UCI-Neutrino No. 90-9
SNO-STR 90-80
18 June 1990

Identifying Tasks for the Data Analysis Software Package

Gerhard Bühler
University of California, Irvine, CA 92717

Abstract

The design and implementation of the Data Analysis software for SNO will be done in a distributed way. This makes it necessary to identify and define tasks that can be interlinked to form the final software product.

As a first step towards this end, the two existing MC codes have been distributed to potential contributors to the software in an effort to provide an educational overview of their components. This report outlines the essential components of the Data Analysis software with a focus on the Monte Carlo package since this package already contains the essential features of an analysis package. One should be able to "trace" the features outlined here within the existing codes in order to get a better feeling what is involved in designing the code.

1 Introduction

The software development effort has been discussed in several SNO documents, and has been summarized in SNO-STR 107-89. This report deals with the task of itemizing the Data Analysis package in way that distributed code development is made feasible.

One of the essential features of the analysis package is that it will provide the collaboration with a well-documented, modular and centrally managed core code common to all collaborating institutions. Individual analyses and "home-made" software will be easily implemented within this structure owing to the modularity of the code. This feature will also allow upgrading the software as technology, such as with graphics, progresses. Providing the global code package to the collaboration does not eliminate individual efforts, e.g. for event reconstruction, statistical analysis, modeling etc. On the contrary, using the tools of central code management these efforts should be incorporated into the global framework and provided to the whole collaboration if they are found to improve or better facilitate tasks.

However, the central effort ahead is to clearly define the software package and to establish the overall task definition, the sub-packages to farm out to individual contributors and the links between those sub-packages. A first stab at this is done in this report.

2 Overview

The following sections identify packages which can be made to form autark entities within the Data Analysis software package. The order in which they appear is in a large part chronological for an event model, i.e.:

- Detector definition
- Event simulation
- Photon transportation
- PMT response
- Event reconstruction

- Data storage

The modular design of the software package is important for maintenance and code improvements. On the other hand, the modular design needs a clear and precise definition of the common variables and the handshaking between different parts of the code.

3 Detector definition

The detector definition is laying the ground for many of the subsequent packages. The interaction generator (EGS, GEANT) and the photon transportation code in particular need to have access to the definitions spelled out here.

3.1 Geometry

The geometry and the structures of the detector require some attention during definition. The geometry definition is straightforward for a concentric shell model of the detector, but gets more difficult with overlapping structures (such as an acrylic/steel collar around the acrylic vessel, neutral current rods in the D_2O, suspension cables for the acrylic and so on). Parameterizing of quantities is essential to allow for accomodation of late changes. The geometry definition directly interacts with the interaction generator and the photon transportation code. It also includes the definition of the photosensors, PMTs and reflectors, within the detector structure.

3.2 Materials

The materials going into the detector have to be defined with their chemical composition, their location within the detector geometry and their physical properties. The actual implementation depends on the necessities of the code which needs this information as input.

4 Event simulation

The event simulation is seen here as the part of the code where the initial vertex is placed and the physics is initiated, up to the point where some energy has been deposited into the detector within some material. After this point, the photon generation and transportation takes over.

4.1 Signal definition

This part of the MC code generates initial parameters such as vertex, momentum and particle properties according to the user's request, including generation of multiple particle events. The signal is defined here as the physics quantity of interest, in contrast to backgrounds (which are, of course, interesting, too). The initial parameters may have to be chosen from spectral information in a random manner. Charged current, elastic scattering, and neutral current reactions will have to be based on a variety of primary sources, such as ^{8}B neutrinos and *hep* neutrinos.

4.2 Background definition

Arbitrarily, the distinction is made here between the signals (cf. Section 4.1) and backgrounds. Here also the vertex, momentum and particle properties have to be chosen, from primary sources such as the beta decay of ^{232}Th and ^{238}U , high energy gamma sources and muons.

4.3 Primary interaction generator

The initial parameters are handed over to a generator package such as EGS or GEANT which then calculates the physics interactions of the primary particle(s), determines the energy loss and determines the subsequent geometric location of the particle(s). Information is needed about boundaries within the detector from the geometry package.

5 Photon transportation

The photon transportation code is taken here to start at the energy deposition as defined by the interaction generator, and leads via light generation (number of photons, frequency distribution, direction) and attenuation processes to a terminal point for the photon. The termination can be either extinction, leaving the sensitive detector volume, or hitting a photosensor. It has been proven useful to establish tally counters which account for the photon generation processes, the histories and final outcome of the transportation process.

5.1 Light generator

The energy deposition delivered by the interaction generator needs to be converted here into either Čerenkov light or, in case of yet to be decided neutral current detectors, scintillation light. The light generator selects the direction and frequency of the photon and, in the case of Čerenkov light, its polarization. A decision about whether or not to transport the photon is then made according to the overall sensitivity of the photosensors, and the tally counters are updated.

5.2 Attenuation processes

The photon is transported throught the detector by taking into account various attenuation processes: Absorption, scattering and reflection.

5.2.1 Absorption

Light absorption can occur in all materials of the SNO detector. A decision to absorb the photon over a given distance is made based on some absorption parameter as a function of light frequency. In the event of an absorption, the tally counters are updated.

5.2.2 Scattering

Light scattering affects the direction of the photon and, in a rigorous treatment, its polarization. A decision is made whether or not to scatter the photon over a given distance and if so, where the scattering will occur. The new direction and location have to be fed back into the transportation code and the tally counters have to be updated.

5.3 Reflection

Light reflection/refraction will occur at all interfaces or boundaries within the detector. The reflection/refraction results in a new direction and location of the photon and potentially a changed polarization vector. The new parameters have to be fed back into the transportation code and the tally counters have to be updated.

6 PMT response

The PMT response code starts where the arrival of a photon on a photosensor has been flagged. For sensitivity enhancement structures such as reflectors, the photon has to step into another transportation phase before a decision is made whether it has been seen by the PMT or whether it is terminated elsewhere. If the photon leaves the photosensor going back into the system, the new direction and location have to be fed back into the transportation code and the tally counters have to be updated. If the photon is seen by the PMT, the signal data of location and arrival time have to be registered.

A potential switch can be build in here to decide whether to follow individual photons or whether to work with average values for acceptance and reflectance for the photosensors. The latter would speed up the simulation process immensely.

6.1 PMT/reflector definition

The PMT/reflector definition is different from the above mentioned geometry definition of the photosensors within the detector geometry. Here, the geometry of the surfaces of PMT and reflectors have to be defined along with their physical properties, in order to correctly account for absorption, scattering and reflection within these structures.

6.2 Photon path

The photon path within the photosensor needs to take into account the same processes as outlined in Section 5.

6.3 Signal and data generation

After a PMT has seen the photon, the signal data (location and relative time) have to be logged and passed on to the event reconstruction part of the code.

The data format into which the data have to transferred should be identical with the format by which the DAQ hardware writes data.

7 Event reconstruction

In a simple minded way, all the event reconstruction has to do is to determine the vertex, direction and time of occurrence of an event in the detector, given the raw input data from the PMTs. The task may become involved for combined occurrences of scintillation and Čerenkov light. The event reconstruction is one of the central tasks of the Data Analysis package and a continuing development of this part of the code is anticipated throughout the course of the experiment. This means that concurrent versions of the reconstruction code have to be allowed for as indicated in Section 1. This makes it the more important to have a clear cut definition for the input/output interfaces to the rest of the code.

8 Data storage

The data storage is the final stage of an event simulation (if one does not count interpretation and collaboration meetings). The data to be stored have to be defined, since there are not only the condensed information on reconstruction but also auxiliary information incurred during the data analysis process itself. The storage itself and the organization of histograms have to be defined, and a set of "standard" histograms should be provided to users as part of the global code. Since most of the simulated data will be kept, a DST (for Data Summary Tape) definition must be included as well.

8.1 Storage definition

The storage definition is in some respect dependent on the dynamic data management system used. With ZEBRA, the structure of *stores* and *banks* has to be defined in a suitable way. The orientation here should be towards existing experiments to avoid double-learning processes and large and time consuming overheads.

8.2 Histogram organization

Histograms and graphics are likely to change over time and it is useful to concentrate the initialization and filling of histograms in one unit. In this way, any changes in the definition are easily accessible and are not spread out over many code modules. A clearly defined histogram organization (such as for general information on the event generation, for events before and after reconstruction, for analysis parameters and so on) will make exchange of data between institutions easily feasible.

8.3 DST definition

The Data Summary Tape (DST) format needs to be defined with much the same concept in mind as the histograms. For both, the use of dynamic management systems are invaluable since the change in definition does not have to render older versions obsolete.

> It should be pointed out that defining the histogram or DST structure or organization does not say anything about the *contents*. The organization is the key to common access and exchange of data, where the actual contents of the histograms are clearly defined by the user. As an example, using an HBOOK structure (based on ZEBRA) which allows for the definition of directories for histogram definitions, defines the organization of the histograms. Whether a user puts garbage or Nobel material into this structure is up to h$^{im}_{er}$.
>
> This does not affect the global definition of "standard" histograms provided as a service to all users of the code.

9 User interface

Last, but not least, the user interface has to be designed such that it makes the access to the code independent of a detailed knowledge of the code idiosynchrasies. The determination of initial parameters in a batch or interactive query session as well as the logging of messages have to be made robust (or foolproof, if you prefer) and safe. Frequent checks on input consistency are