LAL/RT 07-15
EUROTeV-Report-2007-056
November 2007

GUINEA-PIG++: An upgraded version of the linear collider beam-beam interaction simulation code GUINEA-PIG

C. Rimbault, P. Bambade, O. Dadoun, G. Le Meur, F. Touze, M. del C. Alabau Pons*
LAL, Univ Paris-Sud, CNRS/IN2P3, Orsay, France

D. Schulte
CERN, Geneva, Switzerland

* and IFIC (CSIC-UV), Valencia, Spain

*Contribution to "***PAC'07***"*
Albuquerque, New Mexico, USA, 25-29 June 2007

U.M.R
de
l'Université Paris-Sud

Institut National de
Physique Nucléaire et de
Physique des Particules du CNRS

LAL ~ Bât. 200 ~ B.P. 34 ~ 91898 Orsay Cedex ~ France

LAL/RT 07-15
EUROTeV-Report-2007-056
November 2007

GUINEA-PIG++: An upgraded version of the linear collider beam-beam interaction simulation code GUINEA-PIG

C. Rimbault, P. Bambade, O. Dadoun, G. Le Meur, F. Touze, M. del C. Alabau Pons*

LAL, Univ Paris-Sud, CNRS/IN2P3, Orsay, France

D. Schulte

CERN, Geneva, Switzerland

Abstract

GUINEA-PIG++ is a newly developed object-oriented version of the Linear Collider beam-beam simulation program GUINEA-PIG. The main goals of this project are to provide an reliable, modular, documented and versatile framework enabling convenient implementation of new features and functionalities.

*and IFIC (CSIC-UV), Valencia, Spain

1 Introduction

GUINEA-PIG++ (GP++) is a new developed object-oriented version of the beam-beam interation simulation C code GUINEA-PIG (GP) [1]. This tool is used by a large part of the linear collider community, since it offers a high modelisation level of electromagnetic and quantum phenoma occuring during a e^-beam-e^+beam collision [2]. It provides beam particle distributions after interaction as well as background (beamstrahlung, pair production, hadrons, Compton scattering...) and luminosity spectra, for which purely analytical treatments do not exist.
To facilitate the implementation of new features and code extensions, GP was converted in C++, GP++, since object oriented programming offers many advantages. An object is an instantiation of a specifying class. It combines data with functions (methods) accessing the data or modifying it. All classes are organized in a class hierarchy (by inheritance), enabling the use of common code. Such a structuring leads to more flexibility and extensibility. In fact, the data being local rather than global, code evolution becomes easier. Moreover, new objects and their behavior can be defined as incremental modifications and extensions of existing objects. Finally, the code is written in terms of similarities more than differences, this allows the separation of general structures from specific implementations (abstraction).

2 Description of GP++ version

Starting from the GP C version, the original algorithms are first kept. The structures existing in the GP C version become classes in the C++ one, so that most of the "static" data are put into objects, local to classes. Thereby, it becomes easier to control their scope and their management. So done, it should be possible to reconsider some class structuring or algorithms to improve the code efficiency.
The GP++ code is managed using the configuration management tool CMT, first developed at LAL-Orsay [3] and designed to formalize sofware production around a package-oriented principle. The environment provides conventions (to name and address packages, files and directories) and tools to automate so far as possible the implementation of these conventions. It allows to describe the configuration requirements and to infer automatically the effective set of configuration parameters needed to build or run the packages. The management of the code environment is thus very secure.
The versioning, updating and releasing of GP++ are achieved with the collaborating version manager SVN [4], successor of CVS. It provides up-to-date tools derived from the long experience of CVS. The main feature of SVN is to enable a good tracking of all operations made by all developers, offering a web interface and a clever visualization of the differences between any given revisions.
GP++ is developed and distributed using the web software collaborating development tool TRAC [5].
Recoding GP from C to C++ had no consequences on CPU time and an improvement of performances should be obtained reorganising and modifying some algorithms. This was

tested measuring speed of execution of GP and GP++ running on a Darwin computer architecture. As an illustration, the amount of CPU time spent for varying numbers of particles is shown in table 1.

Macroparticles	10^4	10^5
GP	0:56.71	11:14.37
GP++	0:54.07	7:33.58

Table 1: Comparison of the amounts of CPU time, in minutes, spent to run GP and GP++, for two different number of macroparticles used in the simulation.

The GP++ was validated testing several variables, as the luminosity, the beam particle distributions, the angular variables displayed in figure 1, the background spectra, like for the electromagnetic pairs shown in figure 2 or Compton scattering shown in figure 3. Comparisons between GP and GP++ results, produced for the Nominal ILC beam parameter set, present a very good agreement. The small differences are due to some branching statements made on criteria based on random values, depending on floating variable representation.

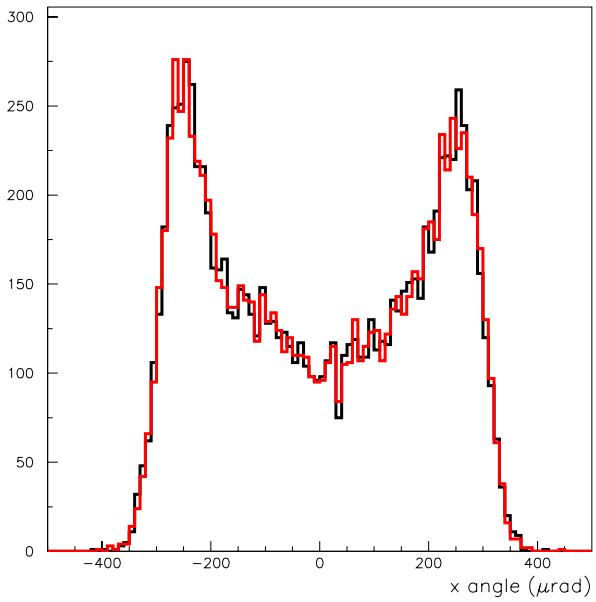

Figure 1: Angular divergence of the beam particles after interaction for Nominal ILC beam parameters, produced with GP (in black) and GP++ (in red).

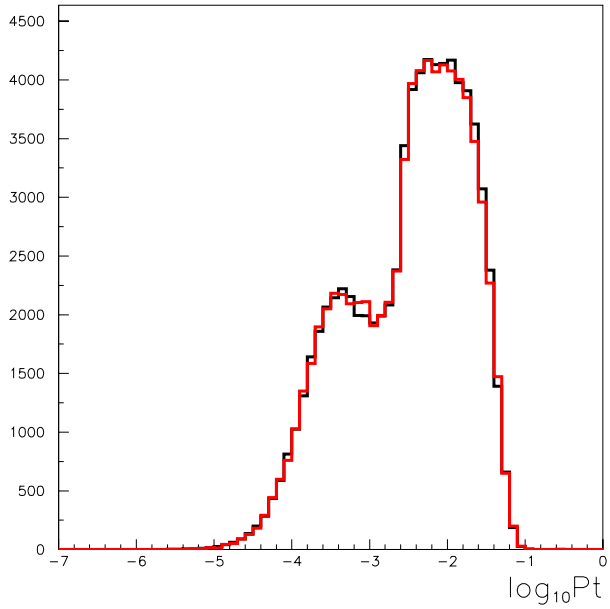

Figure 2: Logarithmic distributions of the electromagnetic pair background transverse momenta for Nominal ILC beam parameters, produced with GP (in black) and GP++ (in red).

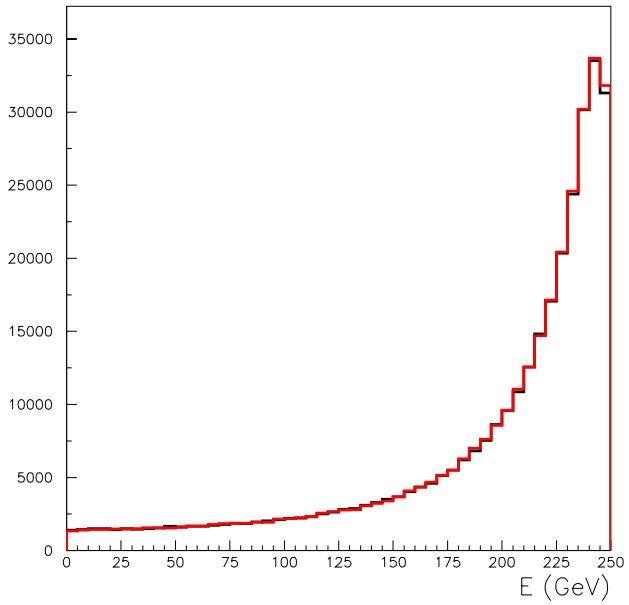

Figure 3: Distributions of the electron energy in the Compton electron process for Nominal ILC beam parameters, produced with GP (in black) and GP++ (in red).

3 New features

GP++ offers new running features. The original random number generator assumed a 32-bits computer. In GP++, a new algorithm, Haynes'algorithm [6], is added in order to enable the code to run on 64-bit computers. The appropriate choice of the random generator is then checked before the computation. In addition, a new keyword **rndm_seed** allows to choose a specified seed for the random generation, this is particularly useful to generate simultaneous runs requiring different random sequences. This new option was tested running GP++ on the LCG-EGEE [7] GRID under ILC Virtual Organisation to produce high statistics GP++ files [8].
Physics simulation is also improved, with the possibility to apply beam-beam space charge effects on a sample of Bhabha events, using dedicated code to generate the input file. This developement is detailed in [9, 10] and was used to study the impact of beam-beam effects on precision luminosity measurements at the ILC.

4 Performance optimisation

Since beam-beam interaction simulations are often used in a full accelerator simulation, for example to study bunch stability or feedback procedures, it is important to optimise the computation time. In the simulation, the beam particles are replaced by typically 20000-500000 macro-particles. The beams are cut longitudinally into slices, each slice of one beam interacting with each slice of the other beam. Slices are transversally cut into cells, and the charge of the macro-particles is distributed onto this grid according to their position. Then the potentials on the grid points are evaluated and from the forces on the particles. This computation is the most time-consuming operation (as long as background computations are not involved). To integrate the field equations, it is important to use the discrete Fourier Transform option, provided by the Fastest Fourier Transform in the West (FFTW) library (the computation time is then reduced about a factor four compared to the locally programmed FFT). An interface with the latest 3.2.1 version of FFTW [11] was implemented.
The computation of a large number of particle interactions is time-consuming as well. Good speed-up could be expected with parallel computing. An approach using Message-Passing Interface (MPI) [12] is under development: a simple way to implement a parallel algorithm would be to distribute particles among nodes and perform associated computations on each assigned CPU.

5 Further developments

Abstract Input/Output (I/O) interface enables to change the program design. Abstract classes are an appropriate way to allow user's application to plug any I/O formats as well as graphical interfaces. GP++ currently operates with ASCII format. For future intensive ILC simulations, other formats could be considered, like for example HDF5 [13] which is a general purpose library and file format for scientific data storing.

The specifications of the grid used to compute the beam-beam interaction are defined by the user. One needs to enter consistent dimensions and numbers of cells, according to the beam-beam configuration. For some particular cases (large offsets, large disruption...), it can be difficult to find an appropriate grid parameter computing set. For this reason, automatic consistency checks and adjustments of internal computational grid parameters are under study.
Concerning the physics, implementation in GP++ of the spin depolarization is the next main development.

6 Documentation

All informations concerning GP++ can be found from this address:
https://trac.lal.in2p3.fr/GuineaPig.
This web site consists of, among other things:

- a wiki page containing all informations, documentations and links for downloading the code.
- An interface for browsing the SVN repository allowing one to access all versions of the code stored in the SVN repository.
- A ticket management system.

The ticket management system provides an efficient tool for bug reporting or suggesting improvements and developments.

7 Acknowledgement

This work is supported by the Commission of the European Communities under the 6^{th} Framework Programme "Structuring the European Research Area", contract number RIDS-011899.

References

[1] D. Schulte, Ph. D. Thesis, University of Hamburg 1996. TESLA-97-08.

[2] K. Yokoya, P. Chen, "Beam-beam phenomena in linear colliders", KEK Preprint 91-2, April 1991.

[3] http://www.cmt.site.org

[4] http://subversion.tigris.org

[5] https://trac.lal.in2p3.fr/GuineaPig

[6] D. E. Knuth, " The Art of Computer Programming", Volume 2: Seminumerical Algorithms, 3rd edition (Addison-Wesley, Boston, 1998).

[7] http://public.eu-egee.org/

[8] http://flc-mdi.lal.in2p3.fr/spip.php?rubrique17

[9] C. Rimbault, P. Bambade, K. Mönig, D. Schulte, "Impact of beam-beam effects on precision luminosity measurements at the ILC", EUROTeV-Report-2007-017, to be published.

[10] C. Rimbault, P. Bambade, K. Mönig, D. Schulte, "Bias on Absolute Luminosity Measurements at the ILC from Beam-Beam Space Charge Effects", PAC07 conference, FRPMN012.

[11] http://www.fftw.org/

[12] http://www-unix.mcs.anl.gov/mpi/

[13] http://hdf.ncsa.uiuc.edu/HDF5/