TRS Design Proposal

Brian Lasiuk, Thomas Ullrich
Yale University, Physics Department

May 4, 1998

1 Introduction

This document describes a preliminary design of the new TpcResponseSimulator (TRS). It is divided in two parts. In the first section we concentrate on a more technical view describing how we intend to model the processes in order to approach a correct and fast description of the TPC response. In the second section we present an first attempt of a class design which reflects the approach described.

The current attempts to factorize the simulation into several components that are more or less, independent of one another. For example, the transport of ionization to the sense wires requires information about the gas used and properties of the field cage. The amplification and signal development processes depend on details of the wire grid geometries and voltages on these wires. The digitization depends on detailed properties of the electronics. These *categories* have been described elsewhere [1]. This should allow a better isolation of problems that require study. Also, by varying the degree of detail in the description of these different processes it will allow the simulation to be done in several different modes with varying degrees of accuracy and speed:

Slow mode (SM): Simulation is performed as accurate as possible and at the single electron level in the extreme. Processing time plays minor role since this mode is not intended to be used for large scale production but rather for testing cluster related algorithm and providing useful means to understand the TPC response in detail. The current equivalent pam is tss. *Moderate priority. Will provide valuable testing for parameterizations to be used in faster versions.*

Fast mode (FM) Still accurate but with the emphasis on fast processing time. This mode is intended to be used for mass MC production. It is mainly based on a parametrisations which also allow fine tuning of the response without understanding the origin of certain defects. No current equivalent pam. *High priority*.

Hit mode (HM) Similar to the fast mode but does not generate pad but hit objects. This is useful for very fast simulations to test large scale effects like acceptance. This will compromises the level of accuracy, but will allow for fast calculations. The current equivalent pam is tfs. *Low priority*.

It turned out to be rather difficult to incorporate all three modes in one (simple) model. The current design is a first attempt to achieve *this* goal. Several prototypes have been designed written and rejected because of the lack of generality. It is highly favourable to unify all three modes in one package in order to minimize the overall code and to reuse already existing functionality. Having to maintain (update, correct bugs) of one package instead of three is highly desirable.

Much work has already be invested in the creation of a STAR Class Library (SCL) which encapsulates components of CLHEP [2] as well as other functionality and will be heavily used here. It is described elsewhere (see [3]). Also following the direction of CERN [4], no use of any RogueWave libraries (i.e. *Tools.h++*) will be used.

2 Technical View

We assume that input to TRS consists of a list (table) of `g2t_tpc_hit` objects each representing a segment of charge deposited by a track. The object has three important attributes:

- the total charge of the segment (dE)
- the path length of the segment (dl)
- the position or coordinate of the mid-point of the segment ($\vec{x} = (x, y, z)$)

The following describes the technical method of the response simulation in several steps.

1. the coordinate system is rotated such that the wires run along the x axis for each segment.

2. The charge segment is divided in N **subsegments**. Helical decomposition of the segment is foreseen. Each of the new objects is represents by a position $\vec{x}_i$ and a charge dE_i. Information about the length of the segments is neglected from here on. The number of division depends strongly on the mode. In the slow mode one could even envision to divide the segment up to the electron level or close to that. In the fast mode however N of 2-5 (may be even 1) may be sufficient. Tests have to be made to pin down a reasonable number.

3. Each charge subsegment is transported from the center of the segment at the point of production to the sense wire grid taking into account the effect of the presence of an inhomogeneous electric field and a magnetic field as well as charge loss through absorption. Transmission of the charge through the gating grid will be accounted for based on the voltages on the series of wire grids. The transport results in a new position $\vec{x'}_i$ and possibly a modified charge dE'_i. The detail of the transportation, i.e. processes taken into account, depends considerably on the mode:

   Slow mode: $\vec{E} \times \vec{B}$ effects, detailed diffusion models, effects due to T fluctuations, space charge, grid transmission, etc.

   Fast mode: The coordinates are simply projected along z with parameterized distortions (lookup table?).

   It is important to note that the conventional way of taking into account track crossing angles by increasing the width of the charge distribution in a parameterized pad response function is not followed here. Instead, effects of diffusion will be dealt with in the broadening of the charge cloud in the transportation of the electrons to the sense-wire plane.

4. In the next step the wire structure of the readout chambers is incorporated. To do so the individual subsegments with positions x'_i, y'_i (z'_i) are assigned to the closest wire. The charge deposited on the wire is represented by the center of gravity $\hat{x'}_i, \hat{y'}_i$ ($\hat{z'}_i$) on the wire. In order to achieve this in minimal time, the following procedure is envisioned:

   (a) The wires are represented by a simple (i.e. equidistant) histogram which can be implemented in the form of a smart vector. Each bin represents a wire k (i.e. $\hat{y}_k$) , or region in y, in which the charge is collected (i.e. the wire spacing). The histogram is scaled such that a simple typecast `(int)` y'_i yields the resulting bin k.

(b) For every bin the x'_i and z'_i values of the subsegments falling in this bin are summed such that the mean position along the wire $\hat{x}_k$, and in time (i.e. $\hat{z}_k$) can be computed.

(c) After the whole **segment** is processed $\hat{y}_k$ is given (or can easily be determined) by the bin number. $\hat{x}_k$ is obtained by $\hat{x}_k = \sum x'_i/n$ in this bin, where n is the number of entries. $\hat{z}_k$ is obtained in the same way.

(d) There must be a mechanism to keep track of the lowest and highest filled bin for an efficient search and reset of the histogram.

After this step the whole charge segment given by the `g2t_tpc_hit` structure is split on $k = 1 \ldots M$ wires. For each wire the charge, the wire position in $\vec{y}$, and the center of the charge along $\vec{x}$ and $\vec{z}$ is known.

5. Once the drifted electron (or clouds of electrons) are at the wires, the amplification procedure can occur. This can be done by scaling each bin by a factor—either a fixed constant or a number derived from a Polya distribution depending on the simulation mode. In this manner several different amplification effects, e.g reduced wire gain on 1. . .n wires, gas gain variation, etc. can be modelled. Also $\hat{z}_k$ (time) can be modified if needed to account for the effect of electrons arriving over a distributed time window (i.e. not directly on the wire). Again the detail depends on the simulation mode although the underlying idea is very similar for all cases.

6. For each cluster of electrons on the sense wire *k*, the centroid in the wire direction $\hat{x}_k$ and the centroid in the time direction $\hat{z}_k$ are known along with the total charge (Q_k). The spread (σ) of the electron cloud in the $\vec{x}_k$ and $\vec{z}_k$ directions can be calculated either from the individual electron positions within the cloud (SM), or by knowing the transverse diffusion coefficient (σ_T) and drift length of the charge cloud (FM). Using this knowledge the charge signal which is induced on the pad-plane can be generated as a function of the pad-plane coordinates. For each wire *k* the distribution will be generated in the direction parallel to the pad row (i.e. $\vec{x}$), and perpendicular to the pad row (i.e. $\vec{y}$). Assuming the response to be Gaussian (however any distribution can be specified) we have in the pad direction:

$$Q_k(x) = \frac{Q_k}{\sqrt{2\pi}\sigma_x} e^{-(x-\hat{x}_k)^2/(2\sigma_x^2)} \tag{1}$$

where $\sigma_x^2 = \sigma_{PRF_1}^2 + \sigma_{\hat{x}}^2$ Because there are multiple wires per pad, the signal generated by a wire can contribute to adjacent pads. In order to take such an effect into account, the extension of the signal must be generated in the $\vec{y}$ direction as well. This can be done by convoluting the charge distribution given by equation 1 with:

$$F(y) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{-(y-y_i)^2/(2\sigma_y^2)} \tag{2}$$

where $\sigma_y^2 = \sigma_{PRF_2}^2$.[1] Because $F(y)$ is normalized to 1 it does not alter the integral of the charge distribution but only distributes it in the xy-plane. Thus the expression for the charge distribution projected on the pad plane from the k^{th} wire is:

$$Q(x,y)_k = \frac{Q_k}{2\pi\sigma_x\sigma_y} e^{-(x-\hat{x}_k)^2/(2\sigma_x^2)} e^{-(y-y_k)^2/(2\sigma_y^2)} \tag{3}$$

The quantity which must be extracted from equation 3 is the total charge distributed on the j^{th} pad and is proportional[2] to:

$$Q_{pad_j} = \sum_k^{wires} \int_{pad_j} dy\ dx \frac{Q_k}{2\pi\sigma_x\sigma_y} e^{-(x_j-\hat{x}_k)^2/(2\sigma_x^2)} e^{-(y-y_k)^2/(2\sigma_y^2)} \tag{4}$$

In the fast simulator mode it could be envisioned to take only the value of the distribution at the centroid of the pad instead of the integral as this is expected to be very CPU intensive. In the slow mode the integral may be explicitly calculated for each adjacent pad in a defined area. It is also possible to use an analytically exact solution. For example, it is possible to calculate the amount of charge induced on a grounded plane by a point (or line) charge located a distance *d* above it. If the induced charge on a neighbourhood of pads is calculated for a specific granularity of electron positions on the sense wires and entered in a table, the total signal induced on any pad could be obtained by summing all signals that each electron (or cloud of electrons) produce. This is in principle the most accurate way to produce simulated signals on a cathode pad chamber. Such a procedure can probably be used only for simulating a small number of tracks, however these results can be used to check the more general analytical expressions (i.e. equations 1–4) used for the pad response functions used in the faster modes.

[1] Note that in general $\sigma_{PRF_1} = \sigma_{PRF_2}$.

[2] The coupling between the sense wires and pads is in general not 100%.

7. After the total amount of charge on each pad is known, it must be discretized in the time direction which involves properties of the analog electrons. The sampling of the signal as a function of time simulates the behavior of the pre-amplifier/shaper and the Switched Capacitor Array (SCA). The size of the signal (i.e. pulse height) is dependent on the gain of the electronics which should be available from a calibration data base and the clock frequency of the SCA which determines how much charge is collected in a single time bin.

   From equation 4, the amount of charge on any single pad can be deduced. In order to distribute this in the time direction information regarding the shaping time of the pre-amplifier is necessary. The response of a one-stage differentiation and two stage integrator shaper is given in functional form as [5]:

$$s(t) = \Theta(t)(\frac{t}{\tau})^2 exp(-\frac{t}{\tau}) \tag{5}$$

   where $\Theta(t)$ is the Heaviside function and τ is the shaping time of the pre-amplifier. The convolution of equations 4 and 5 provides a way to distribute the charge in time. This is depicted schematically in figure 1. This is of course a statistical convolution over many signals. In the slow mode, the effects of single electron statistics will make this signal discontinuous. This will be modelled by the inclusion of noise to give the signal a more realistic profile. It is foreseen to parameterize the amount and amplitude of the noise as a function of track crossing angles.

   With this function and a knowledge of the sampling (clock) frequency of the SCA, it is possible to distribute the charge between time buckets (δt) either by integrating equation between the appropriate time limits (SM) or assigning the value of the function in the middle of the time bin (FM). Along with each time bin being assigned a fraction of the charge, the appropriate undershoot (or overshoot) can be added to later time bins. This is also an appropriate place to add electronics noise that is not associated with the digital part of the circuit. All charge segments in a localized area (i.e. sector) due to close tracks (or hits) in the area must be generated before the digitization stage as the charge collected per time bin must be done at the analog level.

8. Finally the signals in each time bin can be digitized incorporating any information the calibration data base may contain about the ADC such as stability, linearity, etc. After the digitized signal is generated it must be added to

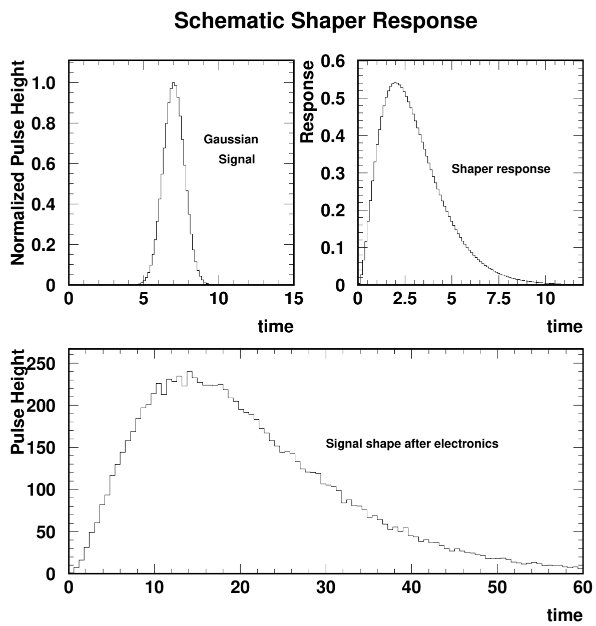

Figure 1: Shown in the bottom panel is the effect of the shaper on a Gaussian signal. This is a superposition of many signals in the absence of noise.

a pedestal for each ADC channel. Before the information for the sector is written out, the pedestals will be subtracted and the data may also be zero-suppressed at this level.

3 Class Model

A schematic of the UML class diagram of TRS is shown in Appendix A. It describes the classes that have been identified in the analysis of the project. Listed below is a brief description of each class.

- *Administration type classes:*

  1. **StTrsManager** is responsible for the administration and memory management of the program. It also stores flags and details of the simulation should require such as the number of subsegments to break each `g2t_tpc_hit` segment into.

- *Simulator classes:*

    1. **StTrsChargeSegment** a `g2t_tpc_hit` object.
    2. **StTrsMiniChargeSegment** a portion of a decomposed *StTrsCharge-Segment*. In the extreme limit it is a single electron.
    3. **StTrsChargeTransporter** is an abstract class whose specifications contains methods to transport the charge depending on the detail of the simulation:

        StTrsFastChargeTransporter uses a parameterized drift map in a lookup table.

        StTrsSlowChargeTransporter transport incorporating the calculation of the drift velocity at each point on a grid of a specific granularity and following the trajectory of each `StTrsMiniChargeSegment`.

    4. **StTrsWireHistogram** is a container which keeps track of the position of each charge segment at the sense wire plane.
    5. **StTrsWireHistogramBin** is a smart vector which is capable of keeping track of the mean and variance of a series of positions
    6. **StTrsAnalogSignalGenerator** Projects the charge accumulated on a wire onto the pad plane.
    7. **StTrsElectronicSignalGenerator** Samples the charge induced on the padplane as a function of time and distributes it into time bins.
    8. **StTrsSector** Contains all timebins with non-zero charge for a single sector. The *StTrsElectronicSignalGenerator* operates on this object to digitize the charge to be written out.

- *DataBase classes:* provide a layer to screen users from the actual implementation. These interfaces provide a means to obtain information that will be stored in the official STAR database.

    1. **StTrsGeometry** provides access to geometrical information regarding of the detector including the the pad plane layout, wire geometry, field cage, etc.
    2. **StTrsGas** provides access to properties of the gas like ionization potential, drift velocity, saturation properties, etc.

3. **StTrsMagneticField** provides access to $\vec{B}$ at any point $\vec{x}$.
4. **StTrsElectronicsCalibration** provides access to gain of individual channels as well as a list of dead channels.

Although this provides a general description of how processes are to be modelled, there is another aspect which is not obvious from such diagrams and that is the *Activity Diagram* or how and what information is passed to each class. This is perhaps best done by text:

1. the main program will set up the *StTrsManager* which will administrate the whole simulation. An *StTrsReader* with an associated file name will be created and assigned to the manager as will an *StTrsChargeTransporter*, an *StTrsWireHistogram*, an *StTrsAnalogSignalGenerator*, an *StTrsElectronicSignalGenerator*, an *StTrsSector*, and an *StTrsWriter* with an associated output file name. Along with the creation of each of these objects the database(s) will be initialized for calling. Any set-up parameters for the simulation that must be communicated between the separate objects will also be stored by the manager and must be specified before entering the main run loop. Examples of such parameters include the magnetic field configuration, level of simulation (i.e. slow or fast), number of events, current sector being processed, current track being processed,[3] etc. After the set up has been specified the manager calls a member function which initiates the simulation.

2. `processEvent()` is a loop over the number of events. The first procedure is to read an event which is handled by the reader. This should simply load a list of `g2t_tpc_hit` structures. These can be transformed into *StTrsChargeSegments* and sorted such that they are grouped by sector number which can be processed sequentially. It is important to note that all three-vector quantities such as position and momentum will be stored in an **StThreeVector** which is a component of the Star C++ Class Library (SCL) [3].

3. `processSector()` At the sector level there is a subset of charge segments to be processed. All the segments within a given sector are looped over at this level.

[3] This is very important because this information must be available to check if the track has been reconstructed.

4. `processSegment()` involves several steps:

   (a) the charge segment must be rotated into the local coordinate system. This is taken as the sector which has the wire direction in the $\vec{x}$ direction and increasing pad row number in the $+\vec{y}$ direction. This will be done with the *StMatrix* class from the SCL.

   (b) the charge segment is split into a number of subsegments (i.e. *StTrsMiniChargeSegments*) which are filled into a vector. The number of subsegments depends on a parameter (i.e. mNumberOf-Electrons) which is set in the manager. In the extreme slow mode this can be at the electron level.

      i. `processMiniSegment()` is a loop over each mini segment.

      ii. Looping over the *StTrsMiniChargeSegment* each is transported through the drift volume of the chamber to the sense wire plane. This process will consist of the modelling of the electrostatics of the field cage, the probability of charge loss due to attachment, and transmission through the wire (gating) grid. The result of this process will be an entry in the *StTrsWireHistogram* which keeps a record of the number of electrons that reached which sense wire and at which position the charge was collected.

   (c) After each *StTrsChargeSegment* is transported to the sense wire plane, a wire histogram is filled over a limited region—typically over three pad rows if diffusion properties are modelled correctly in the transport of the charge. With the position and amount of charge on each wire known, the *StTrsAnalogSignalGenerator* can be used to determine the amount of charge induced on the pad plane. This is done after the gas amplification. It will be assumed that the signal size is directly proportional to the number of electrons after the amplification. This is noteworthy because the motion of the positive ions away from the sense wires which is responsible for the main component of the signal is not modelled here. After amplification, the charge induced on the pad plane can be calculated either by a lookup table or by using analytical expressions for a pad response function. Appropriate noise for single electron statistics will also have to be added at this point. The details of this process are described in section 2. The signal generator is then responsible for summing the total charge that is to be sampled by the *StTrsAnalogSignalGenerator*. This class will fill an *StTrsSector* which is a smart vector which contains only the time bins that have a non-zero amount of charge therein.

   (d) After all the charge has been distributed over the appropriate time bins for a given segment, the *StTrsWireHistorgram* is cleared and the next *StTrsChargeSegment* is processed. This is done so that the charge information can be combined at the analog level instead of after the digitization.

5. After all segments in a single sector have been processed, it is possible to have the *StTrsDigitalSignalGenerator* digitize the signal. This will consist of a two step procedure where the pedestals are first added to the charge data, the data digitized, and the pedestals subtracted. This information will be then written out and the next sector can be processed.

4 Summary

This document summarizes the way the simulator has been designed. The most important design considerations of this package have been to factorize the processes into as many independent components as possible such that single processes can be targeted for study. By having this factorization and common interface between the different classes it is also possible to implement the situation where a fast simulation of the transport of the ionization in the gas volume is done and a detailed simulation of the analog signal generation is possible. It is also possible to add effects that have been neglected at a later time because of the modular construction. As always this is a living document and comments should be directed to:
starsimu-l@bnl.gov.

References

[1] http://www.rhic.bnl.gov/STAR/html/comp_l/simu/TpcRespSim/src/stateAnalysis.html

[2] see *A brief report on CLHEP at STAR* at:
http://www.rhic.bnl.gov/STAR/html/comp_l/simu/TpcRespSim/src/Welcome.html

[3] http://www.rhic.bnl.gov/STAR/html/comp_l/simu/TpcRespSim/src/papers/SCLdoc.ps

[4] see *A presentation at BNL about the STATUS of LHC++*at:
http://www.rhic.bnl.gov/STAR/html/comp_l/simu/TpcRespSim/src/Welcome.html

[5] W.G.Gong, *The STAR-TPC Slow Simulator*, SN0197, February 26, 1995.

[6] http://www.rhic.bnl.gov/STAR/html/comp_l/simu/TpcRespSim/src/ps/UML_full.ps

A UML Class Diagram

This section contains a preliminary version of the class structure that will form the basis of the simulation package. There are no member functions nor attributes illustrated in this diagram. For a more complete version see [6].

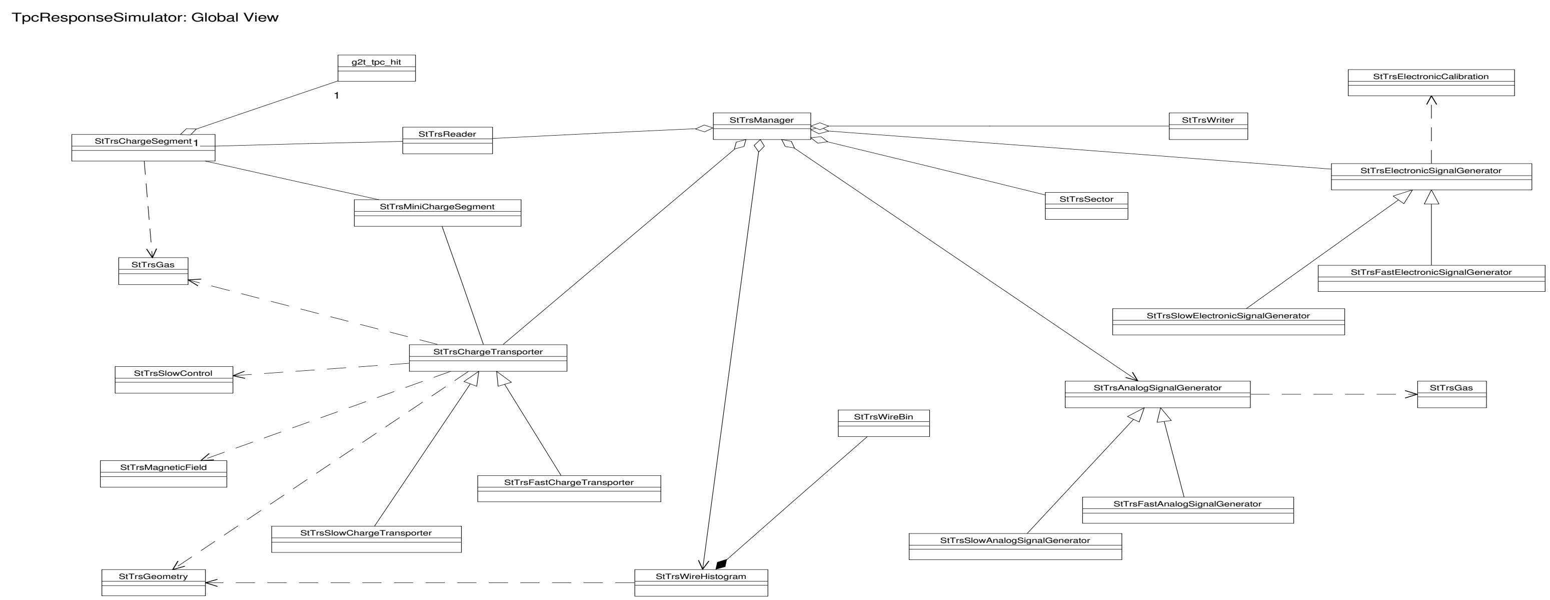

Figure 2: Preliminary class design for TRS.