Journal of the Arkansas Academy of Science

Volume 47 Article 25

1993

User-Interface Coding for the CERN/GEANT Nuclear Physics Program

David L. Roetzel
University of Arkansas at Little Rock

Wilfred J. Braithwaite
University of Arkansas at Little Rock

User-Interface Coding for the Cern/Geant Nuclear Physics Program

David L. Roetzel and W. J. Braithwaite
Department of Physics and Astronomy and
Department of Electronics and Instrumentation
University of Arkansas at Little Rock
Little Rock, AR 72204

Abstract

Explanations will be given of the various user-written routines required by the Monte Carlo detector-modeling program GEANT, developed by CERN, the European Organization for Nuclear Research. User-written routines must be linked with the CERN library to accomplish the researcher's intentions. Examples will illustrate how GEANT passes information to subprograms needed to model events. Various data structures used by GEANT library calls and included in each user routine, are similarly illustrated. Both computational-speed and memory-size limitations need to be factored into the construction of a simulation model. This will constrain the calls used in the user-written routines. Examples are provided of GEANT input data flags, defined by the user to determine simulation parameters and to control various testing choices in GEANT.

Introduction

Computing resources available to the majority of scientific users have increased dramatically in recent years. This has made it possible for most scientific users to start taking advantage of the more sophisticated and powerful software programs. An example of this type of program is the Monte Carlo event processor GEANT (Geant, 1992)–one of the most widely used by the high-energy and nuclear physics community. This program was developed by CERN, the European Organization for Nuclear Research in Geneva, Switzerland.

Until recently only large institutions with large programming staffs had sufficient computing power to run this type of program. As a consequence the GEANT user interface is quite complex and requires intricate, custom-coded user routines. To help new users, this paper will illustrate these user-subroutines, which are usually written in FORTRAN to take advantage of common data structures used by GEANT. (GEANT itself is written in FORTRAN.) Then these user-subroutines are linked with the rest of the GEANT package and run as a single unit.

GEANT code, relegated to the user, is concerned with several issues. These include the type and energy of each incoming particle, the shape and construction of the detector system, when to take in additional information and what information to output.

During 1992 the CERN program library was ported to a cluster of Digital Equipment DEC-5000 workstations at the University of Arkansas at Little Rock (UALR). Along with GEANT, other programs downloaded were HBOOK, ZEBRA, PAW, and PATHCY. These programs, along with a structural flowchart, were provided with the GEANT package to help the user.

Materials and Methods

Before the programs provided with the GEANT package were useful in our own simulations, many phone conversations with expert users were needed (Gagliardi, 1992; Prindle, 1992; Stancu, 1992; Throwe, 1992). This section communicates some of what was learned by our group, to provide beginning users a clearer picture of how to use these routines. A structural guide, summarizing the use of these routines, is included in the Results and Discussion section.

Memory size is the first concern to be addressed in the user-generated MAIN program. The user must set the size of dynamic memory in a call to GZEBRA. In order to minimize the swapping of code between memory and disk, the user should set the size of dynamic money close in value to the maximum real memory that is available. If the user runs out of memory during a simulation, decreasing the number of primary events tracked at once, which depends on the complexity of the primary events, is an option. Since the memory used to store data in each job is subsequently cleared after the processing of each group of primary events, simulation data in memory must be accommodated in some fashion (e.g., stored on disk) before going on to the next group of primary events.

UGINIT is the first user-subroutine called from the user-generated MAIN program. UGINIT sets up each run by reading in free format data records with a call to

GFFGO, which must be provided by the user. These input data records control the following processes: general run parameters (standard histograms, number of events, time); physical processes (Compton, Annihilation and other flags, energy cuts, minimum and maximum energy of cross sections); debug and I/O (events to debug, data structure names to save or print; user flags); user applications (flags, data areas); and other. Also, customized data records may be defined with each call to FFKEY.

GPART and GMATE are respectively called to set up tables for the standard particles and the standard materials. If custom material is used in the detector, GSMATE must be provided by the user and subsequently called to augment the call to GMATE. When using GEANT to simulate the response to new material compositions, calls to GSMATE can be made flexible with the use of customized data records (with calls to FFKEY). If this material is to be used in a detector, it must be made sensitive with a call to GSTMED with the ISVOL parameter set greater than zero.

To define detector geometry, the user must provide user-subroutine UIGEOM (subsequently calling it within UGINIT). This is done by providing sizes and shapes of the various volumes, made from simple geometric shapes: boxes, tubes, spheres, cones, etc. These may be placed side by side or inside each other to make complex composite shapes. A material number is assigned to each component volume separately. Any of these may be listed as sensitive volumes (in GSDET or GSDETV), indicating they are detectors not just structural material. Customized data records (with calls to FFKEY) may be used for greater flexibility, when using simulations to decide detector sizes or tuning proposed setups for detector shapes.

The MAIN program now calls the GRUN routine of GEANT. This sub-program controls the event and particle tracking until the end of the run. GRUN calls GTRIG, which in turn calls user-subroutine GUKINE, the initial kinematics setup. Calls within GUKINE to GSVERT describe each incoming beam particle's position, assigning a track number to each beam particle. Also within GUKINE, each call to GSKINE assigns a particle type and a momentum to each track number.

There are interfaces available to GEANT to access other Monte Carlo event generators to provide beam information, but calls to GSVERT and GSKINE are the easiest procedure when initially testing a detector. For example, when initially testing the response of silicon detectors to pions, user records (with calls to FFKEY) may be used to vary the incoming energy in assessing detector response. (Also, calls to FFKEY may be used to change particle type, to cross-check the predicted detector response with known plots.)

GTRIG next calls user-subroutine GUTREV. GUTREV usually consists of a call to GTREVE, followed by executing subsequent instructions within GUTREV for debugging and drawing tracks. The drawing part of GEANT has been left out of the discussion, due to the many differences in video libraries. However, drawing routines are not difficult and several examples are provided in the GEANT manual.

GTREVE takes over from GTRIG to do loop control. This consists of tracking the particles through the detector until they exit or decay. Secondaries may or may not be tracked according to the setup. GTREVE first calls user-subroutine GUTRAK which in turn calls GTRACK which subsequently calls user-subroutine GUSTEP.

GUSTEP is the main data-taking module. Here the user decides if something has happened. This is done by checking whether the index NGKINE is greater than zero. The user may then accumulate energy deposition, test particle types for decay, etc. Then the user can accept or reject particles for continued tracking. The user can also find out if particles have gone from one volume to another or how close they are to doing so. Also, the user may store the space points for subsequent plotting with GSXYZ.

GTRIG again takes over to call user-subroutine GUDIGI if it has been linked, but using program calls to HBOOK in the next user-subroutine GUOUT seems a much easier alternative. HBOOK files may be read by PAW, CERN's Physics Analysis Workstation program for detailed output (e.g., plots).

After user-subroutine GUOUT is finished, GRUN checks to see if all particles have been tracked to the limits of the detector. In that case, the user-generated MAIN program takes over and passes control to the user-subroutine UGLAST for final output and file closings.

Results and Discussion

Information about each current event is passed from one sub-program to another by using labeled-common data areas that can be accessed by user written code at anytime. Additional labeled-common areas may be constructed by the user if there are no naming conflicts. The important common areas and the data structures using them are indicated in the following tables. Table 1 and 2 summarize the authors experience with the common areas, the data structures and the subroutine calls within the program GEANT. Similarly, Table 3 illustrates the naming relationships among various calls to GEANT subroutines. These tables were constructed by the authors to illustrate the most important variables in GEANT.

Table 1. Module Structure of GEANT Subprograms

DEFINITIONS OF TERMS

ZEBRA - memory allocation monitor
event - particle has entered the detector
track - kinematics of the particle
step - user-defined distance along track before sampling
hit - interaction between particle and detector
[base, cons, geom, hits, kine, trak - GEANT manual sections]

PROGRAM SET-UP

MAIN user routine
GZEBRA initialize ZEBRA system, core allocation

INITIALIZATION

UGINIT user routine
base CALL FFKEY (Card, Variable, # of Variables, Type)
GINIT initialize variables
GFFGO interpret user run control records
GZINIT initialize ZEBRA core divisions and link areas
clears GCBANK common
GPART create JPART data structure
uses GPHYS common
GMATE create JMATE data structure
uses GCMATE common
UIGEOM user routine
cons CALLL GSMATE(#, Name, Parameters,...) uses JMATE
CALL GSTMET(#, Name, Material, Record Hit,...)
uses JTMED data structure
uses GCTMED common
geom CALL GSVOL(Name, Shape, Medium #<...)
uses GCVOLU common
CALL GSPOS(Name,..., Mother Volume, Position,...)
uses GCPUSH common
hits CALL GSDETV(Set for Detector, Volume,...)
creates JSET data structure
uses GCSETS common
CALL GSDETH(Set, Name,...)
phys GPHYSI prepares cross-section and energy loss tables
uses GCPHYS common

Table 2. Module Structure of GEANT (Continued)

RUN CONTROL

GRUN loops through events
GTRIG called by GEANT
kine GUKINE user routine controlling initial event kinematics
CALL GSVERT (Vertex Position, Track #,...)
create the JSVERTX data structure
uses GCKINE common
CALL GSKINE(Momentum, Particle, Vertex #,...)
create the JSKINE data structure
uses GCKINE common

EVENT PROCESSING

trak GUTREV user routine controlling looping through tracks
GTREVE loops through tracks - calls the following
GUTRAK user routine track control
GTRACK track control - calls the following
GUSWIM transports particle in steps
GUSTEP put hits → JHITS, put space
points → JXYZ, use GCTRAK common
NGKINE event happened? if yes then
CALL GSKING (Track #) put track and
particle info → GCKING common
CALL GSXYZ store space coordinates
GUSKIP user routine skips unwanted tracks

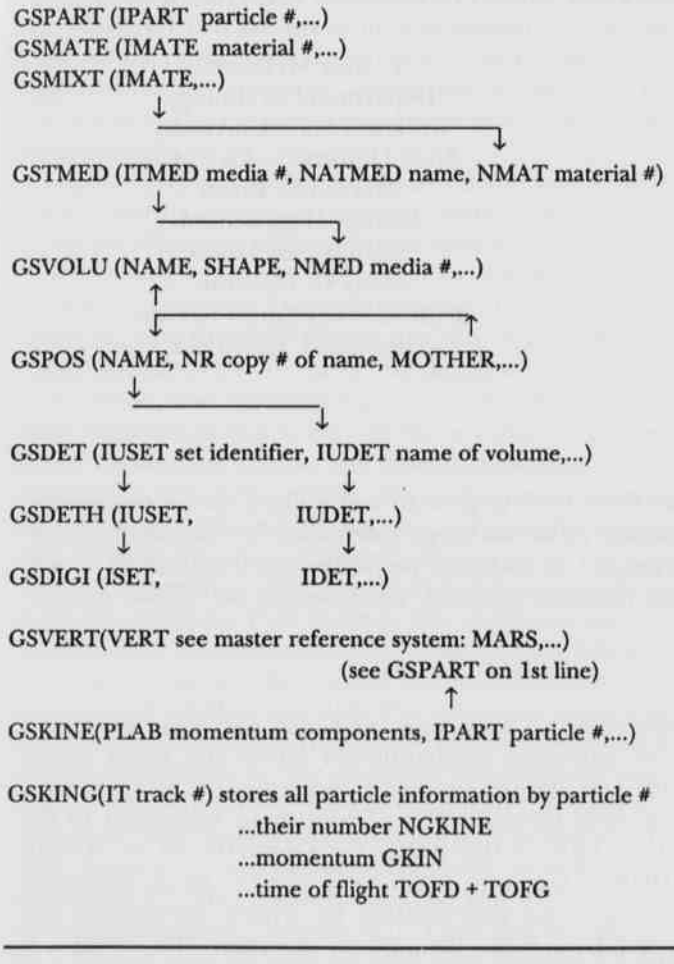
OUTPUT

base GUDIGI user routine setting digitization of data structures
hits CALL GSDIGI to create JDIGI data structure
[HBOOK has taken over much of GSDIGI's function]
base GUOUT is the user routine for output code
CALL GTRIGC to clear memory for the next event

TERMINATION

UGLAST the user termination routine
HBOOK CALL HISTDO to output stored histograms
base CALL GLAST for standard termination

Table 3. Data Name Relationships of GEANT Calls
[Arrows shows connnections between names]



Acknowledgements

This work acknowledges the help of colleagues in the Solenoidal Tracking experiment (STAR) at the Relativistic Heavy Ion Collider and colleagues in Experiment NA35 and Experiment NA49 at CERN. This work acknowledges the help of Charles Byrd and Christine Byrd in porting GEANT to the DEC-5000 workstations, and to Dr. Marcus D. Gabriel for help with the DEC-5000s. This work was supported in part by the U.S. Department of Energy and the Arkansas Science and Technology Authority.

Literature Cited

Gagliardi, S. 1992. Private Communication, CERN (Geneva, CH).

Geant 1992. Geant user's guide. CERN internal report. CERN Data Division, Geneva, CH. 3.15: 365 pp.

Prindle, D. 1992. Private Communication, University of Washington (Seattle, WA).

Stancu, I. 1992. Private Communication, Rice University (Houston, TX).

Throwe, T. 1992. Private Communication, Brookhaven National Laboratory (Upton, NY).