# The Virtual Monte Carlo

I. Hřivnáčová
*IPN, Orsay, France*
D. Adamová
*NPI, ASCR, Rez, Czech Republic*
V. Berejnoi, R. Brun, F.Carminati, A. Fassò, E. Futó, A. Gheata, A. Morsch
*CERN, Geneva, Switzerland*
I. González Caballero
*IFCA, Santander, Spain*
*For the ALICE Collaboration*

The concept of Virtual Monte Carlo (VMC) has been developed by the ALICE Software Project to allow different Monte Carlo simulation programs to run without changing the user code, such as the geometry definition, the detector response simulation or input and output formats. Recently, the VMC classes have been integrated into the ROOT framework, and the other relevant packages have been separated from the AliRoot framework and can be used individually by any other HEP project. The general concept of the VMC and its set of base classes provided in ROOT will be presented. Existing implementations for Geant3, Geant4 and FLUKA and simple examples of usage will be described.

## 1. Introduction

The concept of Virtual Monte Carlo (VMC) has been gradually developed by the ALICE Software project [1]. From the beginning, the ALICE collaboration has adopted a strategy for the development of the simulation framework that would allow a smooth transition from the currently used transport code, Geant3 [2], to new ones Geant4 [4] and Fluka [6]. Instead of maintaining the Geant3 based code written in FORTRAN and developing in parallel a new framework, based on a new simulation program, the user code was gradually migrated from FORTRAN to C++ and a general C++ interface to a transport Monte Carlo (MC) was developed.

The VMC development went through the following phases:

1. The C++ class, TGeant3, providing access to Geant3 data structures (common blocks) and functions was introduced. This provided a starting point for a full migration of the user code from FORTRAN to C++.

2. The abstract C++ class, AliMC, was defined as a generalization of TGeant3. This gave the initial step for the development of the Geant4 interface and the explicit Geant3 dependencies in the user code were also taken away. However, the implementations of the AliMC interface for both Geant3 and Geant4 were dependent on the ALICE software.

3. The interfaces to the user Monte Carlo application were introduced. The dependence of the implementations of the AliMC interface on the ALICE software could then be removed and the VMC was also made available to non-ALICE users.
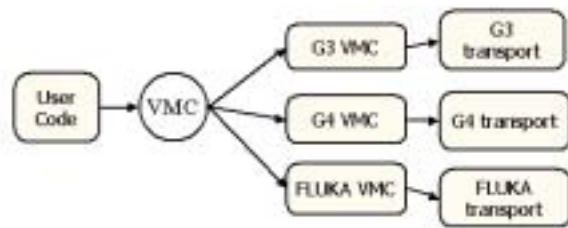


Figure 1: The Virtual Monte Carlo concept.

## 2. Architecture

### 2.1. The VMC concept

With the VMC concept the user Monte Carlo application can be defined independently of a specific transport code (see Fig. 1). It can then be run with all supported Monte Carlos, without changing the user code, ie., the geometry definition, the detector response simulation and input or output formats. The selection of a concrete Monte Carlo (Geant3, Geant4 or Fluka) is made dynamically at run time.

The VMC is based on the ROOT system [8], which is used mainly for scripting and dynamical loading of libraries. Once the VMC application has been defined, simulations can be run interactively from the Root UI or using Root macros.
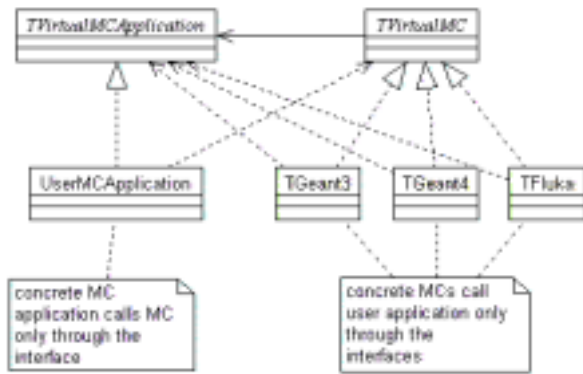
Figure 2: The Virtual Monte Carlo design.

## 2.2. Design

In order to completely decouple the user code from the concrete Monte Carlo, the interfaces to both the Monte Carlo itself and to the user application code have been introduced as shown in Fig. 2. In the following subsections, all interfaces will be discussed in detail.

### 2.2.1. Virtual MC

The Virtual MC interface (class TVirtualMC) was the first interface written and it is the most robust one. It has been defined as a generalization of Geant3 functions for the definition of simulation tasks and it provides:

- Methods for building and accessing geometry

- Methods for building and accessing materials

- Methods for setting physics

- Methods for accessing transported particle properties during stepping

- Methods for run control

The implementations of the Virtual MC for concrete transport programs are part of the VMC distribution and are provided for the user. At the present time, the Geant3 VMC and the Geant4 VMC are in distribution, the Fluka VMC will be available soon.

### 2.2.2. Virtual MC Application

The Virtual MC Application interface (class TVirtualMCApplication) is the interface to a user application code. It defines user actions at each stage of a simulation run:

> Construct geometry
> Init geometry
> Generate primaries
> Begin event

> Begin primary
> Pre Track
> Stepping
> Post Track
> Finish primary
> Finish event

The implementation of the Virtual MC Application completely defines the user application and has to be provided by the user.

### 2.2.3. Virtual MC Stack

The Virtual MC Stack interface (class TVirtualMCStack) defines the interface to a user defined particles stack. Users can choose one of the concrete stack classes provided in the VMC examples or can implement their own stack class.

### 2.2.4. Virtual MC Decayer

The last interface in the VMC, the Virtual MC Decayer (class TVirtualMCDecayer), defines the interface to the external decayer. The implementation of this interface by a user is optional.

## 3. Use of VMC

The user VMC application code is written by implementing the MC Application class. In the case of very simple applications the user can write everything in the one class. In more complex cases it can be convenient to define the user application class as a composition of more action classes, as is shown in the VMC examples described in section 5.1.

In this section, three examples of a user code based on the VMC will be given.

## 3.1. Geometry construction

In Example I, an example of a geometry definition is given. The geometry is defined by calls to the Virtual MC interface. In the first block, a volume named "TRTU" of the shape tube of an inner radius of 0 cm and outer radius of 60 cm, is created and associated with a material defined by the tracking medium identifier "idAl". In the second block, this volume is placed at the position (-100 cm, 0 cm, 0 cm) in the mother volume named "EXPH".

The functions for geometry building are compatible with Geant3 (both by name and in the list of parameters). The VMC also uses the Geant3 system of default physical units. In future, using the Root geometrical modeller directly will be possible as an alternative to this Geant3 style. (For more details, see section 6.)

Example I  Example of a user code for geometry construction using the VMC

```
void MyMCApplication::ConstructGeometry()
{
    // Create tracker tube volume
    Double_t trackerTube[3];
    trackerTube[0] = 0.;
    trackerTube[1] = 60.;
    trackerTube[2] = 50.;
    gMC− >Gsvolu("TRTU", "TUBE", idAl, trackerTube, 3);

    // Place tracker tube volume
    Double_t posX = -100.;
    Double_t posY = 0.;
    Double_t posZ = 0.;
    gMC− >Gspos("TRTU", 1, "EXPH", posX, posY, posZ, 0, "ONLY");
}
```

## 3.2. Primary particles

In Example II, an example of how to define primary particles is given. This is done by calls to the Virtual MC Stack interface. The particle type (proton, electron, ...) is defined using the PDG encoding and particle static properties (mass, charge, ...) are taken from the particle database in ROOT (represented by the TDatabasePDG class).

## 3.3. Detector response

In Example III, an example of a user stepping function is shown. This function is called by MC at each step. In this example the properties of the particle transported are obtained via calls to the Virtual MC interface and then saved in the user own hits objects. For large detectors it is recommended to delegate this function to stepping functions defined in subdetector classes.

## 4. Available MCs

In this section we will outline the present status and give a short description of the implementations of the Virtual MC for the three transport MCs: Geant3, Geant4 and Fluka.

## 4.1. Geant3 VMC

The Geant3 program [2] was written to describe the passage of elementary particles through matter. Originally designed for high energy physics experiments, it has also found applications outside this domain in the areas of medical and biological sciencies, radioprotection and astronautics. The first version was released in 1974 and the system was developed with some continuity over 20 years till the last release 3.21 in 1994. It has become a popular and widely used tool in the HEP community.

The Geant3 VMC, which implements the Virtual MC interface to the Geant3 program, is provided within a single package "geant3" together with Geant321 itself. This "geant3" package is available from the ROOT Web site [8].

As the Virtual MC was largely inspired by Geant3, its implementation for Geant3 was straightforward and has no limitations.

Besides the implementation of the Virtual MC, the Geant3 VMC also includes the Geant3 Geometry Browser [3], a GUI which provides a variety of functions, namely:

- visualization of the geometry volumes tree

- drawing of volumes and interactive setting of drawing options

- browsing material and tracking medias parameters

- browsing applied cuts and activated physics processes

- plotting of dE/dx and cross-sections for a selected physical process

The implementation is based on the Root GUI classes.

Example II  Example of a user code for primary particles definition using the VMC

```
void MyMCApplication::GeneratePrimaries()
{
    // Define particle properties:
    // PDG encoding: pdg
    // position: vx, vy, vz, t
    // momentum: px, py, pz, e
    // ...

     // Add particle to MC stack
    gMC− >GetStack()
            − >SetTrack(toBeDone, -1, pdg, px, py, pz, e, vx, vy, vz,t, ...);
}
```

Example III  Example of a user code for a detector response simulation using the VMC

```
void MyMCApplication::Stepping()
{
    // Get current volume ID
    Int_t copyNo;
    Int_t id = gMC− >CurrentVolID(copyNo);

    // Check if step is performed in the sensitive volume
    if (id != fSensitiveVolumeID) return false;

    // Get track position
    Double_t x, y, z;
    gMC− >TrackPosition(x, y, z);

    // Get energy deposit
    Double_t edep = gMC− >Edep();

     // Create user hit
    mySD− >AddHit(x, y, z, edep);
}
```

## 4.2. Geant4 VMC

The Geant4 project [4] was started in 1994, the first production version was released in 1998 and the system is continuously under developement by the Geant4 Collaboration. Its areas of application include particle and nuclear physics experiments, medical, accelerator and space physics studies.

The Geant4 VMC, which implements the Virtual MC interface for the Geant4 program, is provided within the "geant4_vmc" package and requires a prior Geant4 installation. This "geant4_vmc" package is available from the ROOT Web site [8].

The implementation of the interface to MC for Geant4 was presented at the CHEP 2001 conference [10]. The design, the implementation and also the problems arising from the G3toG4 approach and their foreseen solutions were discussed. Despite improvements and design changes, the structure and the components of the package presented there can be found in the current Geant4 VMC package. The major change applied since then was that the dependencies on the ALICE classes in AliGeant4 have been replaced by the dependencies on the interfaces to a user application. This meant that all classes from AliGeant4 could be moved to the experiment independent part, TGeant4, that has been then renamed Geant4 VMC.

The main improvement from that time was in minimizing the limitations of the G3toG4 tool in the Geant4 4.0 release. New classes to support the reflection symmetry and also a limited support for the "MANY" volumes positions have been introduced. The G3toG4 tool does not resolve positions with "MANY" automatically, however a user can specify the overlapping volumes using the G4gsbool() function and then the overlaps for these volumes and their daughters are automatically resolved using Boolean solids. The volume with a "MANY" position can only have this position. The corresponding function Gsbool() has been added to the Virtual MC interface.

The VMC interface provides a common denominator for all implemented MCs and cannot cover all commands available in a Geant4 user session through the Geant4 UI. Switching between the Root UI and the Geant4 UI gives the VMC user the possibility of working with the native Geant4 UI when needed or desired.

It is also possible to process a foreign command or a foreign macro in both UIs, for example the Root commands and macros can be processed in the Geant4 UI and vice versa.

In a similar way as the Geant3 VMC, the Geant4 VMC also includes the Geant4 Geometry Browser. It was implemented in an analogous way. It provides the same functionality for browsing geometry. It does not include the panels that allow to browse the activated physical processes and their characteristics.

In addition, the Geant4 VMC also includes an XML convertor. With this convertor the Geant4 geometry can be exported to XML in the AGDD format [11] and then browsed and visualized using the GraXML tool [12]. The XML convertor classes are independent of the Geant4 VMC, they can be compiled in a separate library and also used with a native Geant4 application.

## 4.3. Fluka VMC

The history of FLUKA [6] goes back to 1962-1967. Over the years it went through three different generations, which can be roughly identified as the FLUKA of the '70s, the FLUKA of the '80s and today's FLUKA as a fully integrated particle physics Monte Carlo simulation package. It has many applications in high energy experimental physics, engineering, shielding, detector and telescope design, cosmic ray studies, dosimetry, medical physics and radio-biology.

The Fluka VMC is now under development within the ALICE collaboration and the FLUKA team. The development version of the VMC implementation is in the TFluka package of AliRoot.

Most of the functionality required by the VMC is already fully operational:

- Functions for building and accessing geometry (through Flugg)

- Functions for accessing transported particle properties during stepping

- Recording particles in the VMC stack

- Functions for run management

Under development are:

- Functions for selecting activated physics processes

- Interface to the external decayer

The geometry part of the Fluka VMC is implemented with the use of the Geant4 VMC and the Flugg tool [7]. Flugg was developed a few years ago so that the Geant4 geometry could be used directly in the Fluka particle transport. In the near future this part will be replaced by the Root geometrical modeller. (For more details, see section 6.)

## 5. Examples

## 5.1. Examples provided with the VMC

To demonstrate the use of the VMC, Geant4 novice examples N01, N02 and N03 were rewritten in the VMC framework. The Geant4 novice examples were chosen in order to show the similarities and differences between the two frameworks. Having in parallel, both the Geant4 VMC and the Geant4 native application, allows to verify the consistence of the results and to compare performancies.

The VMC examples demonstrate the implementation of the user MC application and MC stack classes. While in the first example all functions of the MC application are implemented directly in the Ex01MCApplication class, in the other examples the MC application class is defined as being a composition of more action classes (e.g the detector construction class) and it delegates most of its functions to its components.

All examples are executed by processing the provided Root macros. The macros for running the example E01 with Geant3 and Geant4 (run_g3.C and run_g4.C) are shown in Example IV. In both macros, all necessary libraries are first dynamically loaded, then the user MC application is created and initialized with a MC specific configuration macro (g3Config.C or g4Config.C) shown in Example V. After initialization, a simulation run with the chosen concrete MC is executed for the specified number of events.

## 5.2. AliRoot

AliRoot is the ALICE off-line framework for simulation, reconstruction and analysis. The simulation in AliRoot is fully based on the VMC. Hence the AliRoot framework can be given as a complex example of a VMC application.

The AliRoot framework will not be discussed in more detail in this paper, but the reader is directed to the other CHEP03 conference papers related to this subject: "The AliRoot Framework, status and perspectives" [13], "Simulation in ALICE" [14], "ALICE experience with Geant4" [15].

In the following, a few figures demonstrating the use of AliRoot with two MCs, Geant3 and Geant4, will be given. In Fig. 3 and Fig. 4 the geometries for two ALICE detector subsystems (the ITS and the Dimuon arm) are shown - the first drawn with Geant3, the second drawn with Geant4 and GraXML. In Fig. 5, the x and z distributions of hits in the TPC subsystem are shown for two transport MCs: Geant3 and Geant4, giving qualitatively similar results.

Example IV Root macros, run_g3.C and run_g4.C, that show how to run the VMC example E01. MC specific parts are given in the parallel columns.

```
{
    // Load basic libraries
    gROOT->LoadMacro("basiclibs.C");
    basiclibs();

// Load Geant3 libraries            // Load Geant4 libraries
gROOT->LoadMacro("g3libs.C");       gROOT->LoadMacro("g4libs.C");
g3libs();                           g4libs();

    // Load this example library
    gSystem->Load("libexample01");

    // MC application
    Ex01MCApplication* appl
        = new Ex01MCApplication("Example01", "The example01 MC application");

appl->InitMC("g3Config.C");         appl->InitMC("g4Config.C");

    appl->RunMC(1);
}
```
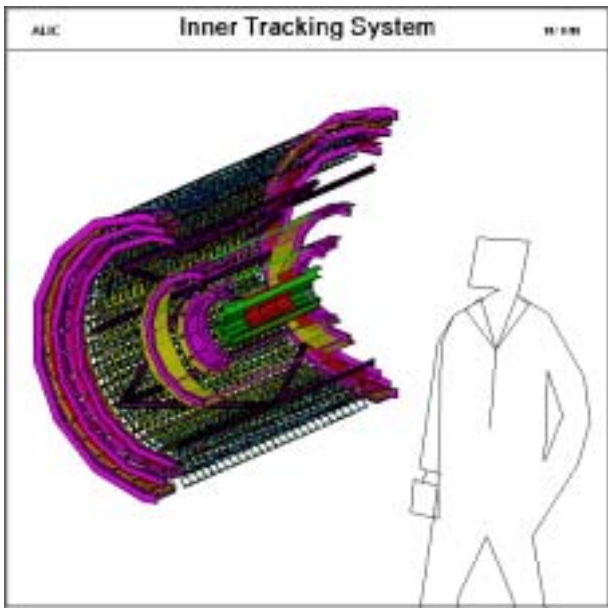


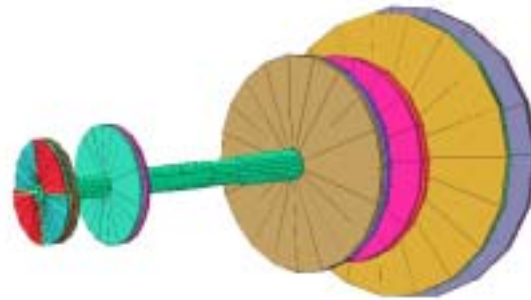Figure 3: Geometry for ITS detector drawn with Geant3.



Figure 4: Geometry for the Dimuon Arm spectrometer drawn with Geant4 and GraXML.

HEP. It is called TGeo and has been included in ROOT [16]. This new geometrical modeller is currently being integrated into the VMC.

The VMC user will then have also the possibility to define geometry directly using the new geometrical modeller. The current interface will be kept for backward compatibility. Apart from making a geometry construction via the VMC more user-friendly, this integration will also improve the performance of the MC simulation. The new modeller has been optimised for performance and runs faster than Geant3 for most geometries tested.

## 6. Future

Over the past two years the ALICE Offline project, in close collaboration with the ROOT team, has developed a new multi-purpose geometrical modeller for

Example V  Root configuration macros: g3Config.C and g4Config.C

---

**g3Config.C:**

void Config()
{
    *// Geant3 VMC*
    new TGeant3("C++ Interface to Geant3");
}

---

**g4Config.C:**

void Config()
{
    *// Run Configuration for Geant4*
    TG4RunConfiguration* runConfiguration = new TG4RunConfiguration();

    *// Geant4 VMC*
    new TGeant4("TGeant4", "The Geant4 Monte Carlo", runConfiguration);
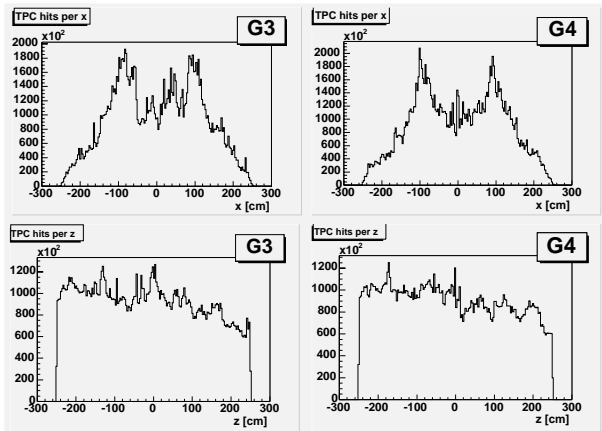}

---



Figure 5: The x,z-distributions of hits in the TPC detector.



Figure 6: The Virtual Monte Carlo and TGeo integration.

For Geant3 and Fluka, the integration of TGeo is being made by replacing the native geometry with the TGeo one. The new version of Geant3 with the new Root geometry package is close to completion and the work on the integration of TGeo into Fluka is already well advanced.

For Geant4, the object-oriented methodology can be exploited and TGeo can be integrated with Geant4 through an abstract interface to the Geant4 geometry. At present, the Geant4 geometry navigator class is not based on an abstract interface, its generalization will be required to allow an alternative implementation using TGeo. Discussions with the Geant4 team have started a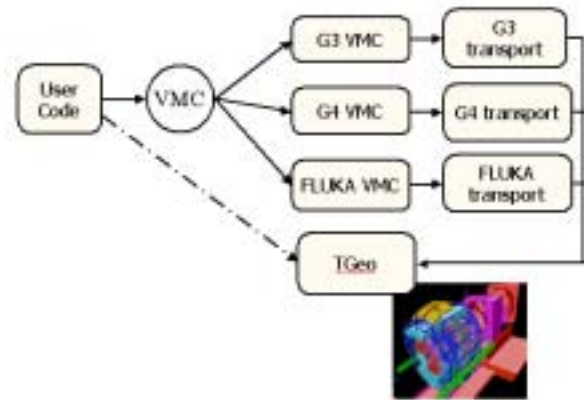nd the design and prototype implementation for the abstract navigator and transportation are now part of the Geant4 developments planned for the 6.0 release [5].

In the meantime, a convertor from Root geometry to Geant4 native geometry, RootToG4, has been developed. It is already operational and close to completion. It will allow the migration from the current VMC geometry definitions to TGeo, before the solution via the abstract navigator is available.

The new concept of the VMC including the Root geometrical modeller is shown in Fig. 6.

## 7. Distribution

The VMC is distributed with the ROOT system [8]. It consists of the following packages:

- mc: the core package (interfaces)

- geant3: Geant321 + Geant3 VMC

- geant4_vmc: Geant4 VMC

- examples

The "mc" package is directly included in ROOT, "geant3" and "geant4_vmc" are available from the ROOT CVS server [9] as independent modules and the package with the examples is provided within "geant4_vmc".

The tarballs with sources are also available from the VMC Web page [17].

## 8. Conclusions

The Virtual Monte Carlo provides a simulation framework which is independent of any concrete MC and is based on the ROOT system. Geant3 and Geant4 have already been integrated and the same work with Fluka is in progress.

The main advantage with VMC is that the user can run the same simulation program with three different transport MCs. The obvious consequence is that the different models can be compared and better understood. VMC also facilitates the use of less user-friendly tools such as Geant3 and Fluka and - being inspired from Geant3 - it is also suitable for users starting with existing Geant3 applications.

The integration of the Root geometrical modeller will give the user a new means for geometry definition, browsing, visualization and also verification directly within the scope of the VMC.

## Acknowledgments

## References

[1] http://AliSoft.cern.ch/offline

[2] Application Software Group, CN Div., "GEANT Detector Description and Simulation Tool (Version 3.21)", CERN Program Library W5013. http://wwwinfo.cern.ch/asd/geant

[3] http://alisoft/cern.ch/people/morsch/Geant3GUI.html

[4] S. Agostinelli et al., "Geant4 - A Simulation Toolkit", CERN-IT-20020003, KEK Preprint 2002-85, SLAC-PUB-9350, submitted to Nuclear Instruments and Methods NIM A. http://wwwinfo.cern.ch/asd/geant4/geant4.html

[5] http://wwwasd.web.cern.ch/wwwasd/geant4/source/planned_features.html

[6] http://pcfluka.mi.infn.it

[7] http://www.fluka.org/projects/flugg

[8] http://root.cern.ch

[9] http://root.cern.ch/root/CVS.html

[10] I. Hrivnacova for the Alice Collaboration: "GEANT4 in the AliRoot Framework", Proceedings of CHEP 2001, Science Press Beijing New York, p.534.

[11] http://atlas.web.cern.ch/Atlas/GROUPS/DATABASE/detector_description/index.html

[12] J. Hrivnac: "GraXML - Modular Geometric Modeller", Presented at CHEP 2003, La Jolla, PSN THJT009. http://hrivnac.home.cern.ch/hrivnac/Activities/Packages/GraXML

[13] R. Brun, P. Buncic, F. Carminati, A. Morsch, F. Rademakers, K. Safarik for the Alice Collaboration: " The AliRoot Framework, status and perspectives", Presented at CHEP 2003, La Jolla, PSN THJT003.

[14] A. Morsch for the Alice Collaboration: "Simulation in ALICE.", Presented at CHEP 2003, La Jolla, PSN TUMT004.

[15] F. Carminati, I. Gonzalez, I. Hrivnacova, A. Morsch for the Alice Collaboration: "ALICE experience with Geant4.", Presented at CHEP 2003, La Jolla, PSN MOMT011.

[16] R. Brun, A. Gheata, M. Gheata for the Alice Collaboration: "A geometrical modeller for HEP", Presented at CHEP 2003, La Jolla, PSN THMT001.

[17] http://root.cern.ch/root/vmc/VirtualMC.html