004 Computer science and technology. Computing. Data processing

ANALYSIS OF GEOMETRY DESCRIPTIONS FOR OPTIMISED SIMULATION PERFORMANCE

Alexander Sharmazanashvili[1][1], Besik Kekelia[2]
[1]Georgian Technical University, info@gtu.ge

Abstract

The simulation is being used for the production of the artificial event for the physics analyses. Several generators simulate the particle collisions and create the hits. The ATLAS simulation generates about 5-7 billion events per year and it requires about 77Mln computing hours [2]. Therefore, one of the important parameters of simulation is performance. The ATLAS uses Geant4 as a simulation infrastructure. The Geant4 uses geometry descriptions as an input for the modelling of the propagation of the particles through the material. On the other hand, there are several methods for the creation of geometry descriptions in Geant4. This paper gives qualitative and quantitative results of the investigation of the influence of the methods of geometry description in Geant4 on the performance of the simulation. The conclusions bring a good understanding together with the quantity measurements, how geometry has to be described for the particular cases in order to ensure the minimum CPU usage and the RAM consumption.

Keywords: Performance, Atlas experiment, Geant4, Simulation, Cern

1. Introduction

The ATLAS detector at the LHC collects data from the proton-proton collisions every 25 ns (40Mhz), providing 23 collisions per bunch crossing [1]. However, ATLAT physics analyses need to study more wide range of physics processes and scenarios. For that reason, simulation is implemented, which enables the carrier of artificial events from the Monte-Carlo generators and creates the output, identical to the real detector.

The ATLAS use Geant4 as a simulation infrastructure. The Geant4 simulation foresees the execution of five consecutive steps [2] (fig.1).

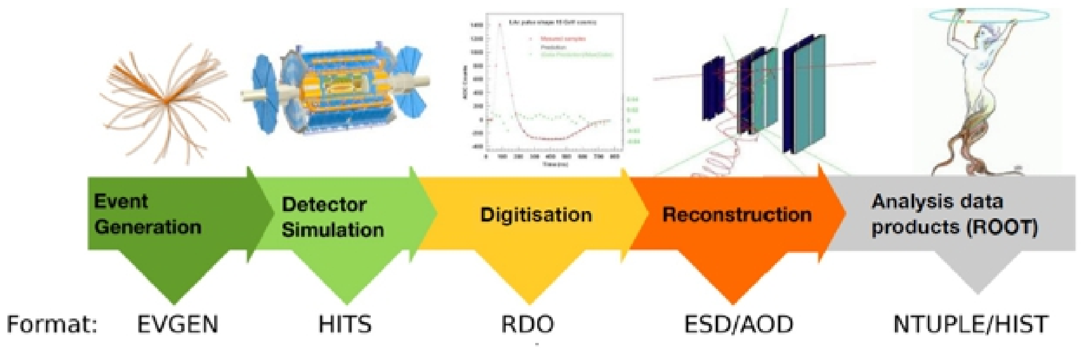


Fig. 1. Consecutive steps of Geant4 simulation

[1] Corresponding author: lasha.sharmazanashvili@cern.ch

The 2nd step "Detector Simulation", generates hits and uses geometry descriptions of the detectors as input data. For geometry description in the ATLAS simulation, three main formats are using - Gdml, GeoModel, and XML. The Gdml is the neutral format of Geant4. The GeoModel was developed for the description of large and complex detector geometries with minimal memory consumption [4]. Descriptions in the GeoModel are splitting into semantical and parametrical parts and use different containers for them. The semantical part is presented in the form of C++ like templates for each geometry description. The parametrical part is in the Oracle tables. The GeoModel description is forming on-line by the ATHENA framework before starting the Geant4 simulation. The XML mainly developed for the description of the passive materials of the Muon system [5]. It uses XML file format and the so-called AGDD - ATLAS Generic Detector Description language as a container of the geometry descriptions.

detector	number of volumes	comments
Pixel	6000	
SCT	40500	
TRT	300000	parameterized
LAr	142500	parameterized
Tile	80500	parameterized
Muon Chambers	451000	parameterized
Toroids	1000	

Table.1 Number of simulated volumes

All three resources - Gdml, GeoModel, and XML are using more or less the same programming methods for descriptions. Shapes are defining as a separate entity, solids [3]. Simple solids are described by the CGS - Constructive Solid Geometry parameterized primitives, like Cylinder, Shell, Boxes, Tubes, etc. More complex solids are defined by the BREP - Boundary Representations, like second-order surfaces or B-spline surfaces. Such kind of surfaces doesn't exist in the ATLAS detector. Therefore, GeoModel and XML don't support the BREP methods. Other methods of the description of the complex solids are the Boolean operations - Union, Intersection, and Subtraction. The Boolean operations are executing on the CGS solids or solids which are the product of previous Boolean operations. Also, complex solids can be built by the component's relative transformations - Move, Rotation, and Scale.

Therefore, in principle, several methods could be implemented for the description of the same geometry. On the other hand, each method causes a different usage of CPU time and memory, because activates the different library functions of Geant4. For the one, particular geometry this difference might be negligible, but for a large number of geometries, it will be important. The as-built geometry of the ATLAS detector consists of ~45Mln CGS primitives. The Geant4 simulation use simplified geometry descriptions and the total number of primitives is less, about on factor 10. Paper [6] brings the number of volumes, which is equivalent to solids, in the GeoModel (Table 1). The overall number of solids is 1'021'500. Thus, investigation of the influence of the geometry description methods on the performance of the simulation is the actual task.

2. Test examples

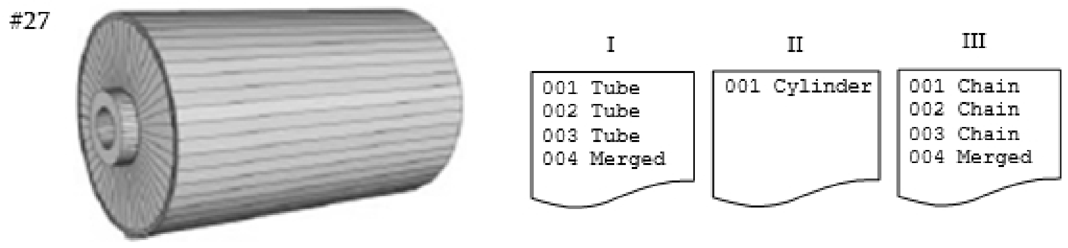
Typical examples of the Geant4 geometry descriptions is necessary for the analyses. Each typical example is a combination of geometry and description methods. As it was mentioned above for each geometry, several description methods are existing in the separate codes. Finally, all methods from the typical geometries bring the full set of test examples for the analyses. The full set then has to be categorized in the classes according to geometry, description methods, and topology of the codes.

The first step for the formation of the test examples is the creation of typical geometry representatives of the ATLAS detector, representing the unique geometry features of the detectors. The geometry of the ATLAS detector consists of relatively simple shapes and doesn't contain splines. Thus, the detector geometry can be mainly described through the CGS primitives.

- The first class of geometries is Cylindrical objects, which can be formed by one or several cylinders with implementation of the several transformations and Boolean operations. After the analyses of the ATLAS detector geometry, 11 primitives were separated.
- The second class is the Prismatic objects, formed by one or several box methods with implementations of the several transformations and Boolean operations. After the analyses of the ATLAS detector geometry, 31 primitives were separated.
- The third class is the Combined objects, formed by the combination of the Cylindrical and Prismatic objects. After the analyses of the ATLAS detector geometry, 24 primitives were separated.

The second step of the test examples formation is the assignment to the geometry objects the possible methods of their description. For the Cylindrical class, the standard methods are implemented: Tube, Cylinder, and Chain; for the Prismatic class are the following methods - Cube, Pyramid, Arbitrary Polygon, Symmetric Polygon, Double Symmetric Polygon. In both, Cylindrical and Prismatic classes, complex objects are described by the Combined, Merged, Subtraction, Union, Intersection, Hybridized methods. As a result, 14 methods dropped into the focus of the simulation performance investigation. For the third, Combined class, all 14 methods are implemented.

The formation of the descriptions by varying the 14 methods, brings for 11 Cylindrical objects 126 descriptions, for the 31 Prismatic objects, 1'883 descriptions, and for the 24 Combined primitives, 704 descriptions. However, most of them have codes with similar topologies. Therefore, the received set of the descriptions has to be rated.



```
001 Tube
002 Tube
003 Tube
004 Merged
```

```
001 Cylinder
```

```
001 Chain
002 Chain
003 Chain
004 Merged
```

Fig. 2. Description methods for the *Cylindrical* primitive

Consideration of the 126 descriptions from the Cylindrical class shows that in the majority of the cases, the codes have a homogeneous topology and use one type of method. For instance, #27 primitive of the Cylindrical class has 3 descriptions, using Tube method for the 1st, Cylinder method for the 2nd, and Chain method for the 3rd description (fig.2). Therefore, it was decided to calculate the total amount of cylinders in the detector geometry and make the direct comparison of methods - Tube-vs-Cylinder-vs-Chain. The same consideration and

conclusion was done for the Prismatic class of objects.

The 704 descriptions from the Combined class was rated according to 3 criteria:

1. Exclusion of the theoretical methods
2. Exclusion of the irrational methods
3. Exclusion of the methods with the similar topologies

For the 1st criteria, complex objects can be described alternatively, by the several of 93 auxiliary CGS primitives and corresponding Boolean operations, either by one 94 Arbitrary_Polygon method. Therefore, the number of methods were excluded.

For the 2nd criteria, it was to pay attention to the number of transactions in the alternative ways of the description of the one primitive. For instance, Primitive #33 has 3 alternatives: 1/using the Cube method and associated 1 transaction 2/using the Arbitrary_Polygon method and associated 2 transactions 3/using the Symmetric method and associated 3 transactions. It can be concluded in advance that methods with more transactions will cause worse performance. Therefore, the methods with the minimum number of transactions remained for further consideration.

For the 3rd criteria, it was analysed the cases with identical topologies. For instance,

#22
001 Arbitrary/Cube/Pyramid/Symmetric
002 Tube/Cylinder/Chain
003 Composition

#34
001 Arbitrary/Cube/Pyramid/Symmetric
002 Tube/Cylinder/Chain
003 Composition

#53
001 Tube/Cylinder/Chain
002 Arbitrary/Cube/Pyramid/Symmetric
003 Composition

Primitives #22, #34, and #53 has each, the 12 alternative methods of description of 2 solids. 1st solid in #28 can be described either Arbitrary, or Cube, or, Pyramid, or Symmetric methods; 2nd solid either Tube, or Cylinder, or Chain methods.
The same is true for the #34 and #53 primitives. As a result, they were excluded from consideration.

Finally, from the 24 Combined primitives and 704 methods, 18 primitives and 22 methods were selected for further consideration.

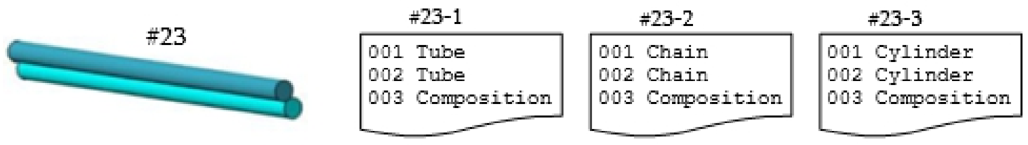
3. Test analyses

The Quantitative measurements of performance are possible through the estimation of RAM consumption and CPU usage. Therefore, for the simulation runs the used RAM size and the CPU time were taken from the log files.

Initially the test infrastructure was investigated. Several tests run of the default example showed unchanged RAM consumption and different CPU times. Ten consecutive runs on the CERN Linux server lxplus703 brings 1.8% deviation of CPU time hereinafter called noise; on the lxplus750 server 3.5% and on lxplus604 server 3.9%. This is connected with the overall activity on the servers. Therefore, it was decided to make test runs before each measurement session and identify the server with minimal noise. It will minimize but not exclude the uncertainty of CPU time measurements and make it possible to receive valid qualitative conclusions.

Investigations started from the *Cylindrical* class of objects. As mentioned above, it was decided to make a direct comparison of the methods Tube-vs-Cylinder-vs-Chain for the homogeneous and non-homogeneous topologies of the codes. The total number of *Cylindrical* objects in the Geant4 description of the ATLAS detector was counted:

6'776 tubes + 169 cylinders + 115 chains = 7'060 cylindrical objects



```
001 Tube
002 Tube
003 Composition
```

```
001 Chain
002 Chain
003 Composition
```

```
001 Cylinder
002 Cylinder
003 Composition
```

Fig. 3. Comparison of methods on homogeneous topology

For the homogeneous example primitive #23 was chosen (fig.3). Three codes: #23-1, #23-2, and #23-3 were built with the same structure and different methods of Tube, Chain, and Cylinder. Results are presented in table 2.

Table 3 *lxplus723.cern.ch* Noise=0.88%

Serie	Measurm.	Tube	Chain	Cylinder	Diff			%		
					Tube	Chain	Cylinder	Tube	Chain	Cylinder
1	CPU/ms	66810	67280	66510	300	770	0	0.45	1.15	0
	RAM/kb	638886	638889	638887	0	3	1	0	0.0005	0.0002
1000	CPU/ms	66740	66770	67800	0	30	1060	0	0.045	1.563
	RAM/kb	641014	641048	641016	0	34	2	0	0.005	0.0003
2000	CPU/ms	67030	69300	68170	0	2270	1140	0	3.2	1.67
	RAM/kb	643161	643222	643163	0	61	2	0	0.009	0.0003
3000	CPU/ms	67230	66570	67910	660	0	1340	0.98	0	1.97
	RAM/kb	645408	645498	645409	0	90	1	0	0.0.014	0.0002
4000	CPU/ms	67780	68010	67790	380	0	1420	0.57	0	2.09
	RAM/kb	647445	647572	647447	0	127	2	0	0.02	0.003
5000	CPU/ms	66720	67000	68910	0	280	2190	0	0.42	3.18
	RAM/kb	649935	650109	649936	0	174	1	0	0.03	0.0002
6000	CPU/ms	66670	67490	69530	0	820	2860	0	1.22	4.11
	RAM/kb	651924	652126	651925	0	202	1	0	0.03	0.0002

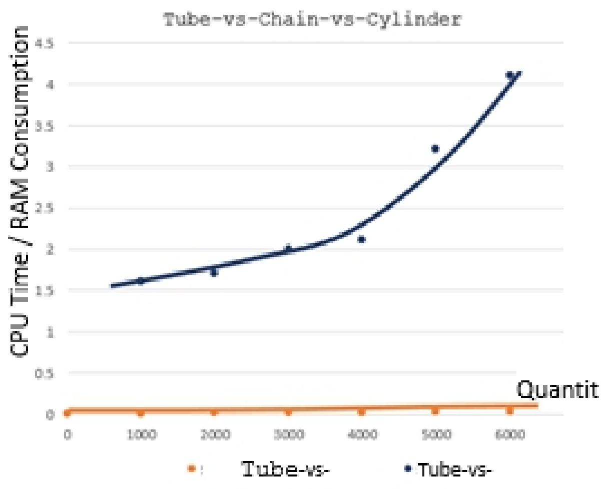
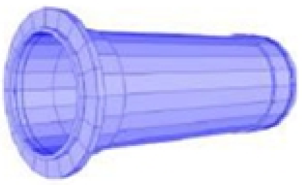


Fig. 4. Tube-vs-Chain-vs-Cylinder difference dependence on quantity

Best results deliver the Tube method in both, RAM consumption and CPU time. The consumption of RAM for the Tube, Chain, and Cylinder methods is almost the same for all quantities. However, Cylinder method uses more CPU time than Tube method (fig.4). This difference comes out from the noise for relatively small quantities <800 and exponentially

growing with higher quantities. For the maximum quantity - 7'000 139 Tube-vs-Cylinder cause ~5% difference in CPU time of the overall simulation session. For the non-homogeneous example, the primitive #11 was investigated (fig.5).



#11-1

```
001 Cylinder
```

#11-2

```
001 Tube
002 Tube
003 Tube
004 Tube
005 Tube
```

Fig. 5. Comparison of methods for non-homogeneous topology

Table 3 *lxplus723.cern.ch* Noise=0.88%

Serie	Measurm.	Tube	Cylinder	Diff		%	
				Tube	Cylinder	Tube	Cylinder
1	CPU/ms	66740	66380	360	0	0.54	0
	RAM/kb	638889	638892	0	3	0	0.0005
1000	CPU/ms	67870	66340	1530	0	2.25	0
	RAM/kb	642302	640355	1947	0	0.3	0
2000	CPU/ms	66970	66880	90	0	0.14	0
	RAM/kb	645859	641794	4065	2	0.63	0
3000	CPU/ms	66930	66630	300	0	0.45	0
	RAM/kb	649363	643350	6013	0	0.93	0
4000	CPU/ms	67710	67260	450	0	0.66	0
	RAM/kb	652618	644724	7894	0	1.21	0
5000	CPU/ms	66960	69200	0	2240	0	3.24
	RAM/kb	656327	646486	9841	0	1.5	0
6000	CPU/ms	68250	70440	0	2190	0	3.11
	RAM/kb	660085	647832	12253	0	1.9	0
7000	CPU/ms	67800	70900	0	3100	0	4.37
	RAM/kb	663342	649306	14036	0	2.12	0

The primitive #11 can be described by one Cylinder method (Code #11-1) or by 5 consecutive Tube methods (Code #11-2). Test measurements were done for different quantities: 1-300-600-900-1'200. Results are presented in Table 3.

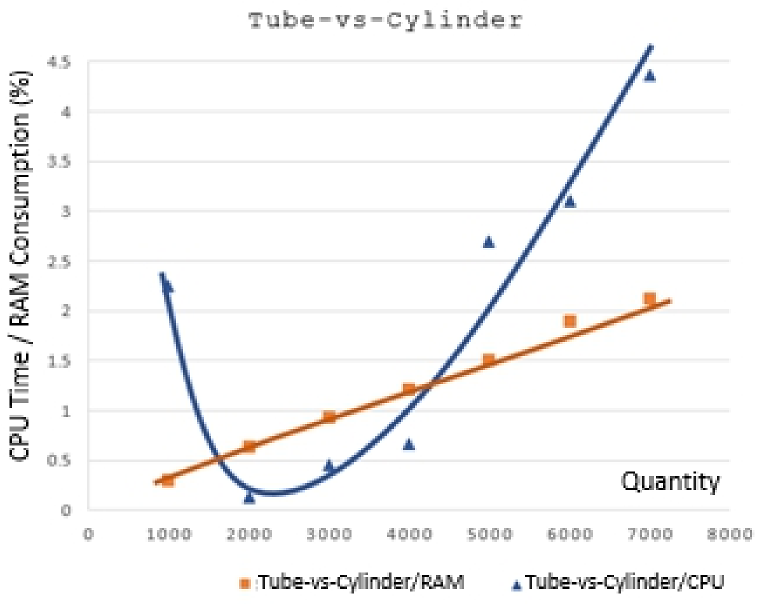
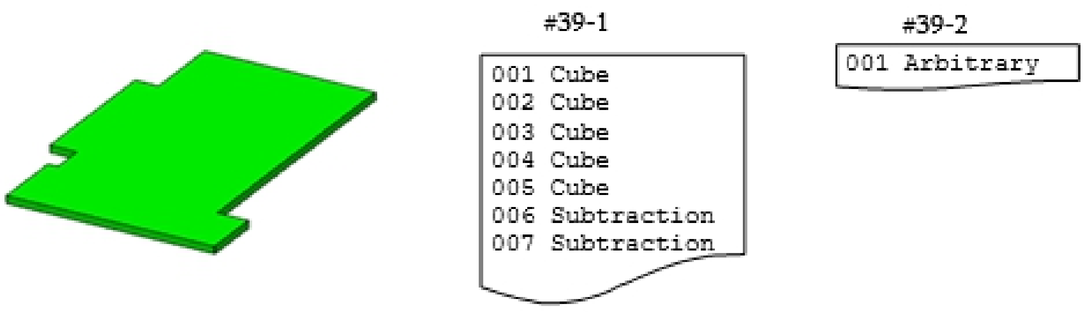


Fig. 6. Comparison of methods for non-homogeneous topology

For all quantities, #11-1 with Cylinder method has better consumption of RAM than the #11-2 with the 5 Tubes. However, for relatively small quantities <1'500, it requires more CPU time, and for the smaller quantities (100-200) can cause a substantial difference in the performance of simulation session ~3% (fig.6). For relatively bigger quantities #11-1 brings better performance and for quantity 7'000 can cause up to 2% difference of RAM consumption and up to 4.5% difference of CPU time.

For the *Prismatic* class of objects, 4 methods were investigated - Cube, Pyramid, Arbitrary_Polygon and Symmetric_Polygon. On the first step, the CGS methods were compared without and with the Boolean operations. The number of boxes was taken from the counted number of the Cubes and Pyramids in the Geant4 description of the ATLAS detector:

13'814 Cubes + 26'310 Pyramids = 40'124 Boxes



```
001 Cube
002 Cube
003 Cube
004 Cube
005 Cube
006 Subtraction
007 Subtraction
```

```
001 Arbitrary
```

Fig. 7. Non-homogeneous codes analyses of *Prismatic* objects

The comparison of CGS methods brought no difference between the CGS methods in RAM consumption of homogeneous code topologies for both cases, with and without Boolean operations. The CPU time difference is always within the noise. The measurements carried out

for the quantities:3k-6k-9k-12k-18k and 40k.

On the next step of analyses, CGS methods compared with the polygon method - Cube-vs-Arbitrary. For that purpose, primitive #39 was chosen. Two codes of description were built - #39-1 with the implementation of a sequence of 5 cubes and 4 Boolean operations, and #39-2 with one Arbitrary method (fig.7). Test measurements were done for the same quantities. Results are presented in Table 4.

Table 4 *lxplus723.cern.ch*

Serie	Measurn.	Cube	Pyramid	arbitrary	Cube vs arbitrary		Cube vs arbitrary	
3000	CPU/ms	69520	68910	67380	2140	0	3.08	0
	RAM/kb	655615	655616	643414	12201	0	1.86	0
6000	CPU/ms	68960	69650	66540	2420	0	3.51	0
	RAM/kb	672590	672591	647846	24744	0	3.68	0
9000	CPU/ms	69870	70350	66880	2990	0	4.28	0
	RAM/kb	689757	689758	652628	37129	0	5.38	0
12000	CPU/ms	69820	70950	67790	2030	0	2.91	0
	RAM/kb	709003	709004	656649	52354	0	7.38	0
15000	CPU/ms	71030	71180	68470	2560	0	3.60	0
	RAM/kb	725126	725127	660945	64181	0	8.85	0
18000	CPU/ms	72840	74730	69600	3240	0	4.45	0
	RAM/kb	743084	743085	666264	76820	0	10.34	0
40000	CPU/ms	77330	78340	71050	6280	0	8.12	0
	RAM/kb	869102	869103	699145	169957	0	19.56	0

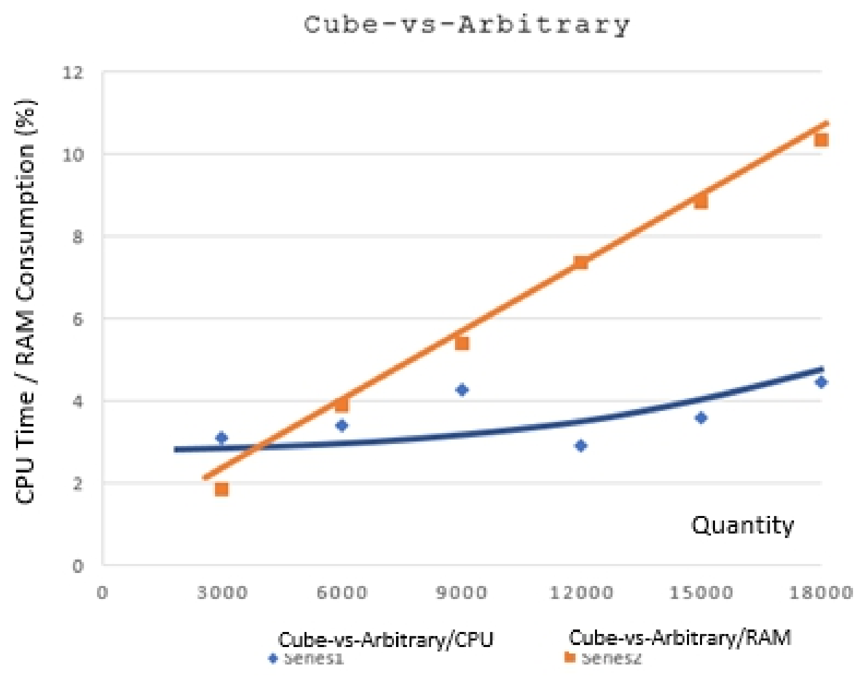
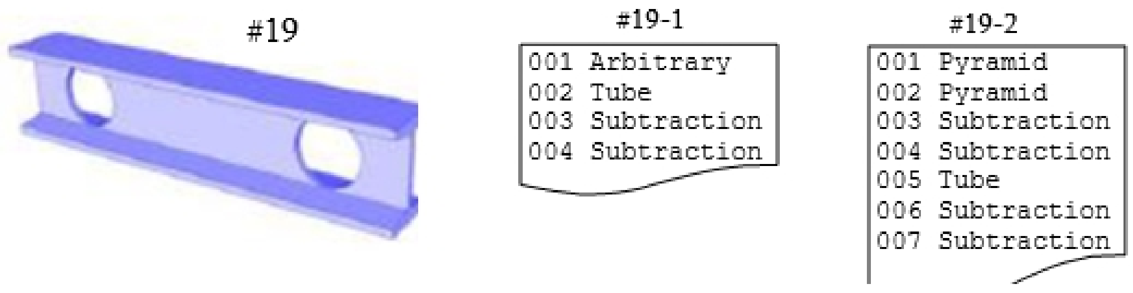


Fig. 8. Comparison of methods for non-homogeneous topology

The Arbitrary method brings always better performance than the CGS methods for all quantities. This difference is most pronounced for RAM consumption. For big quantities, it can cause a >10% difference in the overall simulation session (fig.8). For maximum quantity, the difference is up to 20%. The difference in CPU time is in the range of noise and slightly grow up for the bigger quantities. For maximum quantity difference in CPU time can reach 8%.

Different results have been received for the *Combined* class of objects. The CGS-vs-Arbitrary measurements bring opposite results. The consumption of RAM for the Arbitrary method is less than for the CGS for the small quantities. Below is given results of comparative analyses, done for the *Combined* Primitive #19 (fig.9). Two alternative codes were compared. Code #19-1 using Arbitrary method for the description of the *prismatic* part of #19 primitive; auxiliary cylinder built by the Tube method and 2 Boolean Subtraction for the formation of holes. Code #19-2 describes the *prismatic* part by 2 CGS objects and 2 Boolean Subtractions. The rest of the part, for holes formation, is the same as code #19-1, with the implementation of the Tube method and 2 Boolean Subtractions. The measurements were done for the various number of primitives in the description - 1, 5, 10, 20, 25, 30.



#19-1

```
001 Arbitrary
002 Tube
003 Subtraction
004 Subtraction
```

#19-2

```
001 Pyramid
002 Pyramid
003 Subtraction
004 Subtraction
005 Tube
006 Subtraction
007 Subtraction
```

Fig. 9. CGS-vs-Arbitrary analyses in *Combined* object

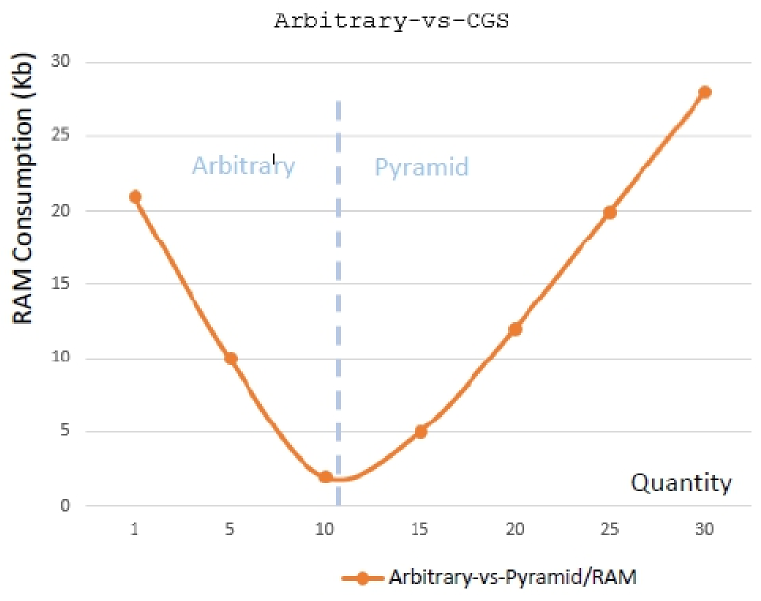


Fig. 10. Comparison of methods in Combined

Therefore, for the small quantities (<10) the CGS methods have less consumption of RAM than the Arbitrary method. For the >10 quantities, Arbitrary method has better characteristics than the CGS (fig.10).

However, for considered quantities, the contribution of this difference in the total consumption of RAM of the simulation is below 1%. Only starting from 6’144 quantity, the difference caused by the methods, contribute above the 1% of the overall RAM consumption of the simulation. The Arbitrary method shows better CPU time. The difference in CPU time slightly rises linearly with the quantity and for big quantities can reach 4% of the overall simulation session.

4. Conclusions

1. Geometry description methods have an important influence on the performance of the simulation.
2. For *Cylindrical* objects, the best performance is given by the Tube method, mostly pronounced for CPU time. In some rare cases and for small quantities, the Cylinder method gives better results.
3. For the *Prismatic* objects, Polygon methods ensure dominantly better performance than the CGS methods, mostly pronounced for RAM consumption. Differences up to 15% can occur often.
4. In the majority of cases of the *Combined* objects, the above-described conclusions are true.

References

[1] The ATLAS Collaboration "The ATLAS Simulation Infrastructure"/The European 209 Physics Journal C (2010) 70:823-874, DOI 10.1140/epjc/s10052-010-1429-9 210;

[2] Flavia de Almeida Dias "The New ATLAS Fast Calorimeter Simulation"/ICHEP2016, 5 211 August, 2016 212;

[3] Geant4 Collaboration "Geant4 A Simulation Toolkit"/Book for Application Developers, 213 Release 10.4, Rev1.0:Dec 8th, 2017 214;

[4] J.Boudreau, V.Tsulaia "The GeoModel Toolkit For Detector Description"/CHEP 215 proceedings 2004, Interlaken, Switzerland, pp.353-356 216;

[5] Laurant Chevalier et al. "An XML Generic Detector Description System and Geometry 217 Editor for the ATLAS Detector at the LHC"/J.Physics:Conf. Ser. 396-022009, 2012 218;

[6] D.Costanzo et al. "ATLAS Detector Simulation: Status and Outlook"/2005.

Article received: 2021-04-14