standards Committee. 2016. *ARM Data File Standards: Version 1.2*. DOE/SC-ARM-15-004, U.S. Department of Energy, Office of Science, Washington, D.C. https://www.arm.gov/publications/programdocs/doe-sc-arm-15-004.pdf.

[5]  Kahn BK, DM Strong, and RY Wang. 2002. "Information quality benchmarks: product and service performance." In *Communications of the ACM,* 45(4):184–192.

[6]  Ballou DP and HL Pazer. 1985. "Modeling data and process quality in multi-input, multi-output information systems." *Management Science,* 31(2):150–162.

[7]  Huang K, Y Lee, and R Wang. 1999. *Quality Information and Knowledge*. Prentice Hall, Upper Saddle River, New Jersey.

[8]  Redman, TC, ed. 1996. *Data Quality for the Information Age*. Artech House: Boston, Massachusetts.

[9]  Federation of Earth Science Partners. 2020. "Sensor Data Quality." Accessed June 25, 2020 at http://wiki.esipfed.org/index.php/Sensor_Data_Quality.

[10]  ADI 1.0.0 documentation. 2020. "Overview." Accessed June 25, 2020 at https://engineering.arm.gov/ADI_doc/overview.html.

[11]  ARM-DOE/ADI. 2020. "ADI." Accessed June 25, 2020 at https://github.com/ARM-DOE/ADI.

[12]  xarray. 2020. "Xarray:N-D labeled arrays and datasets in Python." Accessed June 25, 2020 at http://xarray.pydata.org/en/stable/.

[13]  Eaton B, J Gregory, B Drach, K Taylor, S Hankin, J Blower, J Caron, R Signell, P Bentley, G Rappa, H Höck, A Pamment, M Juckes, M Raspaud, R Horne, T Whiteaker, D Blodgett, C Zender, D Lee. 2020. NetCDF Climate and Forecast (CF) Metadata Conventions, Version 1.8. Accessed June 25, 2020 at http://cfconventions.org/Data/cf-conventions/cf-conventions-1.8/cf-conventions.html.