BRAHMS Software Note

SONATA

A detector simulation and tracking program for BRAHMS.

Flemming Videbaek
Physics Department
Brookhaven National Laboratory
October 23 1997

Disclaimer: Since SONATA will not be developed further, expect this explanation to be incomplete; it though contain information relevant to the methods for simulations , tracking and data formats. Also sonata formed the basis for the SONATA++.

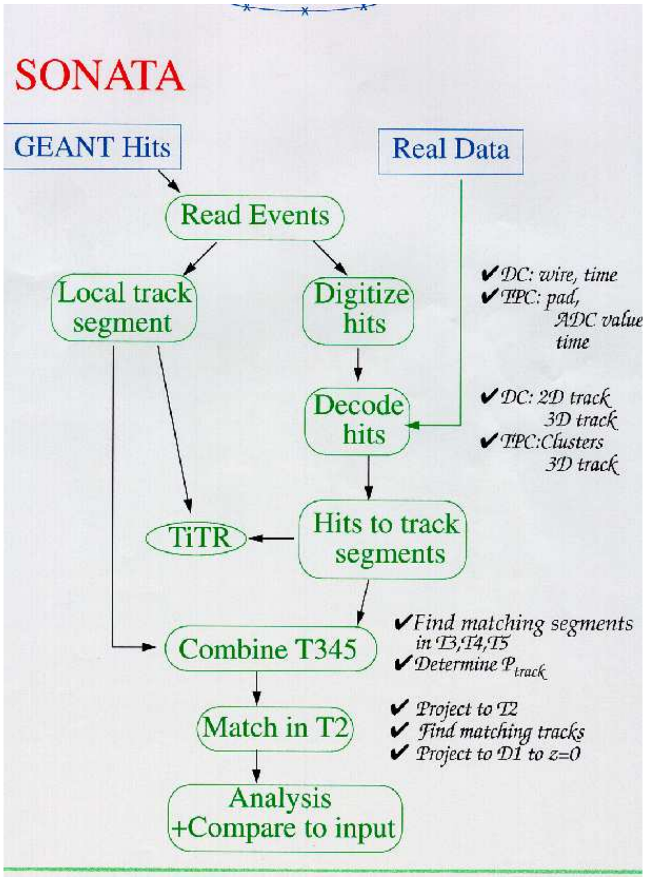
Background

The *sonata* program has been used to develop the detectors , explore and specify their properties for the BRAHMS experiment. It was build as an integrated package for reading output from the BRAHMS GEANT simulation in the form of hits on detector volumes and hits, digitizing the hits for tracking detectors , decoding and performing pattern recognition, making track segments, and combining tracks through the complete spectrometer system. Simple analysis on the tracking, and comparisons to the input could also be performed in a separate section of the program. The purpose of the program was to demonstrate the feasibility of the tracking in the spectrometer, and the quality thereof. It could also serve as a prototype for actual tracking, but was not optimized for efficiency, and has only rudimentary control features. Because it was to be used in a development phase it was decided to make use of the rather flexible data structures and tools (i.e. subroutine calls) as given by the ADAMO implementation of the entity-relation model. To fully understand the code of SONATA it is necessary to have access to the ADAMO documentation, available from CERN. (See eg. The BRAHMS software web pages or use the URL http://www.cern.ch/Adamo/ADAMO_ENTRY.html directly). This table based package is not being updated any more by ALEPH, and thus do not run on platforms other than the SGI IRIX version 5 and AIX .

This document introduces the some ideas of the algorithms, the data flow, and gives hints how to run the program. It is of course expected that SONATA will become obsolete, but it will still be used some for benchmarking and for comparing with the SONATA++ code which is under development.

Data Flow

The program goes though the following steps. The steps are graphically described by the figure from JHLee, and explained in the following sections.



- During the setup phase the user configures the detectors, reads the geometry definition i.e. the position and sizes of detector, the position and sizes of the magnets, and the magnetic field values. The detector configuration consists in defining the type of detector i.e. TPC or Drift Chamber, the number of pad rows, wire-planes, wire angles etc. In the setup phase the program also reads tracking parameters i.e. intrinsic resolutions, cut parameters from either files or commands.

- Following the setup the program will analyze one event at a time. Flags can be set to control the flow. This main pass of the program involves the following steps

  1. The events are read from an input file. Presently two different formats exist for the input a) an ASCII format or b) binary stream file. The latter method is a first step towards a uniform output from the GBRAHMS which can be used to generate Ntuples, ROOT files, and input for SONATA. In both cases the

output file is generated from the GRAHMS using the COMIS routines called after the simulation of each sub-event in GEANT.

2. For each detector volume the code finds the hits. The hits from GBRAHMS are stored as the entrance and exit positions in a larger volume. The segmentation into pad-rows, wire-planes is determined from the detector parameters in SONATA. Thus, when a more detailed description of a detector e.g. in terms of sub-modules as possible for T3-T5 a different description has to be found.

3. For each detector component the digitization is performed. All parameters which determines the outcome of the digitization can be set by KUIP commands. This are e.g. drift velocities, intrinsic resolutions, two-particle resolution. For further details see the help command in SONATA.

4. The next step involves generating local tracks for each detector i.e. T1, T2, T3.,... The method used depend on the detector specification. Essentially two methods are possible.

    - A pad TPC with a given number of rows and pads instrumented. The algorithm roughly follows the steps.

        - Convert for each pad row the time samples into sub-cluster.
        - sub clusters from different pads in the same row are assembled into continuous islands (connected clusters). An island consists of one or more track clusters.
        - For islands which do not match the characteristics of a single cluster i.e. in terms of numbers of samples, variances in pad and time direction, an 2-dimendional search is performed. It looks first for the highest point, exploring the mountain around the peak until a valley is found. The depth of the valley relative to neighboring pads are used as an indicator for terminating the search for a track cluster. The search is then repeated for the un-used samples of the island. Such a search can result in one or many track cluster candidates from a given island.
        - For all track cluster both isolated as well as those from the cluster search space points are calculated using the positions and drift velocities.
        - Local tracks are found by several passes over the pad-rows using the “follow your nose” algorithm, as e.g. described in Chakers thesis (Strasbourg)

    - Drift Chambers. This is a fairly conventional algorithm. It is designed for drift chamber which has several sets planes in two main views (i.e. x and y) and two alternate views (u,v).

        - For all staggered pairs of planes in the main views, the algorithm attempts to find and identify left-right pairs of positions belonging to a single hit. If two hits from consecutive planes match they are combined into one hit on the median position and the other ambiguous hit positions are dropped from the list of possible hit positions (in each view).
        - 2-d tracks are found in each of the main views by looping over all possible hit position candidates and requiring that a tracks has hits in all planes, except for a small number (given as input input parameter)
        - Having all possible 2d tracks in the main view 3d candidates are generated combinatoricaly. These tracks are then compared with hits in the alternate views and matching hits are required on most planes to be considered valid.
        - This procedure quite often generates many ghost tracks. Therefore a cleanup procedure is applied which lumps tracks which shares many hits into one class (subset) and selects the one with the lowest chi-square as the candidate track.
    - The result of this step is thus local tracks for all active detectors. Each track is specified by a reduced set of parameters, essentially position in middle of chamber and direction cosines as parameters.

5. Track matching for tracks before and after all magnets are the done e.g. t4 and t5 through D4 etc. The magnet are treated in the effective edge approximation and matched in the middle of the spectrometer in x, y and y-direction. If a match is found the momentum is calculated.
6. Tracks through t3-t4-t5 is then formed, the momentum recalculated, and the track projected back to T2 to pickup a matching local track. If found a valid global track is found and project back through the spectrometer system to the interaction region. Notice that this procedure does not require that tracks has been found in T1. If they can be found then this could also be used. This is though not implemented.
7. Finally the analysis section is called. This can among other things compared found tracks to input tracks, determine quality etc. All result are in results of HBOOK histograms. This part is not well structured, and quite a few parameters have to be set. Results appear in many HBOOK subdirectories organized according to detector, detector type etc. You will have to look in the analysis module. Also some histograms are filled during tracking to monitor performance.

DATA Input formats

The SONATA program presently takes input in two different formats. This format is generated either by the GEANT analysis routine executed on each event, or by a routine reading Ntuple output from the GEANT analysis.

1. ASCII format. The information on hits in a given detector is stored as a list of hit and track information. This is a strictly formatted file.

2. Stream i/o file. The hit and track information is written as binary structures. The format is quite compact. In addition to be readable by SONATA these files can furthermore be read by a ROOT program, written by Ziping, for further analysis in the ROOT frame work. Available is also a small C++ class which uses CLHEP classes, which can read and store the structure in lists. An event dump program gb2a which can display the contents of the files exists. The files are not transportable between all architectures as today.

Acknowledgements

Sonata has been developed over many years. The early implementation of the drift chamber algorithms was done by Shige Hayashi. Ideas and algorithms to the TPC pad digitization comes from Dieter Roehrich.