HepMC3 Event Record Library for Monte Carlo Event Generators

Andrii Verbytskyi[1], Andy Buckley, David Grellscheid, Dima Konstantinov, James William Monk, Leif Lönnblad, Tomasz Przedzinski and Witold Pokorski

[1]Max-Planck-Institut für Physik, Föringer Ring 5, München 80805, DE

E-mail: [1]andrii.verbytskyi@mpp.mpg.de

Abstract. We present the HepMC3 library designed to perform manipulations with event records of High Energy Physics Monte Carlo Event Generators (MCEGs). The library is a natural successor of HepMC and HepMC2 libraries used in the present and in the past. HepMC3 supports all functionality of previous versions and significantly extends them. In comparison to the previous versions, the default event record has been simplified, while an option to add arbitrary information to the event record has been implemented. Particles and vertices are stored separately in an ordered graph structure, reflecting the evolution of a physics event and enabling usage of sophisticated algorithms for event record analysis. The I/O functionality of the library has been extended to support common input and output formats of HEP MCEGs, including formats used in Fortran HEP MCEGs, formats used in HepMC2 library and ROOT. The functionality of the library allows the user to implement a customised input or output format. The library is already supported by popular modern MCEGs (e.g. Sherpa and Pythia8) and can replace the older HepMC versions in many others.

1. Introduction

The HepMC3 [1] [2] library is designed to perform manipulations with event records of High Energy Physics Monte Carlo Event Generators (MCEGs). The library is a natural successor of the HepMC and HepMC2 [3] libraries used in the present and in the past. HepMC3 supports all the functionality of previous versions and significantly extends it.

2. Data and object model

During the simulation of elementary particle reactions at high energies in MCEGs it is necessary to store and/or modify the information related to the simulation. HepMC3 aims to aid MCEGs with this task. The logical structure of the information in the HepMC3 library follows the general convention of MCEGs. The smallest unit of information is an event which contains particles and vertices. The particles and vertices are stored separately in an ordered graph structure, reflecting the evolution of a simulated physics event and enabling usage of sophisticated algorithms for event record analysis. In HepMC3 the information is represented via C++ objects and can be serialised as C++ structures with plain data types. The main types of objects (plain structures) in HepMC3 are:

- *GenRunInfo (GenRunInfoData)* – type of the main bookkeeping object that holds metainformation about the generated event(s): list of used tools, names of used event weights.
- *GenEvent (GenEventData)* – type of the main data object that holds the position of the primary interaction, list of vertices, particles and attributes in the event.
- *GenVertex (GenVertexData)* – type of the objects used to describe decays and interactions, holds its position, list of incoming and outgoing particles, can have multiple attributes stored in the parent *GenEvent*.
- *GenParticle (GenParticleData)* – type of objects used to describe particles, holds momenta, flavour, status of the particle, can have multiple attributes stored in the parent *GenEvent*.
- *Attribute (std::string)* – base class used to store arbitrary information. The attribute data is stored as a string, which is used to initialise an object of arbitrary type derived from the *Attribute* class.

The *Attribute* objects allow custom information to be stored in the events. Apart from the attributes used to store plain types (double, int, std::string) the library provides implementation for the following attributes:

- *GenPDFInfo* – used to store PDF information, holds same information as the HepMC2 counterpart.
- *GenCrossSection* – used to store cross-section information, holds same information as the HepMC2 counterpart.
- *GenHeavyIon* – used to store heavy information for heavy ion collisions. To improve the treatment of heavy ion collisions the *GenHeavyIon* object has been extended significantly. The final goal of these changes is to satisfy the requirements (e.g. to comply to Lisbon Accord [4]) of groups performing heavy ion physics studies.

Thanks to the usage of ISO/IEC 14882:2011 C++ (denoted below as c++11), the object model of the library has been significantly improved and simplified in comparison to the model used in HepMC2. All custom iterators were removed and the library became more modular, allowing the implementation of custom features without breaking the compatibility with core library components.

Another important novelty in the HepMC3 library with respect to the predecessor is the built in support of LHEF routines and file format [5] [6]. The Les Houches accord on an event file format (LHEF) is used for passing events from a matrix element generator program (MEG) to a MCEG implementing parton showers, underlying event models, hadronisation models etc. The standard implementation in C++ of the LHEF routines had already been maintained by Leif Lönnblad. It was decided to merge that implementation into the HepMC3 library and provide in this way a single package for manipulations with event records used in MCEGs and MEGs.

3. I/O capabilities

The serialisation of the MCEG event record is the most important part of the library. Historically the serialisation was implemented in different packages and in different formats. The number of formats led to problems in the interaction between different simulation packages. For instance, significant technical difficulties arise when the LHC-era MCEGs are used in the simulation and reconstruction chains of older experiments. To overcome such difficulties support for the following formats was implemented in the library:

- IO_GenEvent – the plain text based format used in HepMC2 [3] library, see Listing 1.

```
HepMC::Version 3.0.0
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 0 9.188128e+01 1.298440e-01 7.818181e-03 221 0 7 10001 10004 0 1 1.0000000000000000e+00
N 1 "0"
U GEV MM
C 2.6442255100000002e+03 2.6442255100000002e+03
F 11 -11 9.97420767e-01 9.99999975e-01 9.18812775e+01 1.56824725e+01 2.82148362e+06 0 0
V -1 0 0 0 0 0 1 2 0
P 10001 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4 0 0 -1 0
P 10002 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5881355265109356e+01 4.5881355265109356e+01 0.0000000000000000e+00 61 0 0 -3 0
P 10003 22 0.0000000000000000e+00 0.0000000000000000e+00 1.1864473489064407e-01 1.1864473489064407e-01 0.0000000000000000e+00 1 0 0 0 0
V -2 0 0 0 0 0 1 2 0
P 10004 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4 0 0 -2 0
P 10005 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999998855671230e+01 4.5999998855671230e+01 0.0000000000000000e+00 61 0 0 -4 0
P 10006 22 0.0000000000000000e+00 0.0000000000000000e+00 -1.1443287704082650e-06 1.1443287704082650e-06 0.0000000000000000e+00 1 0 0 0 0
V -3 0 0 0 0 0 0 1 0
```

Listing 1. IO_GenEvent (HepMC2 standard) as implemented in HepMC3

This implementation is fully compatible with one in the HepMC2 library.

- Asciiv3 – native plain text based format, see Listing 2.

```
HepMC::Version 3.0.0
HepMC::Asciiv3-START_EVENT_LISTING
W 0
E 0 7 12
U GEV MM
W 1.0000000000000000000000e+00
A 0 GenCrossSection 2.64422551e+03 2.64422551e+03 -1 -1
A 0 GenPdfInfo 11 -11 9.97420767e-01 9.99999975e-01 9.18812775e+01 1.56824725e+01 2.82148362e+06 0 0
A 0 alphaQCD 0.129844
A 0 alphaQED 0.007818181
A 0 event_scale 91.88128
A 0 mpi 0
A 0 signal_process_id 221
A 0 signal_process_vertex 0
P 1 0 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4
P 2 1 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5881355265109356e+01 4.5881355265109356e+01 0.0000000000000000e+00 61
P 3 1 22 0.0000000000000000e+00 0.0000000000000000e+00 1.1864473489064407e-01 1.1864473489064407e-01 0.0000000000000000e+00 1
P 4 0 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4
P 5 4 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999998855671230e+01 4.5999998855671230e+01 0.0000000000000000e+00 61
P 6 4 22 0.0000000000000000e+00 0.0000000000000000e+00 -1.1443287704082650e-06 1.1443287704082650e-06 0.0000000000000000e+00 1
P 7 2 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5881355265109356e+01 4.5881355265109356e+01 0.0000000000000000e+00 21
```

Listing 2. HepMC3 Asciiv3 (HepMC3 standard)

While being similar to IO_GenEvent, this format is extendable and in comparison to the former requires less storage space, as it does not save meaningless information on particles (e.g. colour flow for hadrons).

- HEPEVT – plain text based format used by many MCEGs written in Fortran (e.g. Pythia6), see Listing 3.

```
E 0 12
  4 -11 0 0 3 11 0.00000000E+00 0.00000000E+00 -4.60000000E+01 4.60000000E+01 5.11000000E-04
       0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
  4 11 0 0 4 12 0.00000000E+00 0.00000000E+00 4.60000000E+01 4.60000000E+01 5.11000000E-04
       0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 61 -11 1 1 5 5 0.00000000E+00 0.00000000E+00 -4.59999989E+01 4.59999989E+01 0.00000000E+00
       0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 61 11 2 2 6 6 0.00000000E+00 0.00000000E+00 4.58813553E+01 4.58813553E+01 0.00000000E+00
       0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 21 -11 3 3 7 7 0.00000000E+00 0.00000000E+00 -4.59999989E+01 4.59999989E+01 0.00000000E+00
```

Listing 3. HepEVT (Fortran era generators) as implemented in HepMC3

The main purpose of the implementation is to provide a compatibility layer for the MCEGs used in the completed HEP experiments at HERA, LEP and PETRA machines.

- ROOTTTree – ROOT [7] based format with GenEvent objects serialized as ROOT TTree, see Fig. 1. This is the format of biggest practical interest. It has several advantages in comparison to the other formats: it allows random access, access over network, has the best I/O performance and requires the smallest amount of storage space per event.
- ROOT – ROOT [7] based format with GenEvent objects written to ROOT file “as is”.
- LHEF – the plain text based LHEF, see Listing 4.

```
  <event>
  6      66 0.50109093E+02 0.88344569E+02 0.75563862E-02 0.12114027E+00
         2 -1    0    0  502    0 0.00000000E+00 0.00000000E+00 0.62728157E+03 0.62728166E+03 0.33000000E+00 0.0000E+00 0.0000E+00
         5 -1    0    0  501    0 0.00000000E+00 0.00000000E+00 -.44481443E+02 0.44739678E+02 0.48000000E+01 0.0000E+00 0.0000E+00
         6  2    1    2  501    0 -.59350665E+02 0.72244533E+02 0.21037840E+03 0.28804239E+03 0.17311146E+03 0.0000E+00 0.0000E+00
        24  1    3    3    0    0 0.28747403E+02 0.66692858E+02 0.11544955E+03 0.15832494E+03 0.80398000E+02 0.0000E+00 0.0000E+00
         5  1    3    3  501    0 -.88098069E+02 0.55516749E+01 0.94928843E+02 0.12971745E+03 0.48000000E+01 0.0000E+00 0.0000E+00
         1  1    1    2  502    0 0.59350665E+02 -.72244533E+02 0.37242173E+03 0.38397894E+03 0.33000000E+00 0.0000E+00 0.0000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.86321681E+00 0.11022720E+01 0.00000000E+00 0.00000000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.91292678E+00 0.10493312E+01 0.00000000E+00 0.00000000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.91292678E+00 0.10493312E+01 0.00000000E+00 0.00000000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.91292678E+00 0.10493312E+01 0.00000000E+00 0.00000000E+00
  <rwgt>
   <wgt id="1001"> 0.50109E+02 </wgt>
   <wgt id="1002"> 0.43255E+02 </wgt>
   <wgt id="1003"> 0.55234E+02 </wgt>
```

Listing 4. First lines of some LHEF event

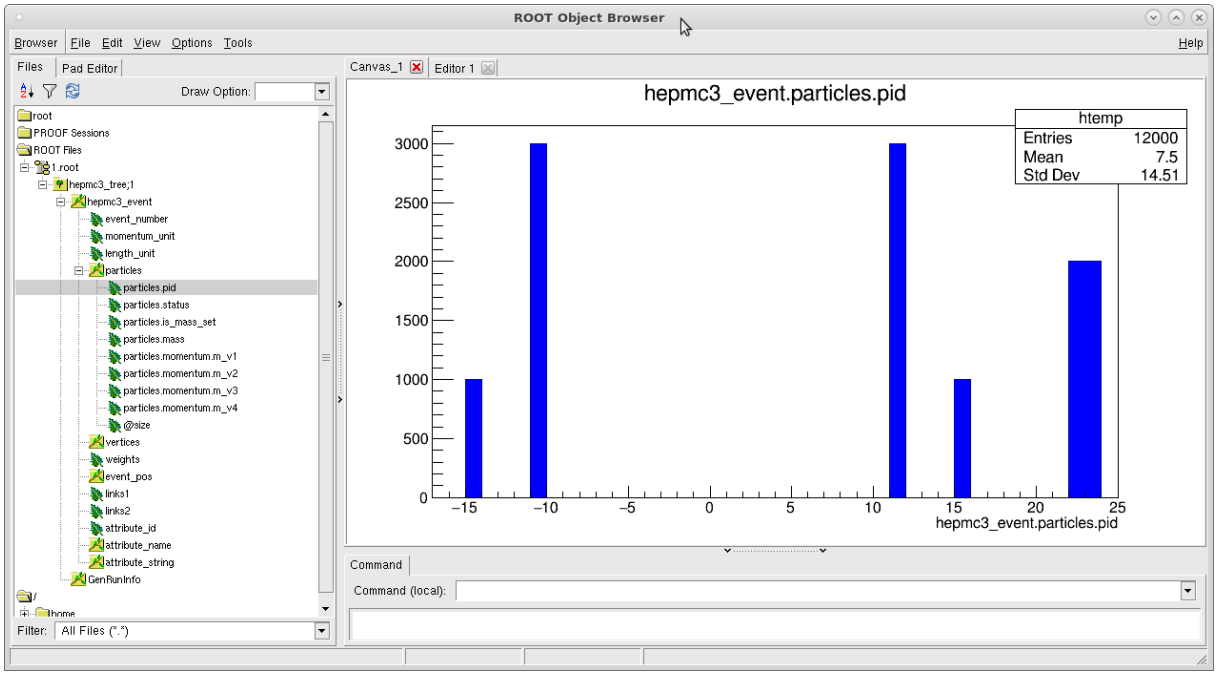

Figure 1. Inspection of $e^-e^+ \to \tau^-\tau^+$ events simulated by the Pythia8 [8] Monte Carlo event generator. The file with HepMC3 events (ROOTTTree) is explored with the ROOT Object browser without loading of the HepMC3 library.

In addition to that the functionality of the library allows user to implement customised input or output format via implementation of custom *Reader* and/or *Writer* classes inherited from the base classes *Reader* and *Writer*. The formats described above were introduced by different groups of people for different purposes. Therefore the amount of information they hold is significantly different. The HEPEVT is the most restrictive format and holds only the information on the particles without any options for extra information. The IO_GenEvent record has fixed format, i.e. the information is limited to particles, vertices, weights, PDF and Heavy ion information and no extension is allowed. The ROOTTTree, ROOT, LHEF and Asciiv3, in addition to the standard, can hold almost arbitrary information via the attributes mechanism. The attributes were used to reach compatibility with the HepMC2 software in the I/O ReaderAsciiHepMC2 and WriterAsciiHepMC2, e.g. the attributes *alphaQCD* and *alphaEM* emulate the corresponding class members of GenEvent class in the HepMC2 library. With this emulation the events can be read from IO_GenEvent files produced by the HepMC2 library without any loss of information.

4. Installation, dependencies and usage

The installation instructions for the package are provided in the README file distributed with the library source codes and are the same for all the supported platforms.

The only basic dependency for the installation of the library is the availability of a c++11 compatible C++ compiler and the build tool CMake3 [9]. The ROOT6 package is needed for support of I/O in ROOT-based formats. The Doxygen [10] and LaTeX packages are needed for the compilation of the documentation. The examples shipped with the library require a FORTRAN77 compatible compiler, and installation of the ROOT6, Pythia8, TAUOLA, PHOTOS, HepMC2 and graphviz [11] programs. The library contains a build-in test suite based on CTest [9].

Operating system	Repository	HepMC3 versions	Contributors
MacOS	homebrew-hep [12]	3.1	Enrico Bothmann
ArchLinux	Standard(AUR) [13]	3.1/3.0	Frank Siegert
Debian9, (+Ubuntu18...)	Standard(NEW) [14]	3.1	Mo Zhou
Fedora28,29,30,31 (+CentOS7,RHEL7...)	Standard(EPEL) [15]	3.1	Mattias Ellert
openSUSE42.3, (+SUSE...)	Some unofficial	3.0	
Windows10	N/A	3.0/3.1	

Table 1. Summary on systems where HepMC3 was tested and the availability of HepMC3 precompiled binaries. Only the Intel-compatible 64-bit architecture (x86_64) was considered.

It has been tested extensively on Ubuntu, CentOS, Fedora, openSUSE, Windows10 and macOSX operating systems. Binary packages are available for multiple operating systems, see Tab. 1 for details. Starting from version 3.1.0, the HepMC3 and HepMC2 libraries can co-exist in one installation, therefore the migration of user code from HepMC2 to HepMC3 can go as easy as possible. As of April 2019 several MCEGs were interfaced to HepMC3, see Tab. 2 for details.

Code	Type	HepMC3 version	Code version, implementation
SHERPA-MC [16]	MCEG	3.1/3.0	master/2.2.5
JetScape [17]	MCEG	3.0	1.0
ThePEG2 [18]	MCEG Toolkit	3.1	master
Herwig7 [19]	MCEG	3.1	via ThePEG
Pythia8 [8]	MCEG	3.1/3.0	in HepMC3/in HepMC3
Pythia6 [20]	MCEG	3.1	in HepMC3
Tauola [21]	MCEG	3.1/3.0	in HepMC3/in HepMC3
Photos [22]	MCEG	3.1/3.0	in HepMC3/in HepMC3
WHIZARD [23]	MCEG	3.0?	
Rapgap [24]	MCEG	3.1	in progress(>3.303)
EvtGen [25]	MCEG	?	in progress(>1.7.0)
GeantV [26]	Simulation	3.0	master
pyhepmc-ng/scikit-hep [27]	Utility	3.1/3.0	master/master
MC-TESTER [28]	Analysis/Ploting	3.1/3.0	in HepMC3/in HepMC3
Rivet [29]	Analysis/Ploting	?	in progress(2.8.0)

Table 2. Summary on the usage of HepMC3 in external projects. "master" stands for the latest version in the used version control system of the official repository, e.g. for master branch of git repository. If known, the versions where support is expected to be released are given in brackets.

5. Interfaces, examples and documentation

The presented library contains some interfaces to the MCEGs, which do not ship the interfaces to HepMC3, see Tab. 2. These interfaces can be used instantly in the production or tests to generate the Monte Carlo simulated events. The library provides some example programs for such simulations, e.g. with Pythia8 and Pythia6 MCEGs. The other sophisticated example programs in the library are the utility to convert the event records between different formats ConvertExample (see Listing 5)

```
[username@hostname1 outputs]$ bin/convert_example.exe -h
convert_example 3.1
Convert between different file formats of Monte Carlo event record.
Example:
         convert_example -i hepmc2 -o treeroot input.hepmc output.root

Usage: convert_example [OPTIONS]... [inputfile outputfile]...

  -h, --help                    Print help and exit
  -i, --input-format=STRING     Input format  (possible values="hepmc2",
                                  "hepmc3", "hpe", "root", "treeroot",
                                  "lhef") (mandatory)
  -o, --output-format=STRING    Output format  (possible values="hepmc2",
                                  "hepmc3", "hpe", "root", "treeroot",
                                  "treerootopal", "hpezeus", "dump", "dot")
                                  (mandatory)
  -e, --extensions=STRING       Extensions, in a form extension=value, could be
                                  passed to readers/writers
      --events-limit=LONG       Limit of events to read from input
                                  (default='100000000')
      --first-event-number=LONG Lowest allowed event number
                                  (default='-100000000')
      --last-event-number=LONG  Highest allowed event number
                                  (default='100000000')
      --print-every-events-parsed=LONG
                                Frequency of parsing information printouts
                                  (default='100')

[username@hostname1 outputs]$
```

Listing 5. Help instructions of the conversion utility.

and a graphical viewer of event structure ViewerExample (see Fig. 2).

The online documentation is available on the HepMC3 home page [2]. It includes the

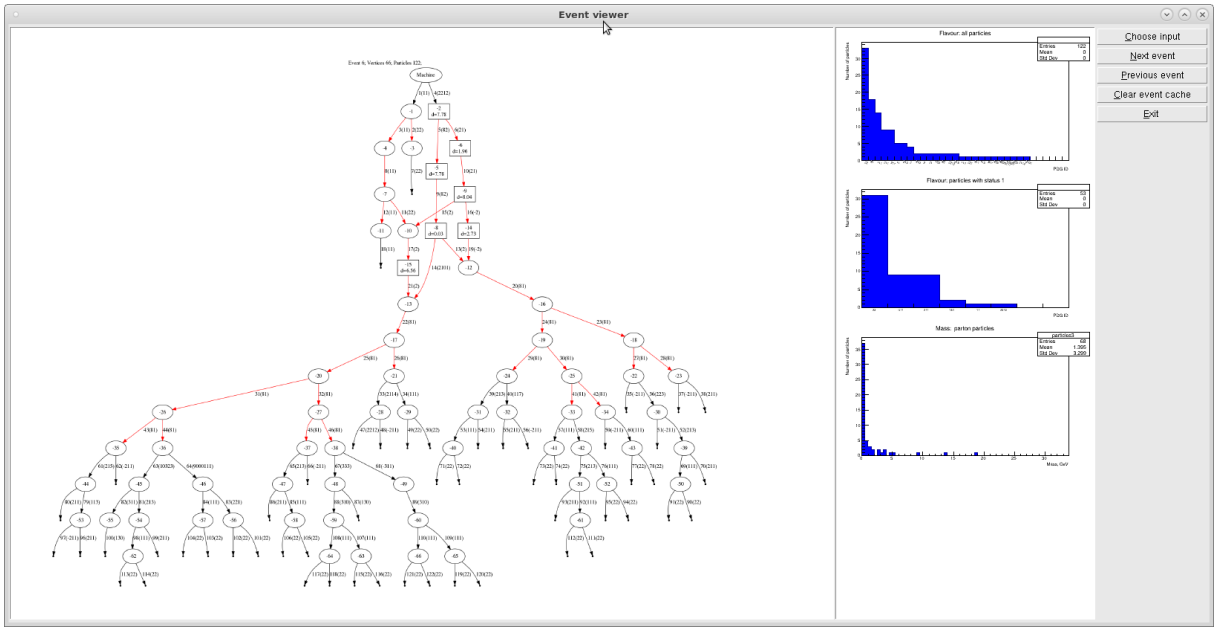

Figure 2. Visualization of $e^{-}p$ deep-inelastic scattering event simulated by the Herwig7 [19] Monte Carlo event generator. The interaction/decay vertices are depicted as ellipses (rectangles in case of violation of 4-momenta) with their vertex ids; the particles are shown as red/black arrows with their ids and flavour given in brackets. The arrows corresponding to the final state particles are shown with dots. The histograms displayed in the right panes show the distribution of particle flavours and masses in the event.

automatically generated documentation on the codes as well as extra material on specific topics, e.g. the LHEF format.

6. Conclusions

The HepMC3 library is designed to perform manipulations with event records of High Energy Physics Monte Carlo Event Generators (MCEGs). The library version 3.1.1(3.1.0) has been

released in April(March) 2019.

The I/O functionality of the library has been extended to support common input and output formats of HEP MCEGs, including formats used in Fortran HEP MCEGs, formats used in HepMC2 library and ROOT. The library is already supported by popular modern MCEGs (e.g. Sherpa, Pythia8, TAUOLA, PHOTOS) and can replace the older HepMC versions in many others.

References

[1] A. Verbytskyi, A. Buckley, D. Grellscheid, D. Konstantinov, J.W. Monk, L. Lnnblad, T. Przedzinski and W. Pokorski, HepMC3 event record library reference manual. In preparation (2019).

[2] A. Verbytskyi, A. Buckley, D. Grellscheid, D. Konstantinov, J.W. Monk, L. Lnnblad, T. Przedzinski and W. Pokorski, HepMC3 event record library, `http://hepmc.web.cern.ch/hepmc/`.

[3] M. Dobbs and J.B. Hansen, The HepMC C++ Monte Carlo event record for High Energy Physics. Comput. Phys. Commun. **134**, 41 (2001).

[4] J.G. Milhano et al., Lisbon Accord: a standard format for heavy-ion event generators. (2017).

[5] E. Boos et al., Generic user process interface for event generators. (2001). `arXiv:hep-ph/0109068`.

[6] J.R. Andersen et al., Les Houches 2013: Physics at TeV Colliders: Standard Model Working Group Report. (2014). `arXiv:1405.1067`.

[7] I. Antcheva et al., ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization. Comput. Phys. Commun. **180**, 2499 (2009). `arXiv:1508.0774`.

[8] T. Sjöstrand, S. Mrenna and P.Z. Skands, A brief introduction to PYTHIA 8.1. Comput. Phys. Commun. **178**, 852 (2008). `arXiv:0710.3820`.

[9] Kitware, CMake, `https://cmake.org/`.

[10] D.van Heesch, Doxygen, `http://www.doxygen.nl`.

[11] Graphviz project, Graphviz - Graph Visualization Software, `https://www.graphviz.org`.

[12] D. Chall et al., High energy physics packages for Mac, `https://github.com/davidchall/homebrew-hep`.

[13] Archlinux project, The Arch User Repository (AUR) , `https://aur.archlinux.org/`.

[14] Debian project, Debian NEW and BYHAND Packages , `https://ftp-master.debian.org/new.html`.

[15] Fedora project, Extra Packages for Enterprise Linux (EPEL), `https://fedoraproject.org/wiki/EPEL`.

[16] T. Gleisberg et al., Event generation with SHERPA 1.1. JHEP **02**, 007 (2009). `arXiv:0811.4622`.

[17] JETSCAPE Collaboration, S. Cao et al., Multistage Monte-Carlo simulation of jet modification in a static medium. Phys. Rev. **C96**, 024909 (2017). `arXiv:1705.0005`.

[18] L. Lönnblad, ThePEG, Pythia7, herwig++ and Ariadne. Nucl. Instrum. Meth. **A559**, 246 (2006).

[19] J. Bellm et al., Herwig 7.0/Herwig++ 3.0 release note. Eur. Phys. J. **C76**, 196 (2016). `arXiv:1512.0117`.

[20] T. Sjöstrand, S. Mrenna and P.Z. Skands, PYTHIA 6.4 physics and manual. JHEP **05**, 026 (2006). `arXiv:hep-ph/0603175`.

[21] S. Jadach, J.H. Kühn and Z. Was, TAUOLA: a library of Monte Carlo programs to simulate decays of polarized tau leptons. Comput. Phys. Commun. **64**, 275 (1990).

[22] E. Barberio and Z. Was, PHOTOS: A Universal Monte Carlo for QED radiative corrections. Version 2.0. Comput. Phys. Commun. **79**, 291 (1994).

[23] W. Kilian, T. Ohl and J. Reuter, WHIZARD: Simulating Multi-Particle Processes at LHC and ILC. Eur. Phys. J. **C71**, 1742 (2011). `arXiv:0708.4233`.

[24] H. Jung, Hard diffractive scattering in high-energy *ep* collisions and the Monte Carlo generator RAPGAP. Comput. Phys. Commun. **86**, 147 (1995).

[25] D.J. Lange, The EvtGen particle decay simulation package. Nucl. Instrum. Meth. **A462**, 152 (2001).

[26] G. Amadio et al., GeantV alpha release. J. Phys. Conf. Ser. **1085**, 032037 (2018).

[27] E. Rodrigues, The Scikit-HEP Project, 23rd International Conference on Computing in High Energy and Nuclear Physics (CHEP 2018) Sofia, Bulgaria, July 9-13, 2018. (2019). Also in preprint 1905.00002. `arXiv:1905.0000`.

[28] N. Davidson et al., MC-TESTER v. 1.23: A Universal tool for comparisons of Monte Carlo predictions for particle decays in high energy physics. Comput. Phys. Commun. **182**, 779 (2011). `arXiv:0812.3215`.

[29] A. Buckley et al., Rivet user manual. Comput. Phys. Commun. **184**, 2803 (2013). `arXiv:1003.0694`.