EPJ Web of Conferences **284**, 16006 (2023) https://doi.org/10.1051/epjconf/202328416006
ND2022

FLUKAVAL – A validation framework for the FLUKA radiation transport Monte Carlo code

Markus Widorski[1,*], *Davide* Bozzato[1,2], *Robert* Froeschl[1], and *Vasiliki* Kouskoura[1]

[1]CERN, European Organization for Nuclear Research, 1211 Geneva 23, Switzerland
[2]Karlsruher Institut für Technologie (KIT), Kaiserstraße 12, 76131 Karlsruhe, Germany

Abstract. The FLUKA general purpose radiation transport Monte Carlo code being developed and maintained by CERN (https://fluka.cern) has adopted modern software development standards including a formal quality assurance process. This includes the FLUKAVAL testing framework that takes into account the specific needs of testing a Monte Carlo radiation transport simulation code. FLUKAVAL allows to submit, process and validate a large number of test cases for a new FLUKA version and compare the results against previous versions and reference data. It produces a quantitative and qualitative comparison and compiles a summary report over all selected test cases, which is reviewed before the new FLUKA version is released. The overall validation requires only few manual steps and makes use of large computing clusters to run simulations in parallel.

1 Introduction

The FLUKA [1, 2] general purpose radiation transport Monte Carlo code being developed and maintained by CERN (https://fluka.cern) has adopted modern software development standards including a formal quality assurance process. This includes the FLUKAVAL testing framework that takes into account the specific needs of testing a Monte Carlo radiation transport simulation code.

FLUKAVAL is a software application that enables semi-automatic batch submission, processing, validation and reporting of test cases. Any FLUKA input and any dedicated tests having access to the FLUKA code at the single interaction level can be adapted and integrated in a few steps into the FLUKAVAL framework. FLUKAVAL makes extensive use of the git version control system to store simulation and reference data sets in a git repository located on the CERN GitLab instance.

The results are processed, validated and compiled in a comprehensive evaluated report, which allows the direct comparison of results from the FLUKA version under validation with results obtained with previous FLUKA versions, with other radiation transport codes and/or experimental data. A quantitative analysis is performed in addition to a qualitative comparison whenever possible.

2 Objectives

The primary objective for the FLUKA validation framework is to confirm that each released FLUKA version delivers reliable and consistent results compared to previous versions. Any changes in the results shall be explained by improvements done to the code. Considering the number of parameters it is obvious that only a fraction of the features can be probed. FLUKAVAL uses test cases which are each designed to cover particular functionalities or processes of the FLUKA code. Scalability is a key element allowing to increase the number of test cases over time.

Beyond the version intercomparison, the validation framework simplifies the comparison of FLUKA results with experimental data or data produced with other Monte Carlo simulation codes.

A future objective, which will be implemented for the next FLUKAVAL release, shall be the possibility of an automated comparison of results from the same FLUKA version with varying input parameters for sensitivity analysis.

3 Code structure

FLUKAVAL is entirely written in Python, using a number of external and standard library modules: GitPython for all git manipulations, the ubiquitous numpy, scipy and matplotlib libraries, as well as jinja2 for the Latex report generation. Two internal projects are included as submodules for the processing of FLUKA output data and cluster job submission. FLUKAVAL is a command-line application, offering commands to initialise the FLUKAVAL instance, submit jobs, perform job status reporting, post-process output data, validate the results and generate a summary report. It is semi-automatic, allowing the user to control each step from submission up to the report generation.

4 Database

The git version control system plays a major role in FLUKAVAL. While it is used for the code development management and versioning, it is moreover used to manage the data results from test case simulation runs and

*e-mail: markus.widorski@cern.ch

FLUKAVAL validation steps. In combination with a GitLab instance it serves as a database with built-in flexibility, good performance and scalability as well as an interface for data retrieval. Different FLUKA versions are distinguished by git branches and the processing steps are tagged for easy checkout at the required processing step for the validation of the results and the generation of reports.

5 Test case preparation

Scalability of FLUKAVAL was a key requirement in its design. The framework allows to include a large number of test cases for batch processing from submission to report generation. The actual limitation on the number of test cases has not yet been identified.

A test case is defined by one or more FLUKA input files, auxiliary files and user routines required to run the simulation. An additional processing routine must be included that aggregates and translates the FLUKA output data into a standardised formatted JSON file. Pre-built functions simplify the construction of this case-specific routine for the test case designer. The same standardised output JSON data format was adopted for reasons of compatibility as defined and used by the GEANT-VAL project (https://geant-val.cern.ch/).

Test cases are self-contained in a stand-alone git repository, which are included in FLUKAVAL as submodules and grouped into test batches for easier batch processing. For particular releases or depending on features to be probed, subsets of test cases can be grouped and executed. Hundreds of test cases can be submitted and processed with a single command. Currently 30 model level test cases and 21 full simulation test cases are implemented and available.

6 FLUKAVAL operation steps

The use of FLUKAVAL is split in these sequential steps:

1. Submission of simulation jobs
2. Job status check
3. Post-processing of output data
4. Validation: Qualitative and quantitative comparison of different FLUKA outputs and reference data
5. Report generation

Each step is applied manually and sequentially on a batch of test cases.

6.1 Submission and status check

FLUKAVAL provides an interface to submit test cases, grouped into batches, to a local machine or to a computing cluster, such as LXBATCH hosted at CERN and based on HTCondor (https://htcondor.org/). In principle any computing cluster can be interfaced. It will split and distribute the requested test cases on the available computing nodes to obtain sufficient statistics and to share the computing resources equally. While jobs are running, it is possible to produce summary statistics on the logging and error outputs to identify potential crashes or failures, which require further investigation. This has turned out to be very helpful to test new release candidates and identify and correct bugs before release.

6.2 Processing

After successful termination of a simulation run, a processing step is required to aggregate and translate the output data into the standardised format. In the validation step only one-dimensional data sets are compared that have to be prepared in histogram, scatter or categorical representation. The JSON data files include metadata encoding essential information on the test case and the input parameters. These files are used in the next step to compare outputs from the same input files but run with different versions of FLUKA. Reference data or data from other Monte Carlo simulations can be prepared in the same format to be available in the validation step.

6.3 Validation

Data sets from different sources, e.g. different FLUKA versions, other codes or reference data, are compared on a point-by-point basis. The ordinate value of each data point at the identical abscissa is compared to the data point from the version under test by calculating the ratio, including the corresponding uncertainty. The ratio is compared to the validation criteria specified for each observable:

- **Coverage factor** (significance sigma): The factor to calculate the expanded uncertainty from the combined standard uncertainty to determine the confidence interval for a data point.
- **Acceptance range**: The maximum accepted deviation from 1 for a ratio value, expressed as an acceptance range, e.g. [0.95 , 1.05], which usually contains the value 1.
- **Relative uncertainty threshold**: If the relative expanded uncertainty of the ratio value is larger than this threshold, a warning is raised (high uncertainty). This is only done for values which fall within the specified acceptance range.

A statistic is provided for each observable, specifying the part of the data points being classified as:

- Compatible with the acceptance range considering the given coverage factor and if the one-sided relative uncertainty is below or equal the specified threshold. (**Accepted**)
- Compatible with the acceptance range considering the given coverage factor and if the one-sided relative uncertainty is above the specified threshold. (**Accepted with high uncertainty**)
- Not compatible with the given acceptance range considering the specified coverage factor. (**Rejected**)

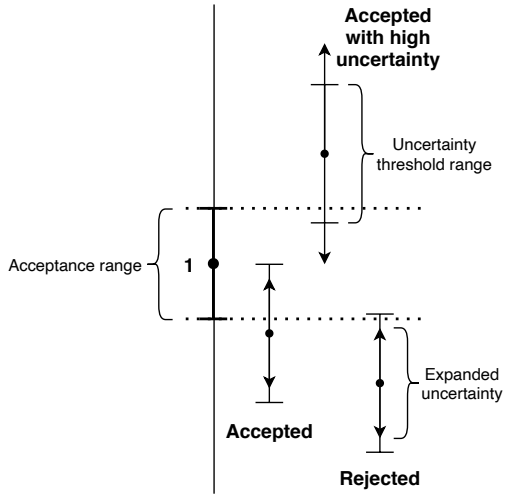

Figure 1. Classification criteria for non-zero data and/or reference points considering a specified acceptance range, the expanded uncertainty and the uncertainty threshold.

Figure 1 illustrates the relation between the ratio value, its expanded uncertainty, the acceptance range and the uncertainty threshold. Figure 2 shows the results of the numerical comparison for each point on the two bottom plots for an example case.

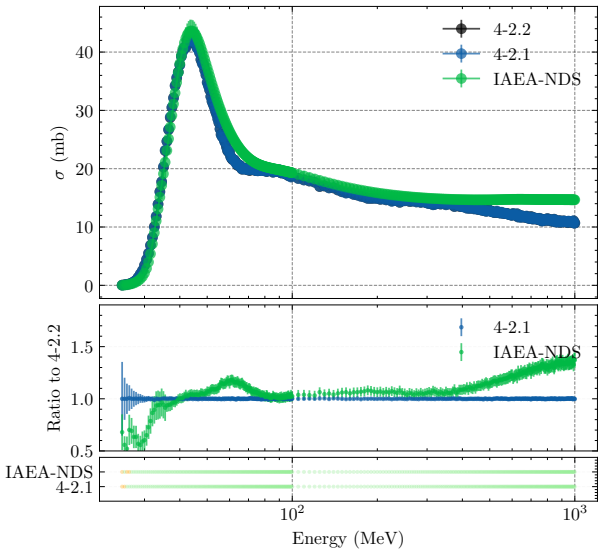

Figure 2. Comparison of a data set for the reaction ^{27}Al(p,x)^{22}Na between two FLUKA versions and reference data [3]. The top graph shows the values from all data sets. The plot below shows the ratio for each data set with data from the version under test and its expanded uncertainty. The bottom graph provides a color coded overview about the classification of the ratio values.

After the validation step, the comparison for specific versions and reference data has been completed and the generated plots and numerical comparison data are stored in the test case git repository.

6.4 Reporting

Any test case with a completed validation step can be included in an automatically compiled report. The report contains summary tables on the numerical comparison results, and the comparison plots. It further contains a detailed description of each included test case. Another information added is a normalised CPU performance metric to visualise changes that may be observed between versions in the required computing time for each test case. The LaTeX document with its figures can then be further edited to include comments on specific observations from the numerical and graphical inter-comparison.

After completing the manual editions, the completed report undergoes an internal review. The validation runs are performed by members of the FLUKA.CERN collaboration, while the review is done by the core developers. The FLUKA.CERN collaboration management releases the final report. The report is part of the quality assurance program set up by the FLUKA.CERN collaboration.

7 Conclusion

The FLUKA Validation framework has proven its value and capabilities for the validation of new releases of FLUKA through a large number of test cases that are executed and processed, with the results compared to previous versions. FLUKAVAL is part of the quality assurance program for the FLUKA code.

Its modular and scalable architecture allows one to add more test cases in the future to probe further features of FLUKA on a regular basis. Test cases which are shared in the MC community, like contained in the NEA SINBAD database, have been already and shall be further included in the FLUKAVAL test case portfolio.

FLUKAVAL is actively developed by the FLUKA.CERN collaboration to include the possibility to perform sensitivity analysis on various input parameters in the future. Further computing clusters and MC codes may be interfaced in the future.

References

[1] C. Ahdida et al., Frontiers in Physics **9**, article 788253 (2022).
[2] G. Battistoni et al., Annals of Nuclear Energy 82, 10-18 (2015).
[3] A. Hermanne et al., Nucl. Data Sheets 148 (2018) 338-382.