MCFast: A Fast Simulation Package for Detector Design Studies

P. Avery [a], A. Boehnlein [b], L. Garren [b], R. Kutschke [b], P. Lebrun [b], M. Lohner [a], P.A. Kasper [b], P. McBride [b], K.L. Sterner [c], T.J. Wenaus [d], J. Yarba [b]

[a] *University of Florida, Gainesville, FL 32611*

[b] *Fermi National Accelerator Laboratory, Box 500 Batavia, IL 60510*

[c] *University of Pennsylvania, Philadelphia, PA 19104*

[d] *Lawrence Livermore National Laboratory, Livermore, CA 94551*

The Simulation Group at Fermilab has developed a fast simulation package for detector design studies. The goal of this package, called MCFast, is to provide a fast and flexible framework for the comparison of detector geometries and, in particular, to compare experiments designed to study the production and decay of B hadrons in a collider environment. The code is written primarily in Fortran and C and the fast tracking is based on the Kalman filter technique. A 3-D Graphics package has been developed to display the detector geometry, tracks and calorimeter hits. Recent updates to the MCFast package include improvements to the tracing, track fitting, calorimetry and graphics.

Key words: Fast Simulation

Mature HEP experiments usually have two Monte Carlo based tools for studying the response of their apparatus. One of these is a detailed GEANT[1] based simulation and the other is a fast, parameterized Monte Carlo which smears generated track parameters according to empirical formulae.

When designing a new HEP experiment, however, there is often a demand for a different sort of tool. Consider the design of a B physics experiment at a hadron collider. In such an experiment, it is necessary to suppress backgrounds which have a cross-section that is perhaps 10^6 times higher than that of the final state of interest. This suppression is achieved by demanding that the B candidate form a secondary vertex. In order to determine the background levels for one iteration of the design of such a detector, one must generate many millions of background events and then perform the tracking and vertexing for these events. This often means that a GEANT based simulation will be too slow to be used effectively in the early design stages. Moreover, it is during these early stages that good parameterizations of the resolution functions are not known. Therefore, neither of the traditional tools is well suited to this problem.

The original motivation behind MCFast [2] was to be the tool which fills the above gap. As the project evolved, the mandate has been expanded to include fast simulation of calorimetry and explicit hit generation. The "fast" in the name MCFast refers to three aspects of the program: there is a short learning curve for new users, the time required to modify a detector description is short and, of course, that the code executes quickly.

MCFast can be viewed as a simulation engine which takes two inputs, creates three outputs and then calls some user code. The two inputs are the detector description and the physics events, which are in STDHEP format. Two of the outputs are a list of raw hits and a list of reconstructed tracks; these are stored as Fortran common blocks and are available to the user code. The third output is a machine independent data file [3], which contains the detector geometry, the generated event information, the tracing points generated during the simulation, the raw hits and the reconstructed tracks. This file can be used as input to one of the MCFast graphics packages [3] or it can be read back in order to run updated user code without redoing the simulation.

One of the distinguishing features of MCFast is that the information about granularity, resolution and efficiency of each detector component is described in the detector description file. Moreover, no user code is required in the simulation step; instead, many of the details of the simulation can be controlled at the command file level. The two main hooks for user code are for trigger simulation and for physics analysis. Another feature of MCFast is that it can simulate multiple interactions per beam crossing.

One point glossed over in the above description is how the pattern recognition is done. The simulation engine knows which hits belong on which track and it simply feeds lists of hits and scattering surfaces to a Kalman [4,5] filter. This ensures that each track is smeared with a distribution which accurately reflects the distribution of hits and scattering material along its own trajectory. It is this detailed level of simulation which makes MCFast an excellent tool for studying vertex based triggering strategies and offline background rejection strategies. A plan for the simulation of errors in pattern recognition will be discussed later.

MCFast knows about many types of detector and scattering elements and a detector is made up of a hierarchical tree of these elements. The list of known elements includes cylindrical drift chambers, planar wire chambers, solenoidal magnets, dipole magnets, barrel silicon strip and pixel detectors, forward silicon strip and pixel detectors, and various calorimeters.

The heart of MCFast is the code which traces particles through the detector. This code has evolved in three steps. In version v2_5_x and earlier, MCFast traced particles along ideal trajectories, either helices or straight lines. No modifications were made to the trajectory to account for multiple scattering (MS), energy loss (ELOSS) or bremsstrahlung. The fit code, however, does include MS, using the TRKERR technique [6]. The output of this version is appropriate for studying questions about vertex, mass and momentum resolution. However it is not appropriate for studying detailed trigger and pattern recognition questions which expect deviations from ideal trajectories.

Starting with version v2_6_0, in December 1996, MCFast includes both energy loss and multiple scattering in the trace. The output of this version is suitable for use with realistic trigger and pattern recognition codes. In this version of MCFast, it is not possible to mix forward and central elements in one detector.

Starting with version v3_0_0, MCFast will have entirely new tracing code. This new version again allows mixed forward/central geometries and it implements MS, ELOSS and bremsstrahlung. The new code also follows the full length of tracks which curl up in the magnetic field. Moreover, this new code, which is written in C++, executes about twice as fast as the previous version. The code is now in an advanced testing stage and will be released in the summer of 1997.

The track fitting code is also undergoing an evolution. Until version v2_6_0, the MCFast track fitting code used an algorithm similar to that of TRKERR [6]. This Kalman filter based algorithm needs the relative positions of the measurements and the scattering surfaces, and the resolution at each measurement. But it does not use the values of the individual smeared measurements. This method does account for multiple scattering, even if the initial trace did not. The biggest drawback of this method is that it does not allow one to model errors in pattern recognition.

Starting with version v3_0_0, the fitting code will no longer use the TRKERR technique and will, instead, run a Kalman filter over the smeared hits. With this new code one can model both errors in pattern recognition and non-gaussian tails in the resolution function of each device. The plan for modeling errors in pattern recognition is to concentrate on local errors; that is, to search the neighbourhood of each hit on a track for hits from other sources and to sometimes pass nearby incorrect hits to the fitter. There are no plans to model more general sorts of errors.

The shower model [7] used inside MCFast is that the core of a shower will follow the same trajectory as would a noninteracting particle. The profile of the shower, centered on this core trajectory, is parameterized as a function of particle type and material type, as is the energy deposition in each cell. Again, these parameterizations are specified in the detector description file. Recent improvements to the shower model include, new shapes for absorbers and calorimeters and the implementation of hadronic showers. In addition, a shower which starts in one detector component is now followed into subsequent components.

The original MCFast 3D graphics package was Explorer based. An OpenInventor based package is now available [8].

For six months MCFast has been used by the C0/BTeV and D0 collaborations and, with this experience, the code has become quite robust. Several other groups have also inquired about MCFast. The following statistics were obtained using MCFast v2_5_2 on a DEC ALPHA EV5 250 MHz processor. The simulation time for $p\bar{p} \to B_s X$ at $E_{CM} = 2$ TeV, was 0.29 s/event using a working design of the C0/BTeV tracking system (no calorimetry). Using a working design of the D0 Run II detector, the simulation time was 0.23(2.0) s/event without(with) calorimetry enabled. For generic soft events the times were 0.18, 0.15 and 1.0 s/event, respectively.

In conclusion, the purpose of MCFast is to give fast and accurate assessments of candidate detector designs. Robust code is now in use by two collaborations. Finally, MCFast is not meant as a replacement for more detailed simulation packages, such as GEANT, but offers an alternative for studying the many processes for which the MCFast techniques are applicable.

References

[1] R. Brun et al., CERN/DD/EE/84-11.

[2] http://fnpspa.fnal.gov/mcfast.html

[3] A. Boehnlein, "MCFAST, A Parameterized Monte Carlo for Detector Studies", Proceedings of CHEP '95, Ed. R. Shellard, T. Nguyen, World Scientific, 1996.

[4] P. Billoir, NIM **225**, 352-366 (1984).

[5] P. Avery http://www.phys.ufl.edu/ avery/fitting.html, Fitting Theory V.

[6] http: //heplibw3.slac.stanford.edu/FIND/FREEHEP/NAME/TRACKERR/FULL

[7] J. Yarba, http://fnpspa.fnal.gov/mcfast/mcfast_docs.html, User's Guide for Calorimetry in MCFast.

[8] P. Lebrun, talk a419 in these Proceedings.