American Journal of Mathematical and Computer Modelling
2016; 1(1): 25-31
http://www.sciencepublishinggroup.com/j/ajmcm
doi: 10.11648/j.ajmcm.20160101.14

Investigation of the HDF5 C++ Library in Development of New Phase-Space I/O for Radiotherapy Simulation Using Monte Carlo Geant4 Code

Jaafar EL Bakkali[1, 3, *], Abderrahim Doudouh[1], Khalid Bouyakhlef[2], Laila Baddouh[3], Keltoum Dahmani[3], Hamid Mansouri[3]

[1]Nuclear Medicine Department, Military Hospital Mohammed V, Rabat, Morocco

[2]University Mohammed V, Souissi, Faculty of Medicine and Pharmacy, Rabat, Morocco

[3]Radiotherapy Department, Military Hospital Mohammed V, Rabat, Morocco

Email address:

bahmedj@gmail.com (J. E. Bakkali), abderrahim.doudouh@gmail.com (A. Doudouh), khalid.bouyakhlef@gmail.com (K. Bouyakhlef), laila.baddouh@gmail.com (L. Baddouh), kdahmani@gmail.com (K. Dahmani), hamid.mansouri@gmail.com (H. Mansouri)

[*]Corresponding author

To cite this article:

Jaafar EL Bakkali, Abderrahim Doudouh, Khalid Bouyakhlef, Laila Baddouh, Keltoum Dahmani, Hamid Mansouri. Investigation of the HDF5 C++ Library in Development of New Phase-Space I/O for Radiotherapy Simulation Using Monte Carlo Geant4 Code. *American Journal of Mathematical and Computer Modelling*. Vol. 1, No. 1, 2016, pp. 25-31. doi: 10.11648/j.ajmcm.20160101.14

Received: October 23, 2016; **Accepted**: November 3, 2016; **Published**: November 23, 2016

Abstract: This study aimed to develop a new phase-space tool for Geant4 code using the HDF5 C++ scientific data library. The tool can be easy incorporated into existing Geant4 applications and provides particle recycling and rotational splitting capabilities which can be useful for faster modeling symmetric systems such as medical linear accelerator. The validation of this phase-space I/O routines has been performed in a very basic geometry. Thus, taking into account a homogeneous water phantom, the depth dose curve of a 20 MeV electron beam hitting a small target made of tungsten has been calculated and compared to ones produced in the simulation without phase-space technique and simulation with IAEA phase-space I/O routines. This study shown an excellent agreement found between different calculated depth dose curves, allowing us to validate our new phase-space I/O routines. Moreover, the CPU time spent by simulation without variance reduction technique can be reduced by 27% when this method was applied which is the same factor obtained with IAEA phase-space I/O routines. The phase-space tool describing in this paper, have been implemented for Geant4 code by using HDF5 C++ data managing library, the associated classes are few and easy to incorporate into an existing Geant4 user code, and it is freely available on GitHub (https://github.com/EL-Bakkali-Jaafar/G4PhpH5/).

Keywords: Geant4, Phase-Space, Hdf5, C++, Variance Reduction Technique, Monte Carlo

1. Introduction

Monte Carlo Geant4 [1] [2] is a useful toolkit for simulating the passage of particles through matter and written in C++ language. This code has been developed primarily for high-energy physics discipline, meeting the efforts many workers from facilities such as CERN (Europe), KEK (Japan) and SLAC (US). It covers a broad range of physics fields including medical sciences, space sciences, high energy and accelerator physics. From a physics point of view, the Geant4 code calculates a physical evolution of each particle step-by-step by Monte-Carlo method. It can define materials and geometries, identify particulate matter and physics interactions, decide for generation of the first event. From a software point of view, the development and maintenance of Geant4 code are spread-head by the Geant4 collaboration. It is indeed very powerful, but also very complex. Consequently, a basic knowledge of C++ is strongly required to use the toolkit optimally. From our experience with Geant4 code, it seems that the Monte-Carlo simulation under this system code is relatively slow. Today significant efforts are being made by the Geant4 collaboration to increase simulation speed, but currently, it can spend too much time on a computer to accurately simulate problems with complex geometry. Thus, way the phase-space approach must be actively taken into account when simulating complex geometry with Geant4 code to deal with this issue.

In definition, the phase-space is a technique to reduce the computing time without affecting the computing accuracy. The phase-space approach is based on the idea of dividing the computation into two steps. During the first step, the data of all particles hitting the scoring plane are recorded in phase-space files from which they are recalled after. Each record in the phase-space file contains a collection of particle data such as the charge, energy, position, direction, and statistical weight. This approach reduces considerably the computing time because the same data can be used in different simulations. Regarding medical linear accelerator, the accelerator head simulation is very slow compared to the dose calculation, the use of phase-space technique can be considered as a suitable approach to improve simulation efficiently. Because the Monte-Carlo method involves simulation of a very high number of histories for achieving a smaller global statistical uncertainty, the file that contains the recorded phase-space data will have consequently a very large size and must be managed by one of the data storing library such as HDF5 library [3].

HDF5 is the new generation of HDF (Hierarchical Data Format) created by the HDF Group, and it is a free and open source general-purpose library and file format for storing and exchanging scientific data. As its name implies, all files with HDF5 based-format have a hierarchical structure which is very similar to the UNIX file system structure, and it is a self-describing file format supporting flexible types defined by the user. The HDF5 file is portable; it can be run on various types of platforms includes Linux, Mac OS X, and Windows. The HDF5 is a rich and large library contains more than 300 functions, which it goal is to provide flexible API supporting a wide range of operations on scientific data. It supports high-performance I/O for both serial and parallel environments. HDF5, it was designed for high volume or complex data. It saves binary data and its corresponding meta-data in the same file. The HDF5 C++ API used in the development of our phase-space I/O has C++ wrappers for the HDF5 C Library, and it is a part of the HDF5 source code.

The reading and writing of phase-space files is not part of the Geant4 kernel but is an application typically developed by users, an example of which can be found in the medical_linac example of Geant4. These files are useful for dividing the simulation into many distinct parts to improve simulation efficient. In this purpose, a modern phase-space I/O routines based on HDF5 file format has been developed specially for Geant4 applications, which can record particle data from five kinds of particle namely photon, electron, positron, neutron, and proton. The HDF5 has been chosen for building our phase-space writing/reading C++ classes because it has been widely used in scientific computing. Wenjie Wei discusses in its dissertation [4] the I/O performance, data structure, functionalities another aspect of three kinds of files format for a particular cosmology code, namely: HDF5, BinX and default binary. He concluded that HDF5 library gives the best I/O performance and has more meaningful features than two others.

The C++ API of HDF5 library is under development, but it is indeed very rich, and it was found useful and sufficient for developing our phase-space HDF5-based file format. We present in this paper a full description of the C++ classes of phase-space I/O routines and some tips needed to successfully incorporate this new phase-space tool into an existing Geant4 application. The validation of this new phase-space I/O consists of running three kinds of simulation namely: analog simulation or simulation without phase-space technique, simulation with IAEA phase-space I/O routines and simulation with our phase-space I/O routines. Each simulation intends to simulate depth dose in a homogeneous water phantom bombarded by a megavoltage photon beam which is created as a result of a 20 MeV electron beam hitting a tungsten target.

2. Related Works

The Geant4 toolkit provides many examples for beginners as well as for advanced users, covering a wide range of fields, and the radiotherapy is one of them. The example called medical_linac is one of the best Geant4-based tool for simulating teletherapy linear accelerator and dose calculation either in homogeneous or heterogeneous phantoms. This tool was fully described in this paper [5]. However, the implementation of its phase-space routines does not take into account the statical weight of each simulated track. Consequently, the users will encounter a hard biasing problem in their simulation when they try to introduce a bremsstrahlung splitting method for enhancing the statistical of photons since all simulated tracks with different statistical weight will be assumed as tracks propagating with unit weight. Another inconvenience of this tool is that it will generate a phase-space file in text format rather than binary format which makes the writing/reading process very slow and expensive time task. Moreover, this tool also lacks the ability to run multiple simulations on a distributed memory architecture which can save a lot of the computing time.

Another solution of phase-space I/O routines for Geant4 code has been developed since 2009 by Cortés-Giraldo [6], it is precisely a Geant4 interface to writing and reading IAEA formatted phase-space file thereby using three major components namely: IAEA routines [7][8] which are written in C language and was designed to cover phase-space files and event generator as well, Geant4 phase-space reader class named G4IAEAphspReader (.hh and. cc), and Geant4 phase-space writer class named G4IAEAphspWriter (.hh and. cc). This solution has been used in our earlier works [9][10].

The IAEA phase-space can record five types of particle namely: photon, electron, positron, neutron, and proton. This solution seems useful and sufficient for the most of the medical applications. However, it imposes the users to implement additional Geant4 User Classes as follows: for writing task, classes inherits from G4UserRunAction, G4UserEventAction, G4UserSteppingAction must be defined by the users, whereas for reading task, not additional Geant4 User Classes are required. The mechanism of recording particle data at Z-plane is as follows:

- The UserSteppingAction method of G4IAEAphspWriter singleton class point to the current step object.
- This method will record the Z-position of the two points of step namely: PreZ for PreStepPoint and PostZ for PostepPoint.
- After that, it will call the StoreIAEAParticle method (for storing particle data) until the value of PostZ is greater or equal to the defined Z-Plane and the PreZ value is small than the value of Z-plane.

3. Implementation of a New Phase-Space I/O Using HDF5 C++ Interface

3.1. The Phase-Space C++ Classes

Stand-alone classes inheriting from G4VSensitiveDetector and G4VPrimaryGenerator were developed for writing and reading phase-space in HDF5 format. Unlike Geant4 IAEA phase-space interface which requires implementation of G4User classes (stepping, tracking, event, and run), our solution has the advantage of not using the G4User classes. Therefore, our input phase-space routine seems to be easy to use than one offered by the Geant4 IAEA phase-space interface. The building of our I/O for phase-space in HDF5 format requires the implementation of the following C++ classes:

3.1.1. Phase-Space Writer C++ Classes

G4UserPhaseSpaceWriter

The writing task of phase-space file with HDF5 format is managed by this singleton class which has the following public methods:

- static G4UserH5PhaseSpaceWriter* GetInstance (): a static method to get the singleton reference to this class.
- void SET_PARAMETERS (G4String _FILE_NAME, G4double zstop, G4double X_PLANE_HALF_SIZE, G4double Y_PLANE_HALF_SIZE, G4LogicalVolume*& logicWorld): this method has six parameters namely: name of phase-space file, z position of recorded plane, half size of x dimension of a plane, half size of y dimension of the plane and a pointer to the logical volume of a world.

The users must be instanced this singleton class into their G4VUserDetectorConstruction abstract base class provided by Geant4 as user initialization class. The phase-space is written in an HDF5 file which has an extension of ".h5". It keeps all information about particles includes the number of simulated histories, particle energy, particle statistical weight, particle PDGE code, particle position, and particle momentum direction. In Addition, an ASCII text file is also generated at end of each Run which has an extension of "summary" and stores meaningful statistical data about a given simulation, it includes number of simulated histories, percent of active events, max-mean-min of all recorded particles, percent of each them, and CPU time spent by simulation. This singleton class has been created to be easy to use, and all sophisticated methods involved by phase-space written task are included in other complex class which is described in the next section.

H5PhaseSpaceWriter

The H5PhaseSpaceWriter class derived from G4VSensitiveDetector class has the following public methods:

1. Mandatory methods which are inherited from G4VSensitiveDetector class
   - void Initialize (G4HCofThisEvent*): a function which is called by the Geant4 kernel at begin of an event.
   - G4bool ProcessHits (G4Step*, G4TouchableHistory*): a function which is called by the Geant4 kernel at each step.
   - void EndOfEvent (G4HCofThisEvent*): a function which is called by the Geant4 kernel at the end of an event.
2. Methods needed for calculating statistical information about scored particles and summarize them.
   - void PHOTONS_ENERGY (G4double): get max-mean-min energies of the photon at scorer plane.
   - void ELECTRONS_ENERGY (G4double): get max-mean-min energies of an electron at scorer plane.
   - void POSITRONS_ENERGY (G4double): get max-mean-min energies of positron at scorer plane.
   - void PROTONS_ENERGY (G4double): get max-mean-min energies of the proton at scorer plane.
   - void NEUTRONS_ENERGY (G4double): get max-mean-min energies of the neutron at scorer plane.
   - void SUMMARY (): dump statistical information about current simulation to an ASCII text file.
3. Methods need to manage phase-space file
   - void SET_PHASE_SPACE_FILE_NAME (G4String FILE_NAME): set the name of phase-space file.
   - void FILL_DATA (G4Step* & aStep): score all needs physical quantities from the current step.
   - void WRITE_PHSP_FILE (G4int i): method to write out the phase-space file to the disk.

3.1.2. Phase-Space Reader C++ Classes

G4UserH5PhaseSpaceReader

Similar to others Geant4 primary particle generator user classes namely: G4ParticleGun and G4GeneralParticleSource, the G4H5UserPhaseSpaceReader is derived from G4VPrimaryGenerator virtual class. It has the following public methods:

1. Mandatory method which is inherited from G4VPrimaryGenerator class
   - Virtual void GeneratePrimaries (G4Event*): Geant4 does not provide any default behavior for generating a primary event which will be invoked at the

beginning of each event. This method is responsible for creating particles from phase-space file which is managed by two public methods that will be described in the next section.

2. Methods needed for managing a phase-space file
   - void SET_PARAMETERS (G4String PHASE_SPACE_NAME, bool CHANGE_VALUE_OF_Z_STOP, G4float VALUE_OF_NEW_Z_STOP, int PARTICLE_GENERATOR_FLAG, int SPLITTING_FACTOR): this method let the user define a set of phase-space parameters, including the phase-space file name, the new z-plane position if is determined, the particle generation mode (0: normal, 1: particle rotational splitting, 2: particle recycling) and the splitting factor if the particle generation is not set to its normal mode.
   - void INITIALIZE (): after defining a set of phase-space parameters by users, this method will dump all phase-space data into RAM memory.

3.2. The Geant4 Application Compilation File

We have changed the GNUmakefile to match the collection of the Geant4 user application that may use our phase-space I/O routines, thereby including the EXTRALIBS flag which will point to the HDF5 C++ library named libhdf5_cpp.a. Here's how the content of modified GNUmakefile might look:

```
name:= Geant4_application
G4TARGET:= $(name)
G4EXLIB:= true
EXTRALIBS += -L/opt/hdf5/lib -lhdf5_cpp
ifndef G4INSTALL
G4INSTALL =../../..
endif
.PHONY: all
all: lib bin
include $(G4INSTALL)/config/binmake.gmk
```

It should be noted that, instead of compiling the Geant4 application with h5c++ which is the native HDF5 C++ compiler, we keep the default C++ compiler, and we add the external library to wrap the functionality of this HDF5 C++ compiler as described above.

4. Incorporation of Phase-Space I/O Routines in an Existing Geant4 Application

The phase-space I/O classes have been designed and implemented to be few and easy to incorporate into an existing Geant4 application. The users must prepare their Geant4 application to read or write phase-space HDF5-based format. Unlike IAEA phase-space routines, our phase-space methods don't impose users to create new user actions classes. Therefore the required changes are made too easier. For writing task, the G4UserH5PhaseSpaceWriter and H5PhaseSpaceWriter C++ classes (source and headers files) must be copied to the directory of the Geant4 user application. Naturally, header files must be copied into the include subdirectory, whereas source files must be copied into src subdirectory. Here, before compilation processing by the user, he must ensure that the two following things are presented:

- The HDF5 C++ library.
- The modified GNUmakefile file as it was described in early paragraphs.

Once, the above steps are successfully performed, the user must call the phase-space writing routine into its application, thereby including the G4UserH5PhaseSpaceWriter.hh header file into it user volume construction class inheriting from G4VUserVolumeConstruction user initialization class, and adding a small code before return statement (which is followed by the name of the physical volume of world volume) as follows:

```
G4UserH5PhaseSpaceWriter::GetInstance ()-
>SET_PARAMETERS (FILE_NAME, Z_STOP,
PLANE_HALF_X, PLANE_HALF_Y,
LOGICAL_WORLD);
```

The generated phase-space file which has as an extension of ".h5" can be directly viewed by a very useful command-line tool provided by HDF5 Group and called h5dump enabling the user to examine the contents of an HDF5 file and optionally dump those contents to an ASCII text file. For the phase-space reading task, the users must put the G4UserH5PhaseSpaceReader.cc header file into src subdirectory and G4UserH5PhaseSpaceReader.hh into including subdirectory of their Geant4 user application. Like writing task the user must ensure that the HDF5 C++ library is presented and the compilation file is modified. Once these steps are made, the user must include the G4UserH5PhaseSpaceReader.hh header file into it primary generator user class which is derived from G4VUserPrimaryGenerator base class, and put the following codes in the initializing block of this class:

```
theG4UserH5PhaseSpaceReader = new
G4UserH5PhaseSpaceReaderr ();
theG4UserH5PhaseSpaceReader->SET_PARAMETERS
("PHASE_SPACE.h5", // name of phase-space file. false,
// if true the user must indicate the new Z_STOP
parameter. 0, // new Z_STOP parameter. 0, // particle
generation mode. 0); // splitting parameter.
theG4UserH5PhaseSpaceReader->INITIALIZE ();
```

For generation of a new particle from phase-space file, the user must add the following code into the GeneratePrimaries mandatory method of it primary generator user class:

```
theG4UserH5PhaseSpaceReader->GeneratePrimaryVertex
(anEvent);
```

Regarding particle recycling and rotational splitting techniques, in the aim to ensure statistical correlations between particles produced by the same original history, all copies of the same track (parent track) are produced at the same event and each daughter track is monitored with a statistical weight that is equal to the parent statistical weight divided by the splitting number. With this manner, we can correctly track all original histories read from phase space file that are shooted in a simulation, and the normalization to dose per primary particle can be successfully performed without altering the physics of the simulated setup.

5. Comparison Between Calculated Beam Data for Three Kinds of Simulation

In order to validate the new phase-space I/O, a very basic experimental setup has been realized. It consists of simulating depth dose of a 20 MeV electron beam hitting a tungsten target (box of 1x1x0.1 cm3 made of tungsten) in a homogeneous water phantom. The 30x30x30 phantom was subdivided into 1x1x30 slabs, which were further divided into 1x1x1 cubes. Three kinds of simulation have been made, the first one is an analogue simulation, the second one is made using phase-space I/O routines based on HDF5 file format, and was split into two stages. In the first stage, we simulate 1/4 of the number of histories considered in the case of analogue simulation. Thus, just 10 millions of histories have been simulated to produce a phase-space file. This latter is used in the second stage as particles generator source for calculating depth dose in a homogeneous water phantom, and the number of simulated histories was set to 40 million equal to the value considered for analogue simulation. Consequently, the phase-space file was reused four times for calculating dose along z-axis. The third simulation with IAEA phase-space I/O routines have been made in the same conditions as those considered for the second simulation where our phase-space I/O routines are examined. Table 1 demonstrates a comparison between simulation phase-space data for the two last kinds of simulation.

Table 1. *Comparison between IAEA phase-space and HDF5 phase-space data.*

	IAEA phase-space	**HDF5 phase-space**
Number of histories	10^8	10^8
Number of particles in Phsp	20383494	24590103
Percent of photons in Phsp	79%	77%
Percent of electrons in Phsp	22%	22%
Percent of positrons in Phsp	0,34%	0,38%
CPU time	1327 s	1192 s

From the above table, it seems that: A) for the same number of histories, the CPU time spent by the generation of an HDF5 phase-space file is about 13% less than one spent by IAEA phase-space file. B) The number of particles recorded in HDF5 phase-space file is 21% more than one provided by IAEA phase-space file. C) the percent of each particle in phase-space is closely the same for two kinds of phase-space.

It is difficult to give an accurate interpretation of the obtained results, but we can say from these results that the efficiency of our phase-space I/O routines is comparable and near closely to the one given by IAEA phase-space I/O routines. The difference between the number of scored particle at phase-space plane for two kinds of phase-space I/O is caused by the fact that each phase-space I/O uses different method to score particles at Z-plane, we remark that the minimal energy recorded by IAEA phase-space I/O routines is ten times much smaller than one obtained with our phase-space I/O routines. More precisely, our phase-space writer routine record all particles that reach an additional volume that is not part of geometry of simulated system which is a very thin vacuum box and placed at position equal to the value of Z-Plane plus the value of half Z-dimension of the box. Whereas in the case of IAEA writer routine, not additional volume is required, instead the particle that reaches the Z-plane are recorded as was described in "related work" section. It should also be emphasized that the two kinds of phase-space files share approximately the same disk size for a fixed number of particles recorded at defined Z-Plane.

In Table 2, we present a comparison between CPU time spent by dose calculation program for three cases examined in this study; we have run a full simulation with a reasonable number of events for testing the new phase-space I/O. It should be noted, that all calculation have been performed in sequential mode, in a personal computer Intel (R) Core (TM) i3 2.20GHz under Ubuntu 12 operating system.

Table 2. *Comparison between CPU time spent by three kinds of simulation considered in this study.*

	Simulation analogue	**Simulation with IAEA phase-space I/O**	**Simulation with HDF5 phase-space I/O**
Number of histories	$4\ 10^7$	$4\ 10^7$	$4\ 10^7$
CPU time	10738 s	6590 s	6585 s

To compare the effective improvement obtained when the phase-space method is applied, we add the CPU time spent by phase-space generation program from one spent by the dose calculation program, and we compare it with CPU time spent by analogue simulation program. The results showed an improvement in CPU time of 26% in the case of the IAEA phase-space method and an improvement in CPU time of 27% in case of HDF5 phase-space method.

A comparison of depth dose data in three cases: analogue simulation and simulation with IAEA phase-space and HDF5 phase-space is presented in figure 1.

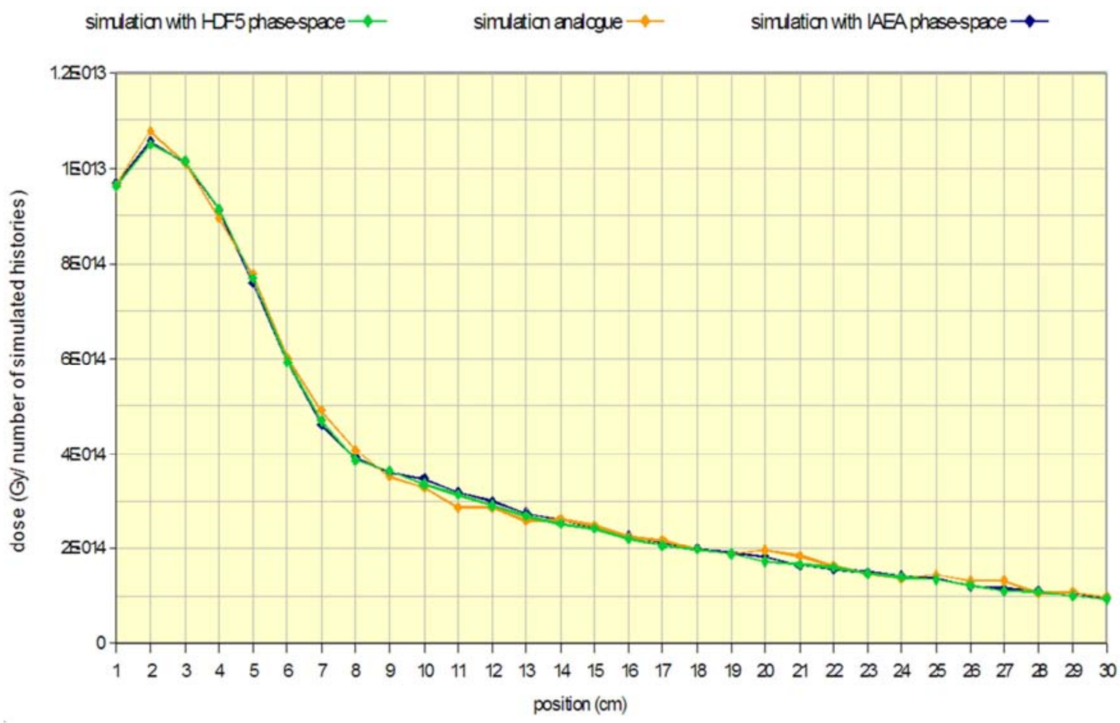

Figure 1. *Calculated depth dose of a 20 MeV electron beam hitting a tungsten target.*

The visual inspection of figure 1, demonstrates in the first time that our phase-space I/O not alter the physics in any way, and that a good agreement is obtained between the two calculated depth dose data, since the two curves are closely the same and there are perfectly coincidence between them. Now we will to see statistical uncertainties related to the calculated depth dose data. Figure 2 gives such response.

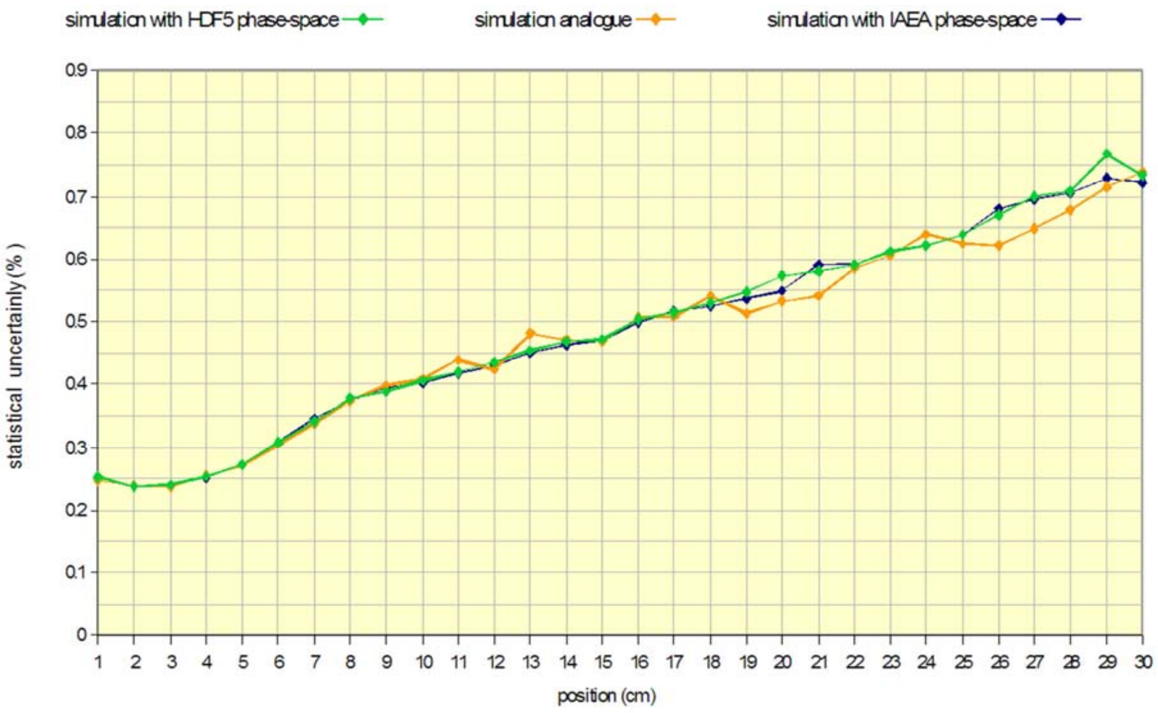

Figure 2. *Statistical uncertainties of calculated depth dose data.*

From figure 2, it seems that the associated uncertainties are quasi-confused for most of the points. We remark that the statistical uncertainty increases nonlinearly with depth because the beam data is not homogeneous, it has multiple energies and multiple particle types as well.

6. Conclusion

The phase-space I/O capabilities are describing in this paper, have been implemented for Geant4 code by using HDF5 C++ data managing library. The code source of this tool has been hosted on GitHub, and it can be downloaded freely from:

https://github.com/EL-Bakkali-Jaafar/G4PhpH5

The important feature of our phase-space I/O which paves the way for its easy application is the fact that the associated classes are few and easy to incorporate into existing Geant4 user code. It has been validated for a very basic experimental setup. Our result shows that this technique can save about 27,56% of CPU time spent by simulation. However, this method is problem dependent, and we believe that the usefulness of these I/O routines becomes critical in a complex system simulation which takes too much CPU time such as Linac, where the modeling of it head is relatively long than dose calculation. With phase-space method, the Linac simulation can be made fast enough, and it will be the subject of our next study.

References

[1] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce et al., "Geant4 - A Simulation Toolkit," Nuclear Instruments and Methods A 506, 2003, pp. 250-303.

[2] Allison J et al., "Geant4 Developments and Applications," IEEE Transactions on Nuclear Science 53, 2006, pp. 270–278.

[3] The HDF5 Group, HDF5 Dara Format. [Online] Available: http://www.hdfgroup.org/HDF5/release/obtainsrc.html.

[4] W. Wenjie, "Comparison of Portable Binary Data Formats within a Cosmological Simulation," M. Sc. in High Performance Computing, The University of Edinburgh, 2005.

[5] B. Caccia, C. Andenna, and G. A. P Cirrone, "MedLinac2: a GEANT4 based software package for radiotherapy," Annali dell'Istituto superiore di sanita 46, 2010, pp. 173–177.

[6] M. A Cortés-Giraldo, J. M Quesada Molina, M. I Gallardo, R. Capote, "Geant4 Interface to Work with IAEA Phase-Space Files", 2009.

[7] M. A Cortés-Giraldo, J. M Quesada Molina, M. I Gallardo, R. Capote, "An implementation to read and write IAEA phase-space files in GEANT4-based simulations," Int. J. Radiat. Biol. 88, 2012, pp.200-208.

[8] R. Capote R and I. Kawrakow, "Read/write routines implementing the IAEA phsp format, version of December 2009". [Online] Available: http://www-nds.iaea.org/phsp/software/iaea phsp Dec2009.zip#phsp rw

[9] J. EL Bakkali, T. EL Bardouni, S. Safavi, M. Mohammed, S. Mroan, "Behaviors of the percentage depth dose curves along the beam axis of a phantom filled with different clinical PTO Objects, a Monte Carlo Geant4 study," Radiation Physics and Chemistry 125, 2016, pp.199-204.

[10] J. EL Bakkali, T. EL Bardouni, "Validation of Monte Carlo Geant4 code for a 6 MV Varian linac," Journal of King Saud University–Science, 2016, in press.