Supplement - performance

Casper Rutjes[1], David Sarria[2], Alexander B. Skeltved[3], Alejandro Luque[4], Gabriel Diniz[5,6], Nikolai Østgaard[3], and Ute Ebert[1,7]

[1]Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
[2]Astroparticules et Cosmologie, University Paris VII Diderot, CNRS/IN2P3, France
[3]Department of Physics and Technology, University of Bergen, 5020 Bergen, Norway
[4]Instituto de Astrofisica de Andalucia (IAA-CSIC), PO Box 3004, Granada, Spain
[5]Instituto Nacional de Pesquisas Espaciais, Brazil
[6]Instituto de Física, Universidade de Brasília, Brazil
[7]Eindhoven University of Technology, Eindhoven, The Netherlands

Correspondence to: Casper Rutjes (casper.rutjes@cwi.nl)

1 Performance benchmark

As complementary, we also want to test how much time the different codes need to complete an equivalent simulation. We do not pretend to do an in-depth performance benchmark of the codes, but we think this is an interesting piece of information for someone who is seeking for a code to be used in the HEAP context. Since the programs are written in different languages (Fortran, C++ and Python) and may be run on different machines with different architectures, we normalized all the completion time with respect to a reference computer configuration. And in the final paper we normalized the results with the fastest code, intermediate results are given in Tab. 1.

1.1 Procedure

First, one need to calculate the normalization factor N_{user}, using the c++ code 'pidec.cpp', written by Xavier Gourdon, and provided in the this supplementary material. It computes 8 digits of pi after a given digit position called n. It should be compiled using the GNU g++ compiler with no options, in particular no optimization options (eg '-O3'). The time taken to complete it with $n = 1000000$ (usually about 10-20 minutes) is called t_{user}. The code itself outputs it in the terminal, and it is equivalent to the 'user time' given by the 'time' bash command. The reference time t_0 is set to 1162 seconds, and the normalization factor is then given by $N_{user} = t_{user}/t_0$.

The one million 1 MeV electron beam simulation is used as the comparison case. If the considered code is parallelized, it should run on one single thread, but any compilation options can be used to make it as fast as possible. In any case, one should make several runs and get an average time to minimize the estimation error. This will give a simulation completion time that must be multiplied by N_{user} to get the normalized completion time.

Table 1. Summary of the performance (completion time).

Code	GEANT4D	GEANT4L	MC-PEPTITA	EGS5	FLUKA	GRRR $\mathrm{d}t = 25$ ps	GRRR $\mathrm{d}t = 2.5$ ps
CPU	Q9650 3.0Ghz			Xeon E-3 1271 3.6Ghz		Xeon X7350 2.9Ghz	
pidec time (s)	1 162 s			596 s		1 362 s	
Norm. Fact.	1			1.95		0.85	
Sim. time (s)	206	241	21 040	425	109	3 017	34 451
Norm. Sim. (s)	206	241	21 040	829	213	2 564	29 283