ЯДЕРНАЯ ФИЗИКА И ИНЖИНИРИНГ, 2023, том 14, № 1, с. 56–60

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ В ЯДЕРНЫХ ТЕХНОЛОГИЯХ

УДК 004.043

MCDST: УНИФИЦИРОВАННЫЙ ФОРМАТ ХРАНЕНИЯ ДАННЫХ МОДЕЛИРОВАНИЯ СТОЛКНОВЕНИЙ ТЯЖЕЛЫХ ИОНОВ

© 2023 г. Е. А. Кузина[a], *, Г. А. Нигматкулов[a, b]

[a] *Национальный исследовательский ядерный университет "МИФИ", Москва, 115409 Россия*

[b] *Объединеный институт ядерных исследований, Дубна, Московская обл., 141980 Россия*

**E-mail: ekaterina.kuzina2@ya.ru*

Поступила в редакцию 22.09.2022 г.
После доработки 22.09.2022 г.
Принята к публикации 03.10.2022 г.

Монте-Карло генераторы, используемые для моделирования столкновений релятивистских тяжелых ионов, имеют нестандартизованные выходные форматы. Как следствие, данные, полученные в результате запуска разных генераторов, требуют разные подходы к их обработке. Для разрешения этой сложности реализованы унифицированный формат и соответствующий интерфейс для чтения и обработки данных. Под унифицированностью понимается возможность преобразования любого выходного формата в указанный. Представленное кроссплатформенное решение, программная библиотека McDst, разработано в объектно-ориентированной парадигме с использованием языка C++ и библиотек CERN ROOT. Библиотека распространяется в формате исходного кода и может быть расширена конечным пользователем. Представлены структура формата и сценарии использования библиотеки.

Ключевые слова: Монте-Карло генератор, столкновения тяжелых ионов, формат хранения данных, программная библиотека, обработка данных

DOI: 10.56304/S2079562922030289

ВВЕДЕНИЕ

Все Монте-Карло генераторы, используемые для моделирования столкновений релятивистских тяжелых ионов, имеют особые выходные форматы моделируемых событий. В связи с этим подходы к чтению и обработке данных должны зависеть от конкретного формата. С целью упрощения, оптимизации и стандартизации анализа данных разработана библиотека McDst.

McDst – это формат хранения данных моделирования, предоставляющий интерфейс для быстрого и простого извлечения информации о столкновениях ионов и образовавшихся частицах: например, угол наклона плоскости реакции, сорт и импульс частицы. Данное решение предполагается использовать в виде динамической библиотеки, но его также можно интегрировать в пользовательское программное обеспечение любым другим способом, поскольку исходный код распространяется под открытой лицензией GPL-3.0.

Библиотека McDst разработана в объектно-ориентированной парадигме. Она представлена набором связанных классов. Для реализации используются язык C++ и библиотеки CERN ROOT [1].

ФОРМАТ MCDST

Файл с данными в формате McDst (расширение .mcDst.root) содержит экземпляр run класса McRun и экземпляр McDst – ROOT TTree. Класс McRun используется для хранения названия генератора, названия кода, используемого для распада нестабильных частиц, и следующих параметров работы генератора:

1. Массы, заряда и импульса налетающего ядра и ядра–мишени;

2. Интервала значений прицельного параметра;

3. Интервала значений угла наклона плоскости события;

4. Сечения взаимодействия.

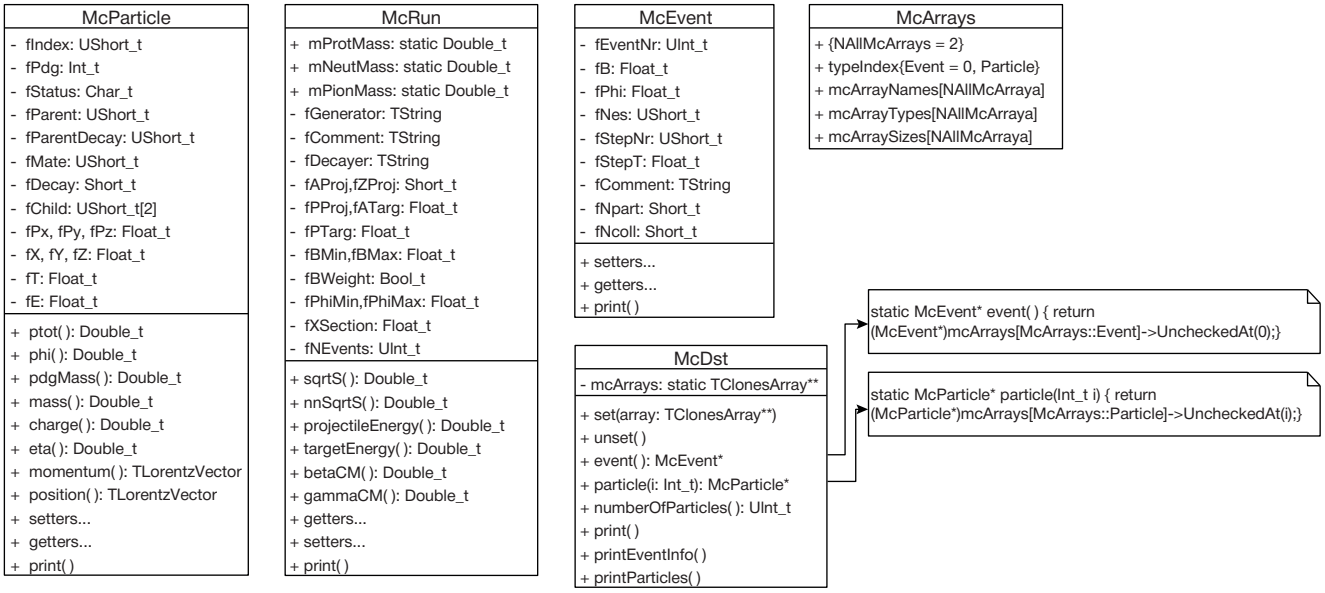

Рис. 1. Диаграмма классов McArrays, McDst, McRun, McEvent и McParticle.

Для хранения данных используется стандартный для физики высоких энергий иерархический контейнер TTree. Экземпляр McDst класса TTree содержит два экземпляра класса TBranch: для хранения информации о событиях и частицах. Экземпляры класса TLeaf являются полями классов McEvent (в случае TBranch Event) и McParticle (в случае TBranch Particle). Классы McEvent и McParticle содержат методы для извлечения параметров событий и частиц соответственно. Поля McEvent отвечают следующим параметрам:

1. Номеру события;

2. Прицельному параметру;

3. Углу наклона плоскости реакции;

4. Количеству шагов по времени;

5. Номеру шага;

6. Длительности шага;

7. Количеству провзаимодействовавших частиц;

8. Количеству бинарных столкновений.

McParticle хранит следующие параметры сгенерированных частиц:

1. Индекс частицы и её родительской частицы;

2. PDG код [2];

3. Индекс частицы, с которой произошло последнее столкновение;

4. Индексы первой и последней дочерних частиц;

5. Компоненты импульса;

6. Энергия;

7. Время и координаты последнего взаимодействия.

Класс McDst используется в качестве промежуточного буфера для хранения информации о текущем событии при обработке .mcDst.root файла. Класс McDst содержит указатель на экземпляры TClonesArray с указателями на объекты классов McEvent и McParticle.

На рис. 1 приведена диаграмма классов представленной части библиотеки McDst. Экземпляры McEvent и McParticle могут быть получены из поля McDst::mcArrays методами McDst::event() и McDst::particle(Int_t) соответственно. McArrays — это вспомогательный класс для настройки работы с ROOT TClonesArray.

СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ БИБЛИОТЕКИ MCDST

Чтение данных формата McDst

Класс McDstReader предназначен для чтения данных McDst. Он содержит указатель на экземпляр класса TChain, который представляет собой набор файлов. Чтение следует начинать и завер-

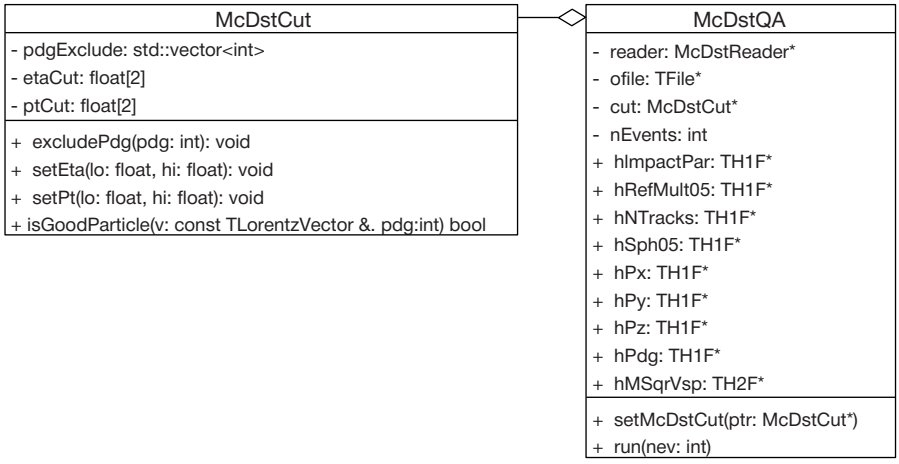

Рис. 2. Диаграмма классов McDstQA and McDstCut.

шать вызовами методов McDstReader::Init() и McDstReader::Finish() соответственно.

Листинг обработки данных приведен в Приложение 1. В строке 5 задается входной файл формата McDst. Также доступна передача файла, содержащего список путей к файлам McDst. Для ускорения обработки можно использовать метод McDstReader::setStatus(const Char_t*, Int_t). Установка значения в “0” означает отмену чтения данной “ветки”, в “1” — наоборот. В строках 8–10 листинга показана возможность сброса (выключения чтения) и добавления (включения) “веток”. Цикл “for” по событиям можно настроить, вернув количество событий во всех обрабатываемых файлах (строка 12).

Для чтения события используется метод McDstReader::loadEntry(Long64_t). Информация об этом событии и соответствующих частицах предоставляется полем McDstReader::mMcDst и может быть получена из него, как показано в строках 24 и 30 листинга. Конкретные параметры событий и частиц извлекаются путем вызова методов классов McEvent и McParticle, как в строках 31, 33, 34, 39.

Проверка качества данных

Библиотека McDst содержит класс McDstQA, предоставляющий интерфейс для проверки качества данных. В классе McDstQA определен указатель на McDstReader, который используется для заполнения ROOT гистограмм по набору атрибутов данных, например, множественности событий и компонент импульсов частиц. Чтобы исключить из обработки данные по определенным параметрам, реализован класс McDstCut. Экземпляр McDstCut с установленными ограничениями на данные передается экземпляру McDstQA с помощью метода McDstQA::setMcDstCut(McDstCut*). Метод McDstQA::run() выполняет обработку и создает ROOT файл с заполненными гистограммами.

Диаграмма классов McDstQA и McDstCut приведена на рис. 2.

В текущей реализации класс McDstCut содержит ограничения только на значения PDG кода частицы, псевдобыстроты и поперечного импульса. Список параметров можно настроить.

Преобразования форматов данных

Разработаны конвертеры для данных столкновений тяжелых ионов, моделируемых Монте-Карло генераторами. Имеются конвертеры для выходного формата PYTHIA 8 [3] и выходных файлов .f13 и .f14 UrQMD [4, 5]. Указанные конвертеры включены в библиотеку McDst. Библиотека может быть расширена конвертерами для требуемых генераторов конечным пользователем.

ЗАКЛЮЧЕНИЕ

Разработан унифицированный формат хранения данных моделирования столкновений тяжелых ионов. Реализованный программный интерфейс охватывает распространенные задачи работы с этим типом данных. Исходный код и краткое руководство доступны в GitHub репозитории [6]. Для компиляции динамической библиотеки под

ОС семейства Linux (с расширением .so) в репозитории также представлен make-файл для автоматизации сборки с помощью GNU make [7]. Обязательным требованием использования библиотеки McDst является предустановка библиотек CERN ROOT.

Приложение 1

ЛИСТИНГ ОБРАБОТКИ ДАННЫХ ФОРМАТА MCDST

```
void analyseMcDst(const Char_t *inFile = "test.mcDst.root",
                  const Char_t *oFileName = "oProcTest.root") {
  gSystem->Load("libMcDst.so");

  McDstReader* myReader = new McDstReader(inFile);
  myReader->Init();

  myReader->setStatus("*",0);
  myReader->setStatus("Event",1);
  myReader->setStatus("Particle",1);

  Long64_t events2read = myReader->chain()->GetEntries();
  TFile *oFile = new TFile(oFileName, "RECREATE");
  // Histograms to fill
  TH2D *hImpactParVsNch = new TH2D("hImpactParVsNch",
                                   "Impact parameter vs. Nch;Nch;Impact parameter (fm)",
                                   300, -0.5, 599.5, 130, 0., 13.);
  TH1D *hPionMom = new TH1D("hPionMom","Momentum of #pi;p (GeV/c);Entries",
                            100, 0., 2.);
  // Event loop
  for(Long64_t iEvent=0; iEvent<events2read; iEvent++) {
    myReader->loadEntry(iEvent);
    McDst *dst = myReader->mcDst();
    McEvent *event = dst->event();

    Int_t nTracks = dst->numberOfParticles();
    Int_t NumOfCharged = 0;
    // Track loop
    for(Int_t iTrk=0; iTrk<nTracks; iTrk++) {
      McParticle *particle = dst->particle(iTrk);
      if ( particle->charge() ) {
        NumOfCharged++;
        if ( particle->pdg() == 211 ) {
          hPionMom->Fill( particle->ptot() );
        }
      }
    }

    hImpactParVsNch->Fill( NumOfCharged, event->impact() );
  }
  oFile->Write();
  oFile->Close();
  myReader->Finish();
}
```

БЛАГОДАРНОСТИ

Работа выполнена при поддержке Министерства науки и высшего образования Российской Федерации, проект “Фундаментальные свойства элементарных частиц и космология” № 0723-2020-0041, и программы “Приоритет 2030” НИЯУ МИФИ. Работа выполнена с использованием ресурсов гетерогенной вычислительной платформы HybriLIT ОИЯИ (ЛИТ)

(http://hlit.jinr.ru) и высокопроизводительного вычислительного центра НИЯУ МИФИ.

СПИСОК ЛИТЕРАТУРЫ/REFERENCES

1. *Brun R., Rademakers F.* // Nucl. Instrum. Methods Phys. Res., Sect. A. 1997. V. 389 (1). P. 81–86.
2. *Workman R.L. et al. (Particle Data Group)* // Prog. Theor. Exp. Phys. 2022. V. 2022 (8). P. 083C01.
3. *Sjöstrand T. et al.* // Comput. Phys. Commun. 2005. V. 191. P. 159.
4. *Bass S.A. et al.* // Prog. Part. Nucl. Phys. 1998. V. 41. P. 255–369.
5. *Bleicher M. et al.* // J. Phys. G. 1999. V. 25 (9). P. 1859–1896.
6. https://github.com/nigmatkulov/McDst.
7. *Stallman R., McGrath R., Smith P.D.* GNU Make: a Program for Directing Recompilation. 1996. Massachusetts: Free Software Foundation. https://www.gnu.org/software/make/manual/make.pdf).

MCDST: A Unified Storage Format for Heavy Ion Collision Simulated Data

E. A. Kuzina[1, *] and G. A. Nigmatkulov[1, 2]

[1]*National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Moscow, 115409 Russia*
[2]*Joint Institute for Nuclear Research, Dubna, Moscow oblast, 141980 Russia*
**e-mail: ekaterina.kuzina2@ya.ru*

Received September 22, 2022; revised September 22, 2022; accepted October 3, 2022

Abstract—Monte Carlo generators for relativistic heavy–ion collisions have non–standardize output formats. Thus, data that are simulated by different generators require different processing techniques. To resolve this issue, a unified storage format with an interface to read and process data is implemented. Unification should be interpreted as the ability to convert any generator output into discussed format. The presented cross–platform solution, McDst software library, is developed on the object–oriented paradigm using C++ language and CERN ROOT libraries. It is distributed in a form of source code and can be extended by end–user. The structure of the format and use cases for the library are presented.

Keywords: Monte Carlo generator, heavy ion collisions, data storage format, software library, data processing