Fermi National Accelerator Laboratory

FERMILAB-Conf-87/230

CDF Detector Simulation*

J. Freeman
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

December 1987

*Presented at the Workshop on Detector Simulation for the SSC, Argonne, Illinois, August 24-28, 1987.

Operated by Universities Research Association Inc. under contract with the United States Department of Energy

CDF DETECTOR SIMULATION

J. Freeman

Fermi National Accelerator Laboratory[1], Batavia, IL

Abstract

The Collider Detector at Fermilab (CDF) uses several different simulation programs, each tuned for specific applications. The programs rely heavily on the extensive test beam data that CDF has accumulated. Sophisticated shower parameterizations are used, yielding enormous gains in speed over full cascade programs.

1. INTRODUCTION

CDF has several distinct uses for detector simulation. It is very convenient to have test data with known properties to use in verifying hardware readout or software programs. Another function is to provide simulated data that models physics signals for developing reconstruction algorithms (tracking or clustering), and for developing physics analysis programs to identify, for example, t-quark signatures. Related to this is modelling background to the physics signals. Another use for detector simulation is to extrapolate detector performance to regions not explored by test beam.

In CDF we have chosen to use several different detector simulation programs to achieve this list of goals, rather than a single, monolithic program. This choice was driven by CPU-time performance requirements: "Computational excess baggage" could be stripped out of the different programs without compromise to their utility. The three programs used by CDF are: CDFSIM, the primary detector simulation, suitable for modelling physics signal and background, and creating test data; QFL, a very fast simulation useable for modelling physics signal events (and limited background processes) for algorithm development and code verification; and GEANT, a program useful for extrapolating detector performance to regions not measured in test beam.

The CPU time required to simulate a typical "two-jet" event are approximately: CDFSIM - 50 VAX-780 seconds; QFL - 1 second; GEANT - 5000 seconds (GEANT, because of its extreme CPU requirements, is not actually used to simulate full events, but rather to simulate test beam conditions of a set of mono-energetic particles incident on a specific point of the detector).

The different programs, due to their different functions, do not create the same types of simulated data. Figure 1 shows schematically CDF data flow from the detector hardware through pattern recognition and physics interpretation. The blocks are YBOS banks. (YBOS is a data/memory management system used throughout CDF software

[1]Operated by Universities Research Association under Contract with the U.S. Department of Energy

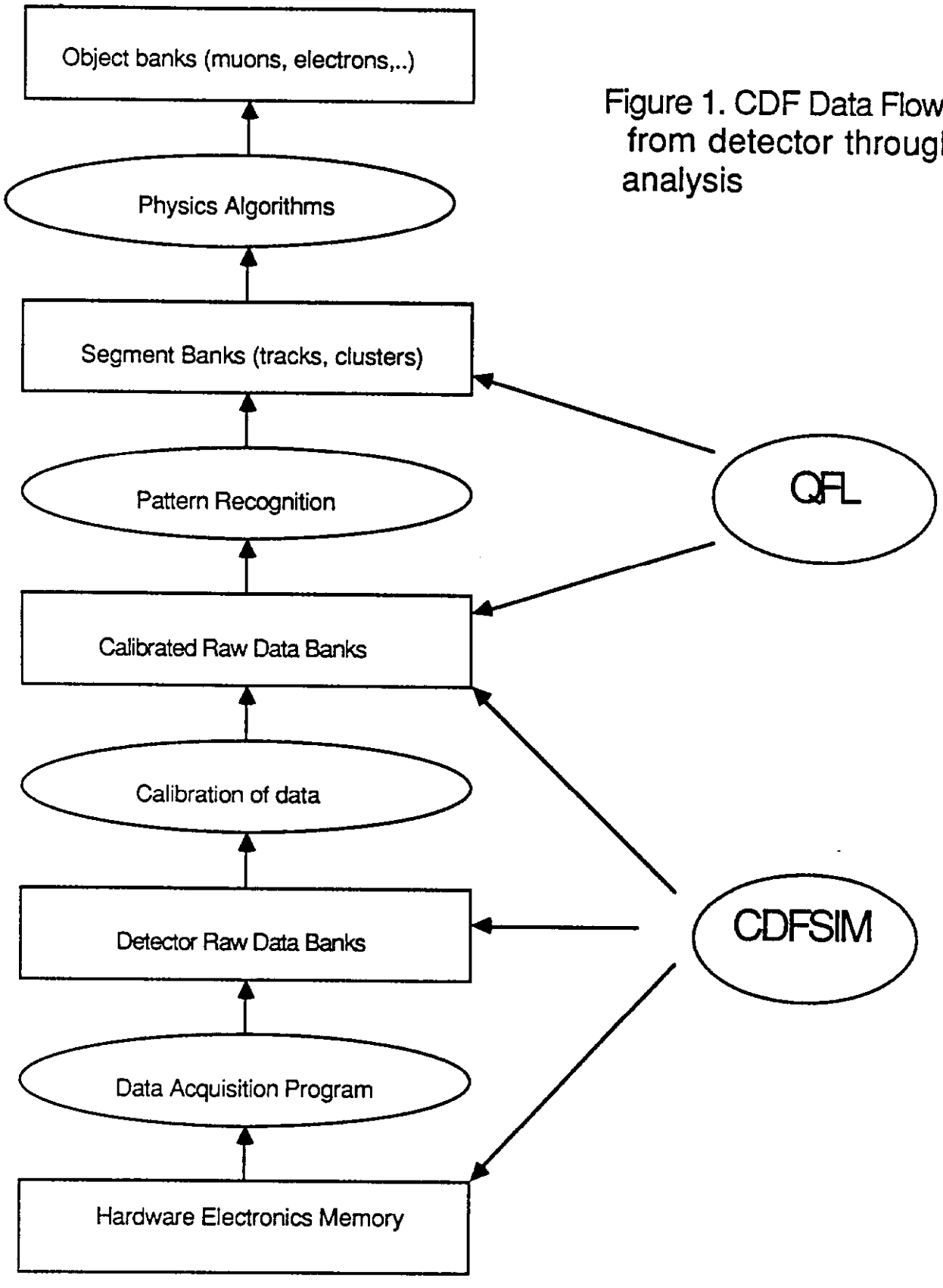


Figure 1. CDF Data Flow from detector through analysis

[1].) Ovals represent data processing programs such as track-finding. CDFSIM, with more detailed modelling of detector geometry, materials, and interactions, is used to model the lower rungs of the data-flow ladder. (Pattern recognition will approximately double the invested time per event.) QFL is used to model upper rungs for use in algorithm debugging, and gaining quick (sometimes rough) understanding of the effects of cuts. QFL, since it simulates the effects of tracking pattern recognition (rather than simulating track-chamber hits, then applying true pattern recognition) would not be suitable, for example, in studys of secondary vertex identification. GEANT, lying totally outside of this picture, is used to generate histograms of, for instance, energy deposition in an element of calorimetry.

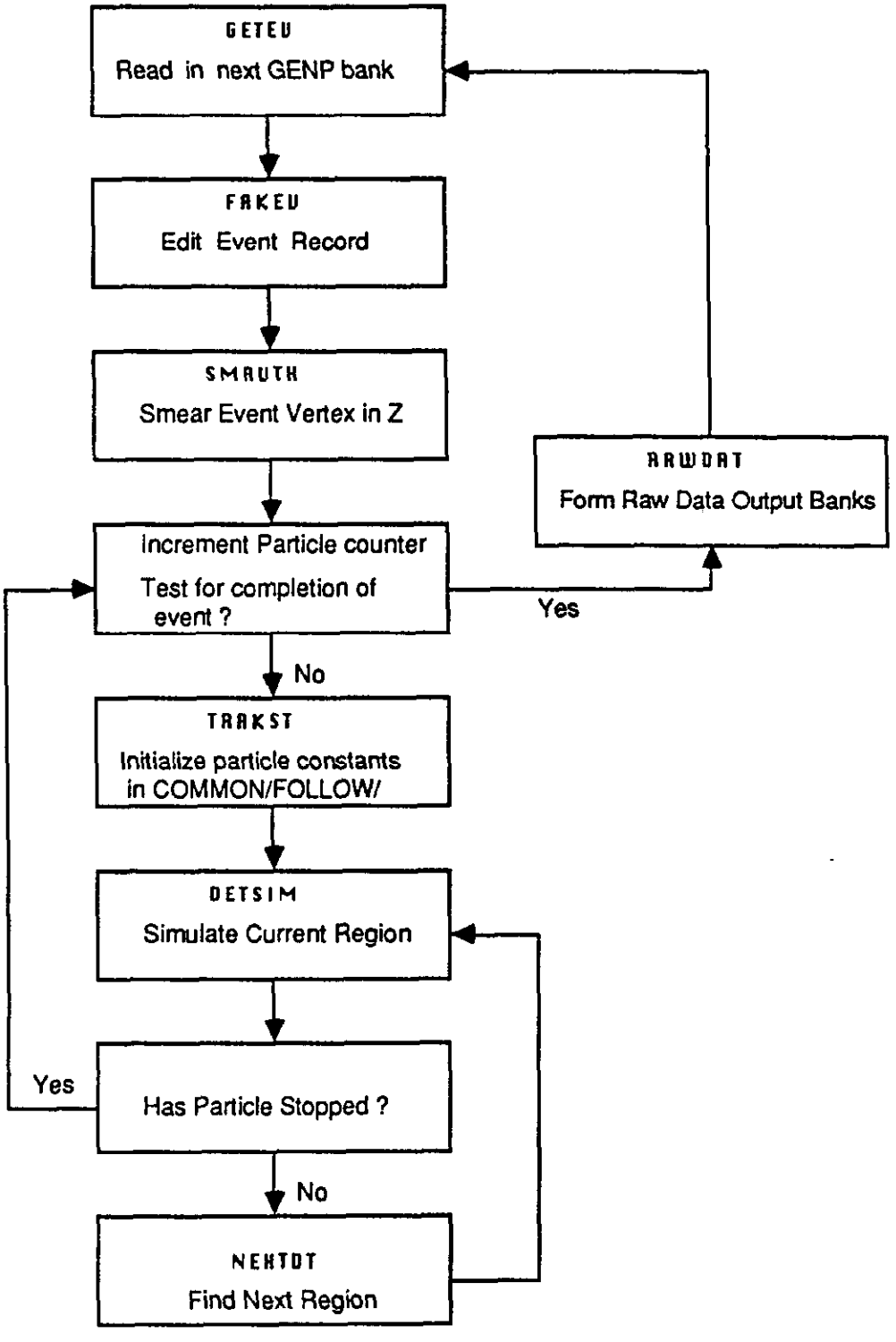
QFL is described in detail in another paper in this volume [2], so for the remainder of this paper, we will concentrate primarily on CDFSIM.

2. OVERVIEW

Figure 2 shows the program flow chart for the main event loop. Subprograms are indicated by bold type, and functionality in normal font. The simulation program views the CDF detector and volume around it as composed of many regularly shaped regions, with the boundary of one region also the boundary of its neighbor so that there are no gaps. Any arbitrary point is therefore in some region. Each region is specified by a a list of geometry constants which describes it spatially, and also by a list of regions adjacent to it. The list of neighbors is ordered in such a manner that when a particle leaves a detector, the most probable detector for it to hit next is first on the list. Last on the list is the least probable detector. Since the detectors are dense with each other, when the particle leaves one detector, we know that it has to enter another. We note that some detectors are not interesting to simulate (for instance, regions of air outside of the total detector volume). We know that when a particle has reached such a region, it is no longer possible for it to generate new data in the detector. When a particle reaches such a region, we forget about that particle and consider the next one.

All particles generated at the primary vertex are randomly assigned a path length that they will travel before decaying (derived from an exponential probability distribution which is a function of particle species and momentum); a conversion path length; and an interaction path length. These values and running totals of accumulated quantities are stored in a common block. This common stores the angular rotation of the particle, the current time in nanoseconds since the primary collision, current region, current local coordinate system, and other variables needed in the simulation of the particle. As the particles are stepped along their paths they suffer energy loss by dE/dx, bremsstrahlung, and delta-ray production, multiple scattering, pair production, decay, and interaction. If a particle travels to its decay point, it undergoes decay and its

FIGURE 2. CDFSIM EVENT LOOP (SIMEV)



daughter particles are added to the particle list. If it reaches its interaction point, secondaries are added to the particle list. When the particle passes through an active region, for instance, drift chamber or calorimetry, hit information is stored. As the event is simulated, a list of particle momenta and decay vertices is filled. After all particles in the particle list have been tracked either out of the interesting volume, or until they have stopped, the event is finished. At the end of simulation of the event, there will be a set of banks containing lists of hit data, one bank per particle that caused any data generation. Finally, the lists of hit data banks are analysed to form the set of detector "Raw Data" banks as would be read out by the hardware.

3. GEOMETRY

The simulation geometry is contained in a geometry database that is accessed at program startup. It is composed of a set of about 500 YBOS banks. Each region in CDF has a YBOS geometry bank associated with it. Contained in the bank are the type of the region (i.e. chamber or inert material); the shape of the region, and its boundary locations; the transformation matrix and displacement vector that transform CDF global coordinates into the local coordinate system of the region; material type for the region; step size for particle stepping; chaining pointers to link the region to subregions interior to it; and other miscellaneous constants that are relevant to the region in question. Access to the data base is through function calls with arguments selecting the region and property desired. Function calls, although generating cpu overhead, protect against possible future reorganizations of the geometry database. Figure 3 shows output of a display program that is driven by the geometry database. This gives an indication of the number of regions and shapes used by the program.

A part of the data base is a list for each region of all regions that are adjacent to it spatially. This list searched through upon exitting from a region to find the next region. The list is ordered from the most probable region to enter to the least probable. In general, the search is satisfied on the first test. A subroutine with input argument the region exitted, and output the region entered, is used by the simulation to access this part of the data base.

The number and types of boundaries for some pre-defined shapes are included as a part of the geometry data base. For each shape described, the number of planes that define the minimum and maximum extent of the shape, and the type of each plane (for instance, plane of constant Z or ϕ) is specified. Examples of predefined shapes include: wedge, truncated cone with cylindrical hole, or rectangular solid with rectangular hole. Any shape composed of arbitrary combinations of planes of (XYZ, radius, θ, ϕ) can be defined in this part of the data base. Only shapes revelant to CDF are actually present. A subroutine is used to test if a test point is inside or outside of a 3-dimensional region.

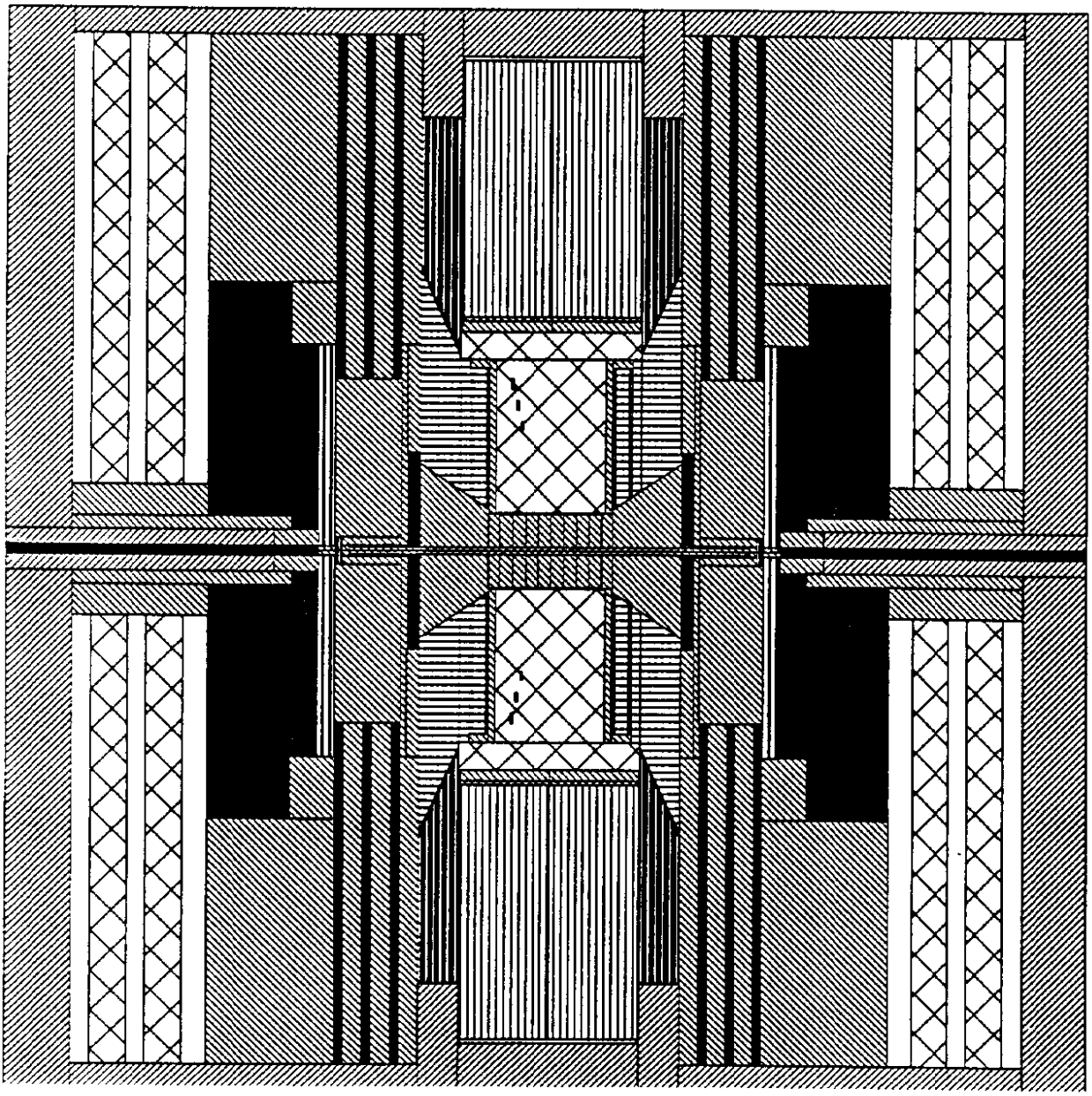


Figure 3 Display of the Geometry Database.

4. MATERIALS

A data base of materials is part of the geometry data base. The data base contains the density, radiation length, absorption length, Z, A, ionization potential, collision length, and critical energy for each material. There are about 30 materials specified. Access to the material properties is through a function call with arguments the material type, property desired, and the position to determine it at. In general, each region is a homogeneous material. Where the approximation of homogeneous material properties is invalid, the region is articially split into enough regions of differing material types to make the approximation good.

In cases where the fine structure of the region is important but where performance penalties of splitting the region into many subregions cannot be afforded, the fine structure of the material is encoded into the function that accesses material properties. There are a handful of special cases like this, in general close to the interaction vertex. The most important special case is the vertex TPC, where cables, amplifier hybrids, and structural supports are encoded into the function.

5. MAGNETIC FIELD

The magnetic fields of the central solenoid and the forward muon toroids determined by field mapping are stored in a data base. The data base is accessed at run startup. When required, the B field is determined by a call to a subroutine. Input is the space point where the field is desired, and the local coordinate system the field should be determined in. Output is a 4-vector of XYZ components of the field direction unit-vector, and the field magnitude.

6. CALIBRATION CONSTANTS

CDF possesses a calibration data base, where all calibration constants used to convert electronics counts to physical units (time, position, or energy) reside. The data base is accessed by specifying the run number to fetch constants for. CDFSIM uses this data base to fetch these constants for use in simulation. Constants are fetched for the run to be simulated (default is run 1), and the inverse of these are used in calculating electronics counts. This technique guarantees automatic consistency between simulation and analysis. In addition, it also insures that code that works on real data also will work using simulated data.

7. PARTICLE TRACKING

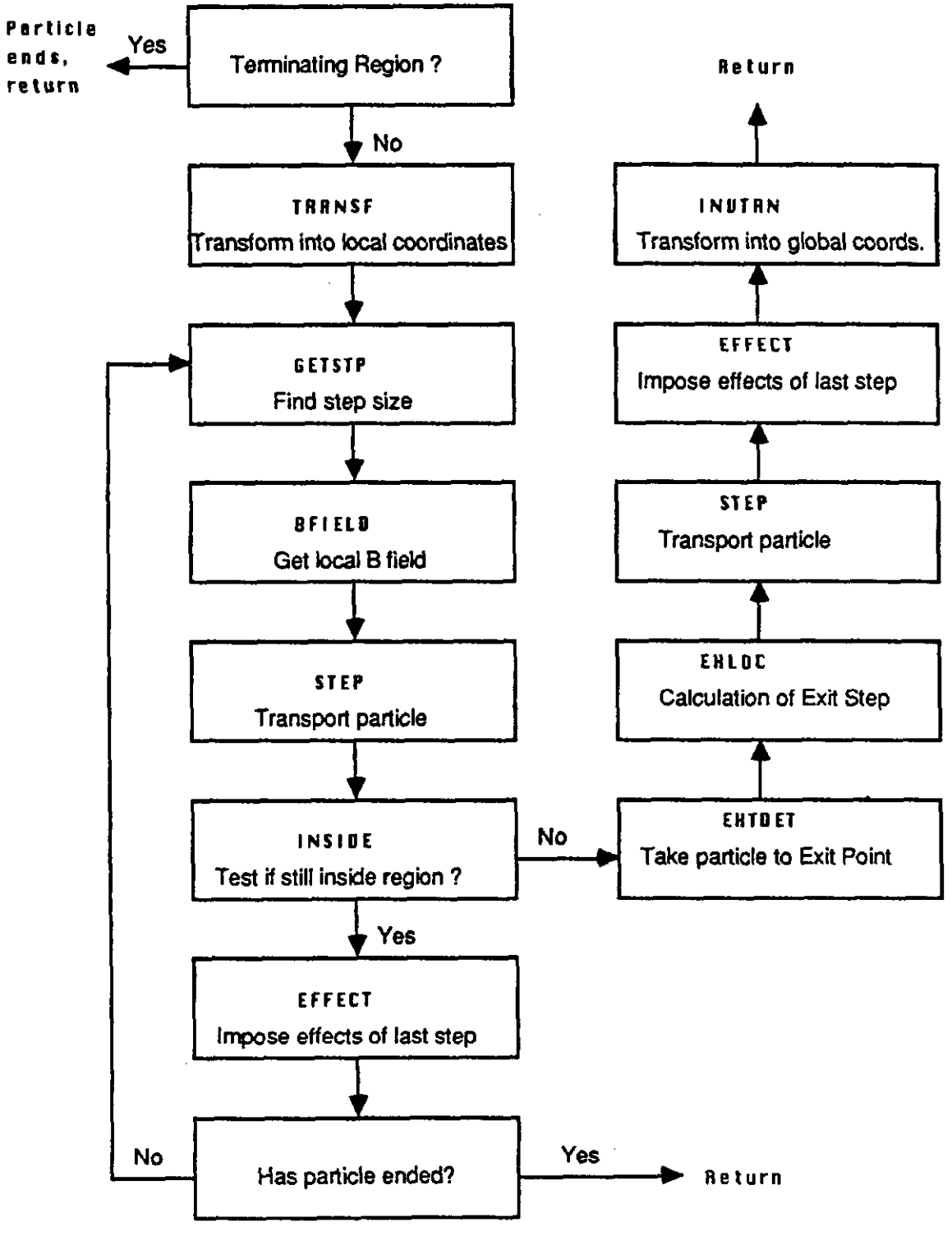
Figure 4 shows a flow chart for tracking particles through a generic region. Initally, the routine checks to see if the detector to be simulated is a terminating region outside of CDF. If so, the particle ends, and control is returned to the calling program. After transforming the 6-component particle position-momentum into local coordinates, a loop of stepping the particle through the volume is entered. An appropriate step size is calculated, taking into account the particle position, momentum, and charge, the current detector, distance remaining before interaction or decay. The local B field is determined. The particle is transported through the displacement, and the new XYZ position of the particle is tested to see if it is still in the region. If it is, effects corresponding to the step are imposed on the particle (energy loss or multiple scattering, for instance). If the region is instrumented to make measurements (a chamber or calorimeter for instance), a routine is called to calculate hits and append them onto the LISP bank. Then a new step size is calculated, and so on.

If the step has transported the particle outside of the region, the last step is undone, and a routine is called to calculate the displacement to the boundary of the region, step the particle to the exit point, transform back into global coordinates, and return. The routine does analytic calculation of the exit step whenever possible (if the B field is 0, if the particle is neutral, or if the equation of the intersection of the particle's motion in the local B field and the boundary to be exitted is analytically soluble). If analytic calculation is not possible, the routine uses an iterative stepping/interpolation procedure to calculate the exit point.

8. TRACKING CHAMBER SIMULATION

If the region currently being stepped through is a tracking chamber, intersections of the particle trajectory and layers of wires are calculated for all layers that are intersected during the step. The hit data stored for each intersection include the distance of closest approach in the drift plane between the track and wire, charge deposited, and ϕ and θ inclination angles. After all particles have been simulated, the hit data may be transformed into detector oriented data. All data for each wire of the chamber is put together, and time-ordered. Resolution smearing is done. For the large volume central tracking chamber (CTC), the effect of finite signal propagation along the sense wires is simulated. Pulse length is simulated using the inclination of the track relative to the drift cell. Inefficiency, noise, and dead time are applied to the data, and the remaining hit times are formatted into the raw data bank. In parallel, the auxillary pointer banks linking raw data to/from the data list banks are generated. ADC banks are formed for chambers which have pad or charge-division readout.

FIGURE 4. FLOWCHART FOR DETSIM
Region Simulation

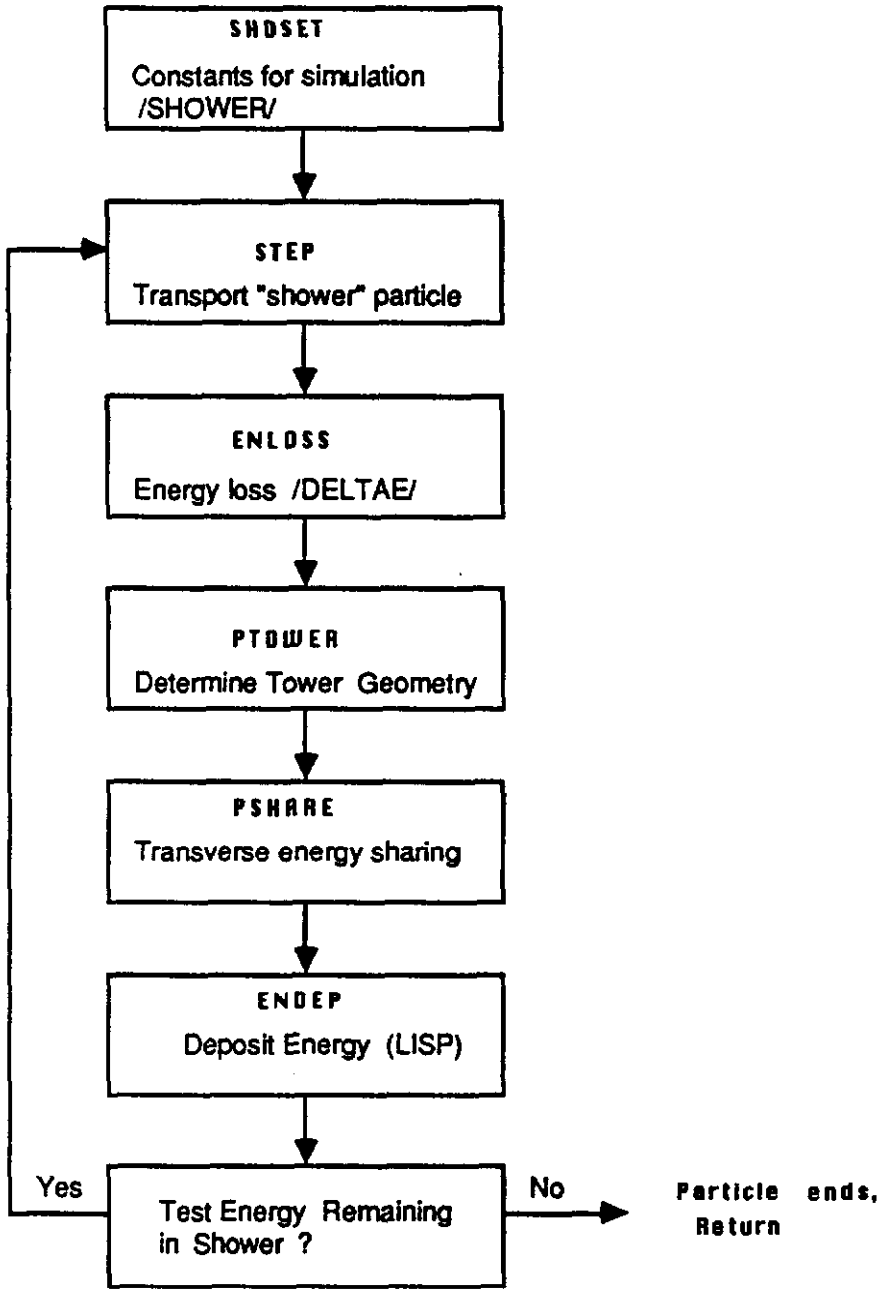


9. CALORIMETRY SIMULATION

As the particle is stepped through different detector elements, the number of radiation lengths and interaction lengths traversed is incremented. If the particle reaches a pre-determined depth of radiation/interaction lengths, it interacts. Interactions upstream of tracking devices generate a list of secondaries which are added to the list of particles to be simulated. Interactions inside of calorimetry are treated differently. There, a parameterized shower model is used. The centroid of the shower can be thought of as a neutral particle travelling in the same direction as the particle immediately before showering. This neutral "shower-particle" is stepped through the various detector elements exactly as a regular particle would be. As the stepping takes place, running totals of the number of grams/cm^2, radiation lengths, and interaction lengths traversed are kept. The shower development is parameterized in terms of these quantities.

After each step along the shower-particle's path, the shower particle undergoes energy loss by integrating the longitudinal shower profile over the number of interaction lengths and radiation lengths in the previous displacement. The energy calculated to be lost by the particle is stored in two parts: the electromagnetic energy loss; and the hadronic energy loss. (Electromagnetic showers have no hadronic energy content, while hadronic showers are a mixture of both.) If the current region is a calorimeter, the energy lost in the current step is deposited in the struck calorimetry tower and its 8 (or 24 depending on the ratio of transverse shower size to tower size at that depth) neighbor calorimetry towers. To do this transverse sharing of energy between pads, the tranverse shower development as a function of grams of material per square centimeter traversed since the shower start is calculated. Energy deposition into cracks is also simulated here, by integrating the transverse profile of the shower over the effective area of the crack. Statistical fluctuations are put into the simulation by calculating the effective number of minimum ionizing particles the deposited energy corresponds to, and applying Gaussian or Poisson fluctuations to that. Detector-dependent intrinsic EM/hadron response differences are simulated, as well as effects such as light attenuation in scintillator plastic. This process of stepping, energy loss, and energy sharing and deposition in calorimetry continues until the particle runs out of energy or exits the calorimeter. At this time, the tower-by-tower sums of energy, as well as timing information for towers instrumented by phototubes are stored in the data list bank for future use. If the shower exits from the CDF volume (i.e. leakage), a leakage record is added to the data list for that particle. In addition, the sum of energy lost by the particle in dead regions is also stored. Figure 5 shows the program flow for shower simulation. After the event is simulated, a final stage of processing the data list banks creates raw data banks, and

FIGURE 5. FLOWCHART FOR SHOWER SIMULATION



auxillary pointer banks.

A problem of the technique of shower modelling is that there is danger in extending the model to energies far from those measured by test beam. CDF was unable to find very low energy test beams (0.4 to 5 GeV) to calibrate its detectors in, so the response for these energies can only be estimated. There is an ongoing effort to use isolated charged tracks with momentum determined in the tracking chamber from real events to probe this region. This study, when completed, will give us confidence about the low energy parameterization of showers. A detailed description of the electromagnetic and hadronic shower models is presented in reference [3].

10. INPUT

Event records from the various event generators used by CDF (ISAJET, PYTHIA, MBR) are translated into a uniform event bank structure before simulation. The translation involves translating particle species ID's and the particle evolution history into CDF standard.

CDFSIM (and QFL) also allow the user to "short-circuit" the event generation, and directly simulate events of arbitrary topology that they themselves create. The "fake event" subroutine, called after the event generator created event is read into the program, allows the user to edit the original event, or totally replace it with a new one. This routine can be used to create events with well-defined properties, for instance, with a single electron of P_{total} = 100 GeV, at θ = 90 degrees, and ϕ = 0 degrees. The utility of creating specially tailored data banks for hardware/software debugging is obvious.

11. CREATED INFORMATION

The output of the simulation is a set of YBOS banks. User-set input flags to the simulation specify which of the possible YBOS banks are to be saved in the event record. There are four sorts of data that the simulation creates: System data; particle evolution history (the physical particles, daughters, and vertices) of the event; detector data (the "raw data" banks of the simulated detector); and "data history" , the data list banks, and pointer banks that link the data lists and the raw data banks.

A set of system data banks is produced for program control, and use in debugging. The run header bank contains all user-set input conditions to document what was actually asked of the simulation. The logical record bank contains the run/event number, and process creation code (i.e. simulation or real CDF data) and creation time. The event classification bank contains seeds for random number generators for the start of the event; the event weight; and other bookkeeping details.

The complete chain of parent-daughters, and associated vertices are stored in a particle bank, and a vertex bank. The particle bank contains: The total number of particles in the event; for each particle the momenta, mass, particle type, pointer to originating vertex, pointer to terminating vertex, path length travelled, and termination code (for example, decayed, showered, or punched through). The vertex bank contains: the total number of vertices of the event; and for each vertex, the X,Y,Z of the vertex, the time in nanoseconds since the primary interaction of the vertex creation, parent pointer, and daughter pointers.

The simulation creates (on demand) all raw data YBOS banks that can be read out by the data acquisition system. All banks have structure identical to the real CDF data.

As well as producing raw data banks, CDFSIM records a data history to help in analysis. The data list banks, as mentioned earlier, contain for each particle, the full set of hits and energy depositions made as the particle passed through the detector. There are also auxillary banks that link from raw data to the data lists, and from the data lists to raw data. Use of these banks allows complete knowledge about, for example, which drift chamber hits on a reconstructed track come from which particles, or how much energy in a calorimetry cluster is due to a given parton. These banks are invaluable for tuning pattern recognition programs, and for associating detector response to the underlying physics.

12. QFL

The QFL is designed to be extremely fast simulation of CDF that is used for initial physics studies to gain a feel for what is important, and what can or cannot work. Although it is new to the collaboration, it has become very popular because a rough answer can be reached in a few minutes of CPU time, rather than a more refined answer from CDFSIM, but requiring 100 times longer to reach. However, due to simplifying assumptions designed to speed up the program, it cannot be used in all cases. The simplified geometry of QFL is a cylindrical volume with uniform axial magnetic field, surrounded by spherical shells of Eelctromagnetic and hadronic calorimetry.

After randomly determining the Z of the primary event vertex, the QFL simulation of CDF tracks particles into the calorimetry. Particle rotation in the central solenoidal field is simulated. Particles may decay in flight, adding daughters to the particle list. After exitting the central tracking volume, the particle strikes the calorimeter (assumed to be a spherical shell with EM calorimeter 20 Xo thick, followed by a hadron calorimeter 5 λ_0 thick). The particle travels to its pre-thrown interaction point (the same code as CDFSIM) or punches through without interacting. At the interaction point, a shower parameterization is generated (the same longitudinal model as used

by CDFSIM, but a simplified triangle-function transverse profile). Energy deposition is calculated by integration of the longitudinal profile over the step (the same code as CDFSIM). Transverse sharing between towers is calculated by integrating the triangle function over the tower geometry. Cracks are simulated by attenuated energy deposition in that region, but no enhanced probability of punch-through (as CDFSIM does have). Gaussian fluctuations generate detector resolution. A global intrinsic EM/Hadron response ratio is assumed. Hadron response non-linearity due to increasing π_0 content as a function of energy is simulated (as it is in CDFSIM). Not simulated in QFL are hadronic or EM interactions that would generate additional particles in the particle list, or the effect of materials, like dE/dx or multiple scattering. After event simulation, the particle list is reformed into the particle and vertex banks (the same code as CDFSIM). Raw data for the various detectors is not created. Rather, for tracking chambers, track segment banks are formed. Performance of tracking and pattern recognition are simulated using the path length and angle of passage of the track through the chamber. The bank of solid-angle ordered energy depositions in the calorimetry is formed. QFL works very well for studies involving principally calorimetry (Jets, for instance). It does worse when tracking pattern recognition becomes important. Work is in progress to improve the tracking pattern recognition simulation, but since chamber hits are not simulated, it is not clear how accurate the program can be made.

References

1. D. Quarrie, YBOS Programmer's Reference Manual, CDF internal note 156.

2. C. Newman-Holmes and J. Freeman , A Fast Calorimetry Simulation for the SSC, this volume and FNAL/CONF/87-231

3. J. Freeman and A. Beretvas, A Short Review of the CDF Electromagnetic and Hadronic Shower Simulation, Snowmass SSC Proceedings, 1986.