First Calorimeter Simulation with the FLUGG Prototype

M.Campanella[a], A.Ferrari[b], P.R.Sala[b], S.Vanini[a]
[a] *INFN sez. di Milano, via Celoria 16, I-20133 Milano, Italy.*
[b] *SL, CERN, CH-1211 Geneve 23, Switzerland, on leave of absence from INFN sez. di Milano, Italy.*

December 10, 1999

Abstract

FLUGG is an electron and hadron MonteCarlo simulation package that integrates the GEANT4 geometrical description into the FLUKA code. At the present stage of development the FLUGG prototype is able to perform simulation in complex multi-level geometries. FLUGG and pure FLUKA simulations of the Test36 lead-scintillator hadronic calorimeter have been performed as a test of the geometry interface. Comparisons with experimental data are also presented.

1 The FLUGG Project

The goal of the project is to allow the FLUKA MonteCarlo[1], coded in FORTRAN, to call the geometry functions of the object-oriented HEP simulation software GEANT4[2], which is coded in C++. The key features of this project are:

- Simulation of particles behaviour using the whole of the FLUKA physics package, like e.g. interaction models, charged particles tracking, biasing. The FLUKA interaction and tracking models have been demonstrated [1, 3] to be quite successful in reproducing a variety existing data. They should ensure a good degree of predictivity thanks to their theory-driven microscopic approach, and are continuously subject to improvements and enrichments. The biasing techniques available in the code are an unique tool for deep penetration studies and in general for all kind of radiation studies.
- Allow the transport code in FLUKA to efficiently access very complex geometries by means of GEANT4 routines ([4]). This will also allow to exploit

the broad range of geometry related facilities available in GEANT4 like exchanging detector geometries with Computer Aided Design - CAD - programs, detector and tracks visualisation. (See [5], [6]).

As a consequence, users will also be able to run FLUKA and GEANT4 MonteCarlo simulations with the same geometry description.

1.1 FLUKA

FLUKA ([1]) is a complete transport MonteCarlo program. It handles hadronic and electromagnetic interactions, charged particle tracking, low energy neutron transport, and so on, in a fully integrated way. It has been succesfully applied to different problems such as shielding, dosimetry, high energy experimental physics and engineering, cosmic ray studies, medical physics. It allows analogue and biased transport. Descriptions of the FLUKA physical models and comparisons with experimental data can be found in refs. [1, 3].

Its geometry package is a modification of the combinatorial geometry (CG) package developed at ORNL for the neutron and gamma-ray transport program MORSE [7]. Important improvements have been made to the original CG package, such as the implementation of additional bodies, the calculation of the distance to nearest boundary and special algorithms that minimize tracking errors due to rounding. Interplay with charged particle transport (multiple scattering, magnetic field transport) is properly managed. A limited repetition capability (lattice capability) is available. These improvements resulted in a very accurate and fast tracking, that, howewer, has some limitations: repetition of identical structures is available at one level only, the description of complex geometries can be rather cumbersome for the user.

1.2 The GEANT4 Geometry Package

GEANT4 ([2]) is an *Object-Oriented Toolkit* for HEP simulation. A detector geometry in GEANT4 is described by listing the different elements it contains and specifing their positions and orientations. This detector description is stored in a database that must bank and provide access to elements of the detector. This is achieved employing the following concepts:

- pure dimensioned geometrical shape: **solid**. The class involved (G4VSolid) provides methods to determine whether a point is inside or outside the solid, to calculate the distance from a point to a boundary either along a straight line or along an arbitrary path, and to compute the outwards pointing normal to the surface closest to the point. A set of complementary implementation of

solids can be created (this allows to import/export detector geometries from CAD, using the ISO STEP compliant solid modeller):

  - solids representing simple shapes are available directly as Constructed Solid Geometry (CSG) entities, by the corrisponding classes.
  - more complex volumes are defined by their bounding surface. They are called Boundary REPresentated Solids (BREPS).

- Unpositioned detector element with attributes: **logical volume**. It is built up from the solid volumes, and carries its attributes, like for instance its material composition.

- Positioned logical volume with respect to an enclosing logical volume **physical volume**. The logical volume can be simply "placed" in its mother volume, specifying a transformation in a given reference system. In alternative, there is the possibility to place repeatedly a single logical volume inside another, using replications or a parametrisation mechanism. In this case a single physical volume represents multiple copies of a volume within its mother (different copies are distinguished by an integer index). In the case of "replicas" the copies are all identical. For "parametrised" volumes the solid type, its dimension, the material and the transformation matrix can all be parametrised in function of the copy number by a user-implemented parametrisation function .

The GEANT4 user can define either a hierarchical geometry or a flat one. In the first case, the detector description is developed by positioning daughter volumes inside mother volumes. Daughter volumes can on turn contain sub-daughter volumes, an so on, allowing the construction of a tree-structured geometry. The volume boundaries are strictly non-intersecting. Positioning a daughter volume is accomplished through the specification of a translation and a rotation. These define the transformation from the global coordinate system to the local coordinate system centered at the centre of the mother volume. The characteristic of hierarchical geometries are:

- good tracking time performance (efficient search/lookup of volumes);

- simple way of positioning a volume relative to others;

- many translation and rotation matrix involved when transforming from the global to the local coordinate systems.

A flat geometry is the one with no hierarchy: all volumes are independently positioned in a single mother volume. The characteristic of this geometry description are:

- simple importing and exporting of CAD geometries, which are flat;
- only one coordinate transformation is required between a global coordinate system and a part of the detector;
- tracking time search is less performant than in the hierarchical case, and can be improved only with the creation of dummy volumes;
- tracking optimisation is, in any case, less performant.

Tracking optimisation is available in GEANT4. It aims to reduce the number of candidate volumes intersected by the particle along a given direction from a given point. This could be achived implementing virtual divisions, in the so called Smart Voxel method, that provides very efficient search in the volume database and involves only a fast initialisation.

1.3 Status of the Project

The FLUGG package has presently the following characteristics:

- particle transport is performed in single level and multi-level geometries;
- the geometry input is in GEANT4 format;
- the physics input and the output is in FLUKA format;
- the package runs on HP and Linux platforms;
- the transport in magnetic field has still to be coupled to GEANT4 geometry.

The status of the project allows for the use of FLUGG in a complex, multi-level geometry. In the present work a full calorimeter simulation is performed as a test of the geometry interface.

2 Technical Details about FLUGG

2.1 First Technical Challenges Faced

First of all, we investigated two kinds of technical problems:

1. the feasibility of **mixing modules** written in different programming languages: C++ code (GEANT4) with FORTRAN code (FLUKA);

2. the possibility of **decoupling physics and geometry** routines in both FLUKA and GEANT4.

The first problem is not straightforward, due to the formal differences between the two languages; we solved it building C++ wrappers as interface ([8]). Analyzing the second problem, we isolated and compared geometry calls in the two programs. These calls are few and similar, thus it is possible and easy to call GEANT4 routines from FLUKA ([8]). One wrapper for each of the FLUKA geometry calls has been written (in fig.1: G1WR, NRMLWR, LKWR).

The first version of the application included the FLUKA physics and the alpha version of the GEANT4 geometry. Several tests with simple geometries have been performed, requiring that FLUKA and FLUGG simulations were reproducing exactly the same random number sequence even after hundreds of histories. This is a simple and very powerful test, since it ensures that all the particle steps are exactly the same in the two geometry packages. However, since even small rounding differences in the two algorithms could eventually lead to different random number selection in the physics, the test is no longer feasible in complex geometries and/or for runs spanning several hours of CPU.

2.2 FLUGG Initialisation

Initialisation was an important task to handle.

The standard FLUKA input includes the geometry definition, material definitions and material to region assignments, setting of transport and interaction thresholds and accuracy, link to neutron data sets, scoring directives, biasing directives. Many of the definitions are given on a region-dependent or on a material-dependent basis. Thus, care had to be taken to keep all the potentialities of the FLUKA user initialization while transferring part of the input tasks to GEANT4 classes.

In FLUGG, the geometry is build exclusively with GEANT4 input classes, so it is completely independent and decoupled from the FLUKA input. At the initialisation stage, the wrapper JOMIWR (see figure 1) is called from FLUKA for **Geometry initialisation**. JOMIWR calls GEANT4 geometry classes for the construction of the detector and returns to FLUKA the number of regions of the detector. The GEANT4 numeration table for physical volumes is used both at run time to identify the current region number, and at input/initialization to assign region dependent parameters.

In the GEANT4 toolkit, **materials** and material-volume assignments are specified in geometry input classes, in the detector constructor file. However, the mate-

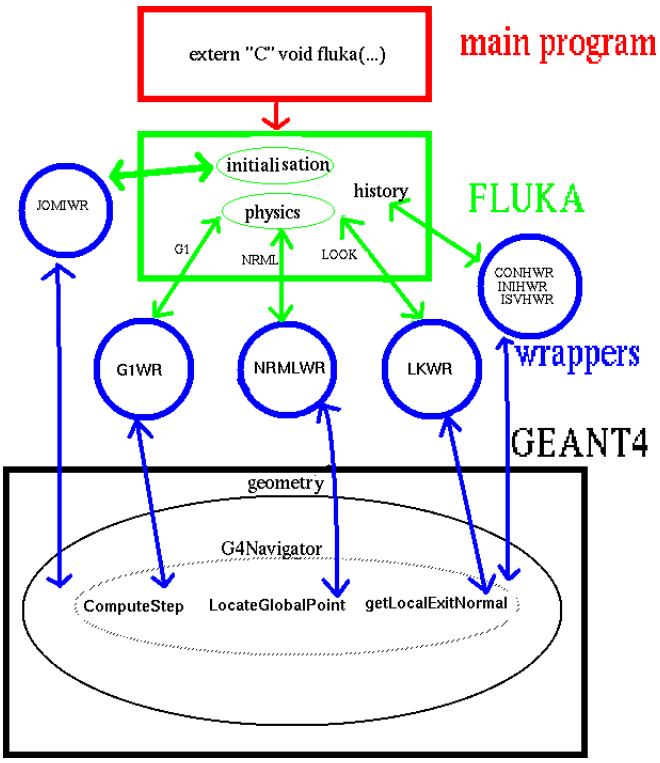

Figure 1: Sketch of the FLUGG application.

rial initialization in FLUGG cannot be restricted to the one provided by GEANT4 classes. In FLUKA, material definitions can include details that are related to the physics of the code. For instance, the user can override the defaults for the ionization density effect parameters, can define effective densities for gases, can refer to different neutron cross section sets depending on the material temperature, can adopt different degrees of accuracy in the physics processes on a material dependent basis, to optimize the balance between precision and speed. The material initialization has thus been divided into two steps: when FLUGG initialises the geometry, material information is read from the GEANT4 detector description, and translated, by means of JOMIWR, into FLUKA-formatted input cards. The newly created file containing the GEANT4 material specifications and volume-material assignments must be included into the FLUKA input file where additional properties can still be defined.

2.3 FLUKA + GEANT4 Facilities in FLUGG: Particle History

The geometry history of a given particle is defined by the volume it belongs to, placed in the geometry tree hierarchy, with all its mother volumes specified. The geometry history is the outcome of particle location computation, and it is a necessary information for tracking the particle in the detector geometry. When secondary particles are created, in a particular location (history), it is helpful to store their history together with the other informations related to them. This avoids relocation computations every time a secondary particle is taken from the stack for tracking.

FLUKA creates secondary particles and banks them in its stack with their history. When FLUKA gets a secondary particle from the stack, it doesn't recalculate the exact location because it knows it from the stored history. This feature is not present, yet, in GEANT4 (it is however planned). There is in GEANT4 geometry, and in particular in the "compute location" routines, no history output to be stored in a FLUKA readable format (both for code language reasons as for program structure). This is the reason why, in its first version, FLUGG was not able to store history information. Only the region number the particle belonged to was given and stored on FLUKA side. But this led to the following drawbacks:

1. Every time a secondary particle was taken from the stack, its history had to be recomputed, with a significant CPU penalty.

2. FLUKA received from GEANT4 only the physical volume number, so it was impossible to distinguish volumes belonging to different replicated mothers. This has no impact on the tracking, but spoils part of the FLUKA scoring capabilities.

To solve these problems, it was necessary to make the geometry classes able to give back to FLUKA the whole history after the location of particles. The first task was then to create C++ objects containing the history information and the number of secondary verteces created with that particular history. We built the new class "NavHistWithCount": every object of this class stores the pointer to the NavigationHistory of a given particle and its counter (that is the number of secondary particles created with that history).

FLUKA has been made able to handle all operations related to those objects via specific wrappers. Every time the FLUKA physics creates secondary particles, and wants to store them in the stack, the current history is saved in a NavHistWithCount object. The pointer to this object can be stored on the FLUKA stack as an integer number (ISVHWR makes a new NavHistWithCount object, saving the navigation history stored in G4Navigator in it, and returns its pointer to FLUKA). Then FLUKA can decrement or increment the history counter, every time a secondary

particle is taken from or saved in the stack (CONHWR updates the secondary particle history counter and deletes the object if the counter is equal to 0). On starting tracking of a secondary particle, taken from the stack, FLUKA reinitialises the history stored in G4Navigator (INIHWR).

In conclusion, the new class "NavHistWithCount" allow to store in memory the history of secondary particles, computed by means of GEANT4 geometry routines, every time FLUKA physics needs it. The pointer to these objects is given back to FLUKA, that manages all the related operations. The G4Navigator can then reuse the stored histories, and not recalculate them, without waste of cpu time in relocalisation.

3 First Calorimeter Simulation

In this paper we simulate with FLUGG a well known and tested Lead-Scintillator calorimeter, named Test-36 ([9]). The goal of the test is to verify the behaviour of the new interface. Since the simulation results must be completely independent from the geometry package used, the FLUGG simulations have to match the standalone FLUKA ones within the statistical accuracy. The choice of a real calorimeter as geometry benchmark allows us to perform also a comparison with experimental data. The test-36 calorimeter was chosen because its geometry allows a full testing of the multi-level capabilities and of the history saving algorithm without requiring a huge effort in building the geometry input.

3.1 Test-36 Lead-Scintillator Calorimeter

The Test-36 calorimeter [9] consisted of three identical modules. Each module was subdivided vertically into three optically decoupled towers. The front part of each tower (EM section) and the back part (HAD section) had separate readout. The depth of the EM section was 1λ (hadronic interaction lenght). The HAD section was 4λ. So, the calorimeter consists of nine towers with a total depth of 5λ for hadronic interactions. Each module had a sandwich structure of 81 layers. The sampling layer consisted of a 10 mm thick lead plate followed by a 2.5 mm thick scintillator (SCSN38) sheet. The thickness ratio between lead and scintillator was optimized to achieve a good energy resolution for hadrons. The lead plate contained 4% antimony to increase the mechanical stability. They were kept at a distance by 3.5 mm thick spacers located at the top and bottom of the plate. The stack was held together by steel rods running through the spacers. In this way no dead material was introduced in the sensitive volume. The EM section contained the first 16 layers and the HAD part the remaining 65 layers. Each tower was read out on both sides by 2 mm

thick wavelength shifter plates (WLS). It was tested at PS and SPS with e, μ and π [9, 10].

4 Simulation Analysis

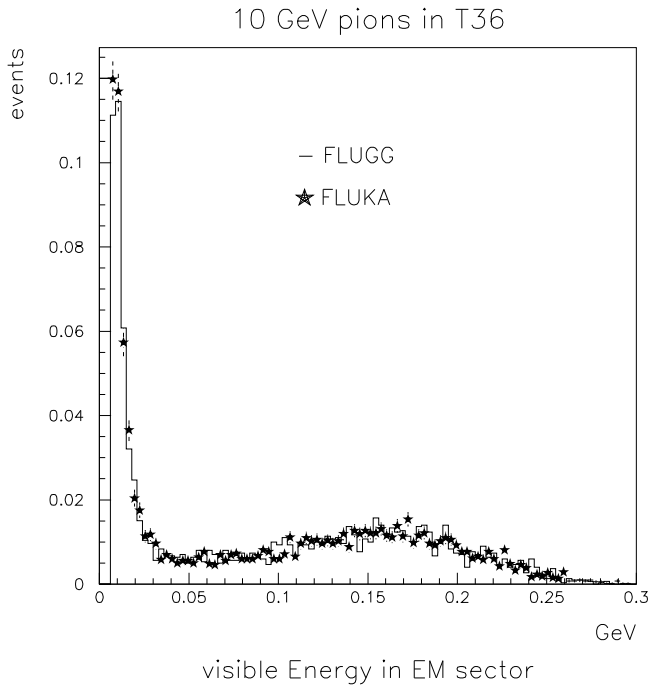

Figure 2: Energy deposited by 10 GeV π in the first 16 scintillator layers (EM sector).

We choose the following parameters for the simulation:

- 10 GeV electron and pion beams;
- detailed treatment of all interactions and of particle transport;
- low thresholds : 100 keV for $e^{\pm}$, 10 keV for γ;
- signal quenching in the scintillator with Birks parameter 0.0085 $g/cm^2/MeV$;

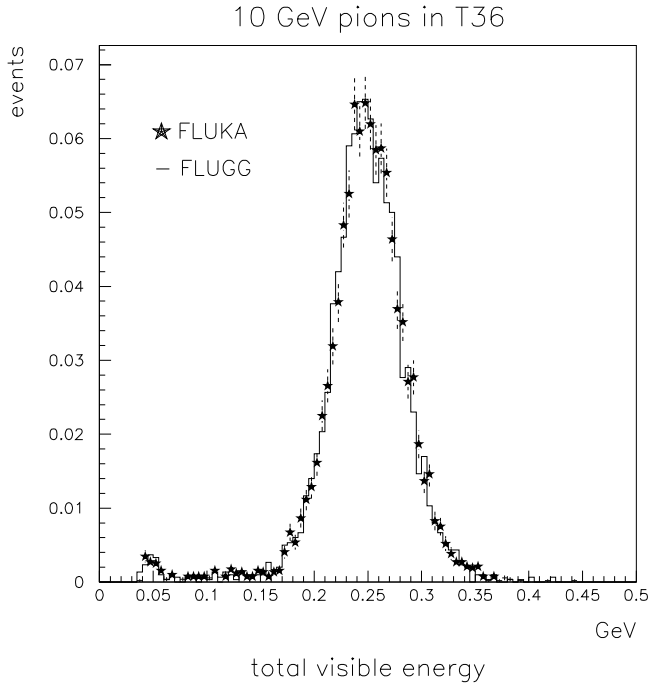

Figure 3: Energy deposited by 10 GeV π in all the scintillator layers.

- no simulation of the light attenuation in WLS fibres.

We simulated, with FLUKA and with FLUGG, the response of the calorimeter to pions of 10 GeV incident on the centre of tower 5 (the number of events in all plots is normalised to 1).

In Figure 2 and 3 the visible energy in the electromagnetic section and in the whole calorimeter for 10 GeV incident pions is plotted, for both simulations. The close similarity of the plots confirms that the tracking in the two geometries gives identical responses.

In Figure 4 the energy deposited in the central tower over the total energy is plotted. FLUKA and FLUGG plots show again no difference, thus confirming that the lateral development of the shower is tracked in the same way in the two simulations.

The ratio between quenched signal and true energy deposition is plotted in Figure 5. This plot is sensitive to the details of the tracking, since the quenching factor depends on the density of deposited energy along the particle trajectory, thus it depends on the length of particle trajectories in the scintillator. In this case, again,

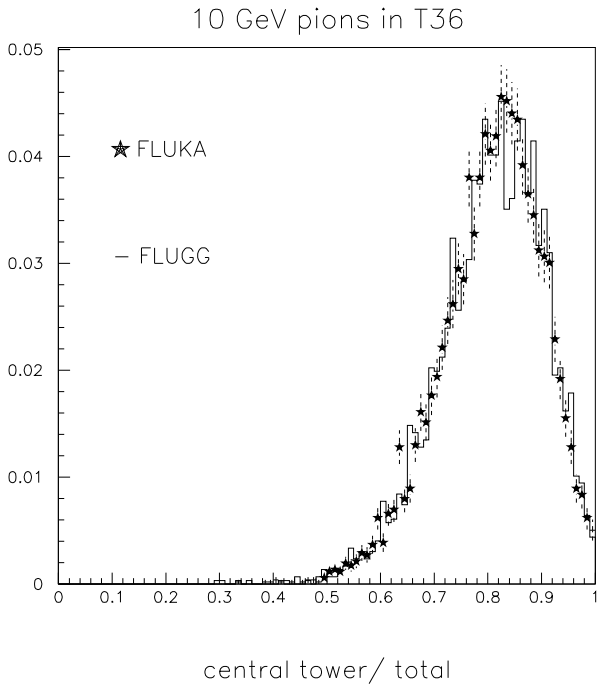

Figure 4: Fraction of energy deposited in the central tower by 10 GeV. π

FLUGG results reproduce exactly the standalone FLUKA simulation.

In Figure 6, the total visible energy is plotted for a 10 GeV electron beam. Again, FLUKA and FLUGG plots are exactly the same.

5 Comparison with Experimental Data

5.1 Energy resolution

The energy resolutions obtained with MonteCarlo simulations have been compared to experimental data ([9, 10]), adopting, as far as possible, the same cuts and procedures. For pion and electron energy of 10 GeV, the pulse height distribution has been fitted with a Gaussian, limited in between $\pm 2\sigma$ from the central value. Pion events with energy in E.M. compartment < 1.5 GeV have been discarded. The contributions of the photoelecron statistic, of the beam momentum spread, and of the fluctuations in light attenuation have been added in quadrature to the MonteCarlo resolutions (values taken from [11]).

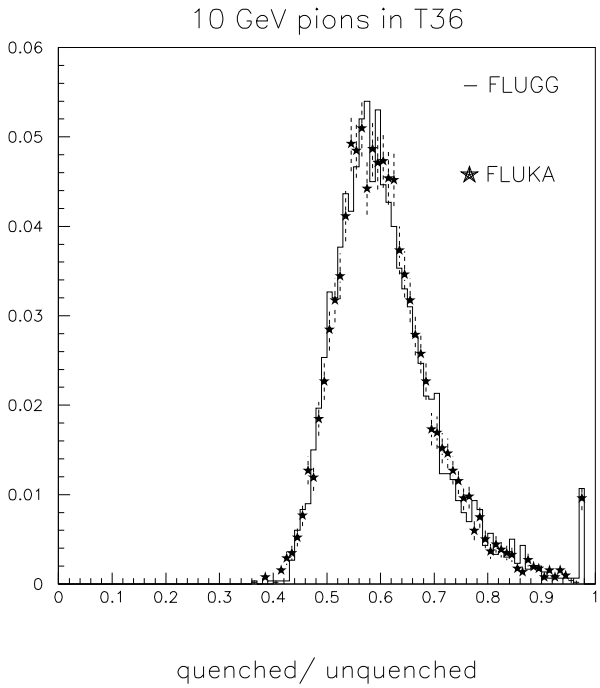

Figure 5: Ratio between quenched and not quenched signal for 10 GeV. π

We obtained:

beam	FLUGG	Exp. data
10 GeV π	$12.7\% \oplus 3.3\% = 13.1\%$	$13.8 \pm 0.2\%$
10 GeVe^-	$6.9\% \oplus 2.8\% = 7.46\%$	$7.5 \pm 0.1\%$

5.2 Shower Profile

The following table contains the energy fraction deposited in each of the nine towers of the calorimeter; we have assumed 95.2% containment as in the original paper. Boldface character values are the experimental data, underneath values are FLUGG simulation results.

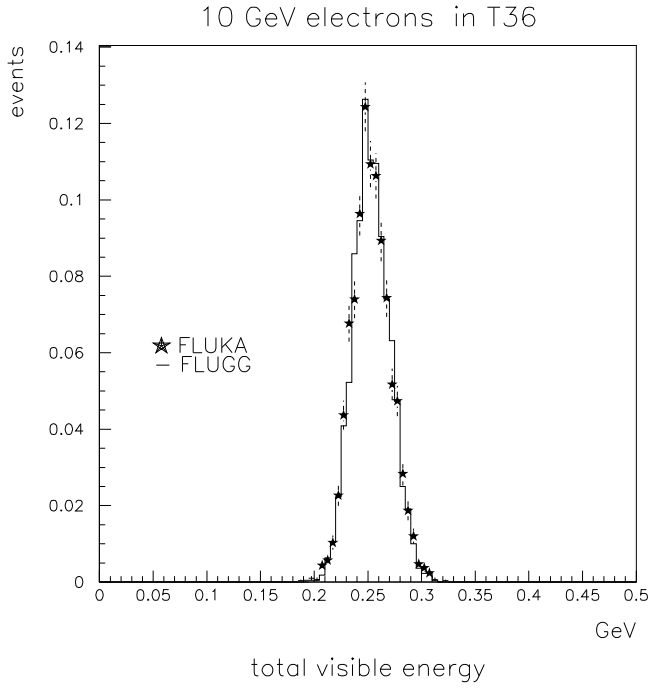

Figure 6: Energy deposited by 10 GeV electrons in all the scintillator layers.

1.4	**3.6**	**1.4**
1.3	3.6	1.3
3.6	**75.2**	**3.6**
3.6	75.5	3.6
1.4	**3.6**	**1.4**
1.3	3.6	1.3

It can be seen that the simulation results reproduce very well the shower spread in the calorimeter.

5.3 e/π

In the FLUGG simulation, the ratio between the mean value $< E_e >$ for electrons and the mean value $< E_h >$ for hadrons is 1.01. It is a bit lower than the experimental data (1.09 before leakage corrections), but, as we can see in Figure 7, this can be affected by the EM-HAD inter-calibration and by the choice of the quench-

ing parameter.

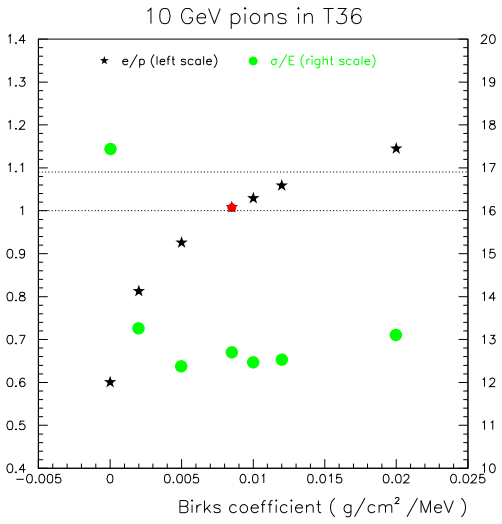

Figure 7: e/π and σ_E versus Birks coefficient

5.4 CPUs and executable file dimensions

We have compared the CPU spent running the described events with FLUKA and FLUGG, finding that the ratio between FLUGG and FLUKA execution time is nearly 1.7. This ratio has to be taken with care, since the Test-36 geometry is still simple enough to be very efficiently treated by the FLUKA geometry package. The ratio could become smaller in more complex cases, where all the GEANT4 geometry capabilities can be exploited at best.

The executable file dimensions are:

- FLUKA: 13 MBytes
- FLUGG: 22 MBytes

The total used memory during execution is about:

- FLUKA: 48 MBytes
- FLUGG: 50 MBytes

It has to be pointed out that the global FLUKA memory allocation does not depend on the characteristics of the simulation. Only part of this allocated memory is really occupied at run time.

6 Future Developments

We plan the following activities for the near future:

1. develop and test the wrappers for tracking in **magnetic field**;
2. apply to more **complex** cases like ATLAS test beams;
3. make FLUGG and GEANT4 **outputs** fully compatible to allow easy results comparison;
4. include in FLUGG the GEANT4 **geometry facilities** (like visualisation drivers, geometry debugger and editor).

7 Conclusions

We built a MonteCarlo simulation environment, FLUGG, which incorporates the GEANT4 Geometry into the core FLUKA program. The FLUGG prototype is now ready to perform simulations

- using the FLUKA input for material and simulation parameters;
- using the GEANT4 geometry input;
- obtaining the output in FLUKA format.

FLUGG has been tested in a **full calorimeter simulation**. The **multi-level geometry** has been built with GEANT4 classes. The response of the calorimeter to electrons and pions has been simulated with both FLUKA and FLUGG, with the same material definitions and simulation settings. The comparison between FLUGG and FLUKA simulations shows that FLUGG reproduces exactly the standalone FLUKA results, as expected.

In particular, we have simulated the response of the calorimeter to electron and pion beams with energy 10 GeV. We have found an energy resolution for electrons of:

$$\frac{\sigma_\pi}{< E_\pi >} = 7.46\%$$

And an energy resolution for pions of:

$$\frac{\sigma_{e^-}}{< E_{e^-} >} = 13.1\%$$

The e/h ratio obtained, for a Birks parameter of 0.0085 g/cm^2/MeV, is:

$$\frac{< E_e >}{< E_h >} = 1.01$$

We compared the simulation outcomes with experimental data, concluding that FLUGG simulation reproduce reasonably well the experimental data.

8 Acknowledgements

We acknowledge the contribution of Laura Perini, who provided encouragement and support for this work, and of Andrea Dell'Acqua, who patiently gave us always good advices.

References

[1] A. Fassò et al., Proceedings of the 3rd workshop on "Simulating Accelerator Radiation Environment", SARE-3, KEK-Tsukuba, May 7–9 1997, H. Hirayama ed., KEK report Proceedings 97-5, p. 32 (1997);
A. Ferrari, and P.R .Sala, Proceedings of the *International Conference on Nuclear Data for Science and Technology*, NDST-97, Miramare-Trieste 1997, SIF Atti e Conferenze, p. 247, Vol. I (1998);
and references therein.

[2] CERN/LHCC/97-40 "GEANT4: an Object-Oriented Toolkit for Simulation in HEP", 1997

[3] A. Fassò et al., NIM A332, 459 (1993);
A. Esposito, et al. NIM B88, 345 (1994);
Z. Ajaltouni et al. (ATLAS collaboration), NIM A387, 333 (1997) (also CERN-PPE/96-178);
M. Biaggi et al., NIM B159, 89 (1999);
G. Collazuol et al, CERN-OPEN-98-032 , submitted to NIM.

[4] P.Kent, "Pure Tracking and Geometry in GEANT4", April 1995 (unpublished)

[5] J.Apostolakis, "An Overview of GEANT-4's Geometry", RD44 collaboration, IT division, 28 April 1997

[6] P.Kent,S.Giani, “The GEANT4 Geometrical Model”, 23 April 1995

[7] M.B. Emmett The MORSE Monte Carlo radiation transport system Oak Ridge National Laboratory report ORNL-4972 (1975)

[8] M.Campanella, A.Ferrari, P.R.Sala, S.Vanini “Reusing Code from FLUKA and GEANT4 Geometry”, ATL-SOFT-98-039, 26 Oct 1998

[9] E. Bernardi et al., NIM A309 (1987), 229

[10] A.L.Tsirou, ”On the optimization of a lead-scintillator compensating calorimeter”, Doctoral Thesis, Univ. Wisconsin (1989).

[11] G. Drews et al., NIM A290 (1990) 335