Geant-val:
a web application for validation of detector simulations

Luc Freyermuth[1], *Dmitri* Konstantinov[2,3,*], *Grigorii* Latyshev[2,**],
Ivan Razumov[2,3], *Witold* Pokorski[3]and *Alberto* Ribon[3]

[1]EISTI (Cergy, France)
[2]NRC "Kurchatov Institute" – IHEP (Protvino, Russia)
[3]CERN

Abstract. One of the key factors for the successful development of Monte-Carlo programs for physics simulations is to properly organize regression testing and validation. Geant4, the world-standard toolkit for HEP detector simulation, heavily relies on this activity. The CERN SFT group, which contributes to the development, testing, deployment and support of the toolkit, is also in charge of running on a monthly basis a set of community-developed tests using the development releases of Geant4. We present the web application Geant-val developed for visualizing the results of these tests so that comparisons between different Geant4 releases can be made. The application is written using Express.js, Node.js and Angular frameworks, and uses PostgreSQL for storing test results. Test results are visualised using ROOT and JSROOT. In addition to pure visual comparisons, we perform different statistical tests (χ^2, Kolmogorov-Smirnov, etc) on the client side using JavaScript Web Workers.

1 Introduction

Geant4 [1] is a toolkit for the simulation of the passage of particles through matter. Its areas of application include high energy, nuclear and accelerator physics, as well as studies in medical and space science. The Geant4 team produces a public release once a year, with monthly internal testing releases. These results need to be validated.

Physics validation of Geant4, which includes regression testing, comparison of performance between different physics models and comparison with corresponding experimental measurements, requires analysing a large number of histograms produced by Geant4 tests.

A tool for release validation was developed to perform statistical and visual comparisons with respect to previous release(s) and to the experimental data.

2 Web application

The developed web application consists of three main parts: a backend (server side), a frontend (client side) and a framework for configuring and running Geant4 tests. The components of Geant-val web application and interactions between them are shown in the Fig. 1.

*e-mail: Dmitri.Konstantinov@cern.ch
**e-mail: Grigorii.Latyshev@cern.ch

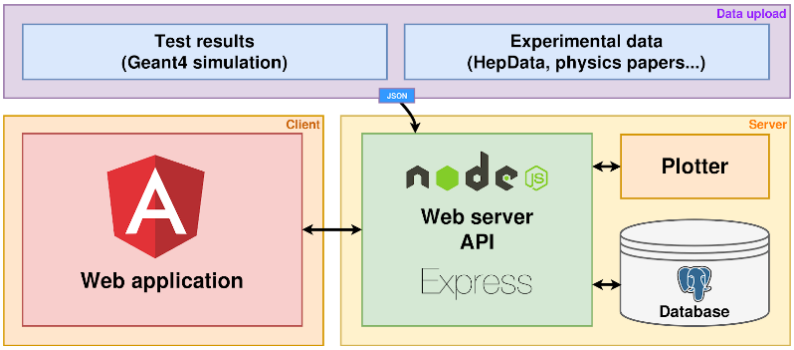


Figure 1. Components of Geant-val web application and flow of data between them.

The server is the core of the Geant4 validation system. It provides a web API that allows clients to access the database, respond to the clients' requests and generate high quality plots "on the fly" whenever they are requested using the ROOT [2] data analysis framework.

It is written in JavaScript and runs with the Node.js engine [3].

The database is used for storing plots containing simulation results or experimental data, together with parameters describing these plots: tool name and version (e.g. `Geant4, 10.4.p02`), name of the test by which the plot was produced, or Inspire (HepData) ID of the article, etc. PostgreSQL [4] is used as database management system for the application. The database instance is provided by the CERN Database-On-Demand service

A ROOT-based C++ `plotter` was developed to produce high quality plots. It uses data in the JSON format (see Appendix 1) which has been introduced as main interchange format between all parts of the application. The utility is not deeply integrated in the server infrastructure and can be used as a standalone application. It supports all types of application's data, can plot histograms with different binning on one canvas, and produce ratio plots. Ranges and scales of plot axes are selected automatically, with ability to override if necessary.

The last component of the validation application is the client frontend, which is an AngularJS [5] single page application (SPA). AngularJS is chosen because it allows to build efficient web applications with highly reusable components. The SPA contains 4 pages:

- User layouts
- Statistical comparisons
- Experimental data
- Look-up table

User layouts page (see Fig. 2) is used to perform fast visual validation of Geant4. One can use one of the existing templates or create and upload their own one. If two or more Geant4 versions are selected, one can also select a "reference" dataset and plot the ratio of other datasets to it.

Statistical comparisons page (see Fig. 3) allows one to perform comparison of simulation with compatible experimental results using statistical tests. It displays results of statistical comparison for pairs of plots with the same parameters' values. Currently χ^2 ($\chi^2/n.d.f.$, χ^2 probability) and Kolmogorov-Smirnov (KS Max(D), KS probability) tests are implemented. All computations are performed asynchronously on the client side using JavaScript WebWorkers. For this purpose, JavaScript code to perform χ^2 and Kolmogorov-Smirnov

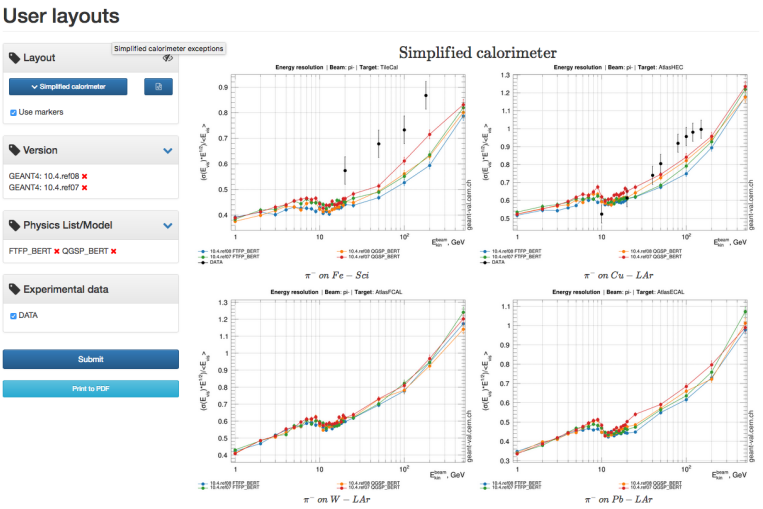


Figure 2. Example of user layout for the Geant4 "simplified calorimeter" test showing test results for two Geant4 reference releases - 10.4.ref07 and 10.4.ref08.

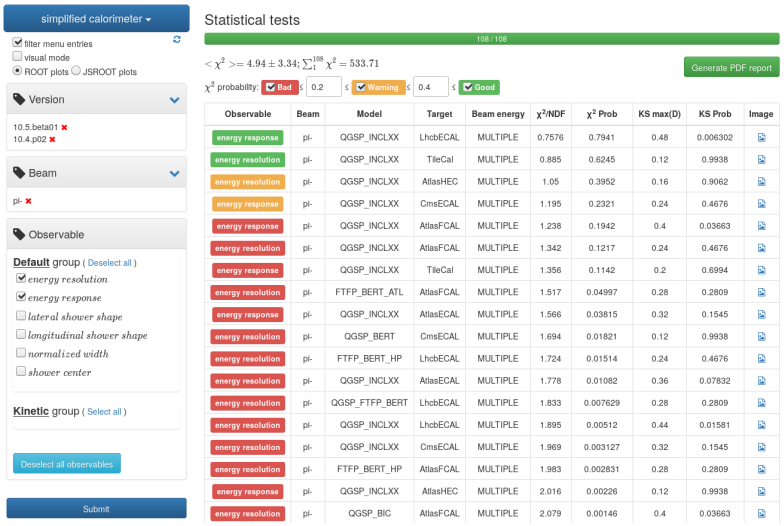
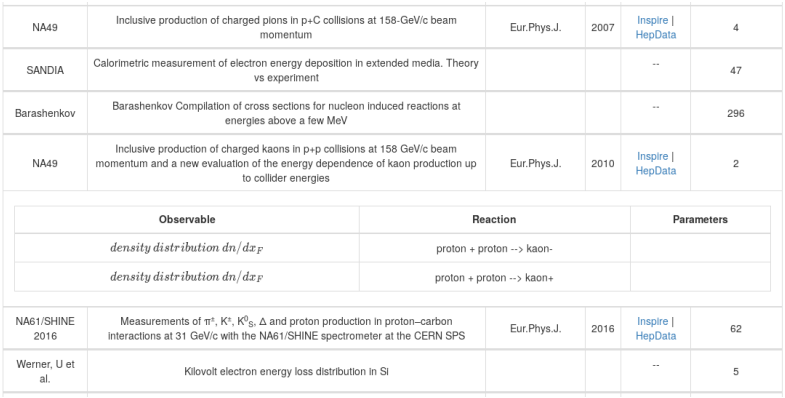


Figure 3. Example of statistical comparison between two official Geant4 releases, 10.5.beta01 and 10.4.p02, for the "simplified calorimeter" test.

tests have been written, and their results cross-checked against the same statistical techniques implemented in the ROOT framework.

On the *experimental data* page (see Fig. 4) a summary table of available experimental data is displayed. The data is extracted from original articles or imported from HepDATA [6] portal.

Each plot on the website can be displayed as a static image produced by `plotter` or as an interactive JSROOT [7] object, which allows changing axes ranges, scales and styles of



NA49	Inclusive production of charged pions in p+C collisions at 158-GeV/c beam momentum	Eur.Phys.J.	2007	Inspire | HepData	4
SANDIA	Calorimetric measurement of electron energy deposition in extended media. Theory vs experiment			--	47
Barashenkov	Barashenkov Compilation of cross sections for nucleon induced reactions at energies above a few MeV			--	296
NA49	Inclusive production of charged kaons in p+p collisions at 158 GeV/c beam momentum and a new evaluation of the energy dependence of kaon production up to collider energies	Eur.Phys.J.	2010	Inspire | HepData	2

Observable	Reaction	Parameters
$density\ distribution\ dn/dx_F$	proton + proton --> kaon-	
$density\ distribution\ dn/dx_F$	proton + proton --> kaon+	

NA61/SHINE 2016	Measurements of $\pi^{\pm}$, $K^{\pm}$, K^0_S, Λ and proton production in proton–carbon interactions at 31 GeV/c with the NA61/SHINE spectrometer at the CERN SPS	Eur.Phys.J.	2016	Inspire | HepData	62
Werner, U et al.	Kilovolt electron energy loss distribution in Si			--	5

Figure 4. Available experimental data.

the data "on the fly" in the same way as it can be done in ROOT. It is possible to export the plots in PNG, ROOT, EPS, Gnuplot [8] or Geant-val JSON format.

Access to the test results produced with internal Geant4 releases is restricted to the Geant4 developers authenticated via CERN Single Sign-On. For public Geant4 releases the access is open to anyone.

3 Validation workflow

To perform the validation of Geant4 using Geant-val one needs to run all the tests, convert their output into the JSON format and upload the produced files to the website.

3.1 Building and running a test

To manage a set of Geant4 tests and their configurations, a `mc-config-generator` Python framework was developed. It allows one to configure and run test jobs in various batch systems (CERN LSF, HTCondor, Torque PBS), and to convert the results into JSON format for further uploading to the application database. The framework is not Geant4-specific, and can be used with other projects (e.g., Pythia8). Source code is available in corresponding Git repository[1].

Each test supported by the `mc-config-generator` consists of a "parameters" file, containing all combinations of parameters used by the given test, and one or more "template" files.

3.2 Producing JSON files

We use JSON, like shown in Appendix 1, for importing data into the database and for exchanging information between different parts of the web application. Each JSON file contains one plot along with metadata describing that plot.

For the supported tests, the `mc-config-generator` framework can be used to convert text and/or ROOT files produced by the test into JSON format. The conversion is done in multiple threads to minimise the execution time.

[1] https://gitlab.cern.ch/GeantValidation/geant-config-generator

3.3 Uploading JSON files

Geant4 developers can upload JSON files by sending a HTTP POST request to Geant-val website using `geant_upload.py`[2] command-line utility.

After uploading the results, these are available on the website for further analysis.

4 Conclusion

The presented software provides an uniform and clear way of performing all stages of physics validation: from creating configuration files and running simulation code to producing final plots.

The Geant-val web application and the `mc-config-generator` framework are actively used by Geant4 developers in CERN and by Geant4 Medical Simulation Benchmark Group (G4BSMG).

References

[1] S. Agostinelli *et al.* [GEANT4 Collaboration], Nucl. Instrum. Meth. A **506**, 250 (2003). doi:10.1016/S0168-9002(03)01368-8

[2] Rene Brun and Fons Rademakers, ROOT - An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. See also (http://root.cern.ch).

[3] NodeJS is a JavaScript runtime https://nodejs.org/

[4] PostgreSQL is an open-source database https://www.postgresql.org/

[5] AngularJS is an MWM JavaScript framework https://angularjs.org/

[6] E. Maguire, L. Heinrich and G. Watt, J. Phys. Conf. Ser. **898** (2017) no.10, 102006 doi:10.1088/1742-6596/898/10/102006 [arXiv:1704.05473 [hep-ex]].

[7] Bertrand Bellenot and Sergey Linev 2015, J. Phys.: Conf. Ser.,664, 062033

[8] Gnuplot is a portable command-line driven graphing utility http://www.gnuplot.info/

[2] https://gitlab.cern.ch/GeantValidation/GVP/raw/master/scripts/geant_upload.py

5 Appendix

```
{
  "article": {"inspireId": -1},
  "mctool": {"name": "Geant4", "version": "", "model": ""},
  "testName": "",
  "metadata": {"observableName": "", "reaction": "",
    "targetName": "", "beamParticle": "",
    "beamEnergies": [], "secondaryParticle": "",
    "parameters": [
      {
        "names": "THETA",
        "values": "60 degrees"
      }
    ]
  },
  // for scatter data
  "plotType": "SCATTER2D",
  "chart": {
    "nPoints: 0,
    "title": "", "xAxisName": "", "yAxisName": "",
    "xValues": [], "yValues": [],
    "xStatErrorsPlus": [], "xStatErrorsMinus": [],
    "yStatErrorsPlus": [], "yStatErrorsMinus": [],
    "xSysErrorsPlus": [], "xSysErrorsMinus": [],
    "ySysErrorsPlus": [], "ySysErrorsMinus": []
  },
  // for histogram data
  "plotType": "TH1",
  "histogram": {
    "nBins: [0],
    "title: "", "xAxisName": "", "yAxisName": "",
    "binEdgeLow": [], "binEdgeHigh": [], "binContent": [],
    "yStatErrorsPlus": [], "yStatErrorsMinus": [],
    "ySysErrorsPlus": [], "ySysErrorsMinus": [],
    "binLabel": []
  }
}
```

Appendix 1. Example of JSON format used by the web application.