SBS software: Simulation digitization/Decoder status

Eric Fuchey University of Connecticut

Winter 2021 SBS collaboration meeting



SBS Software project overview

* Main goals:

- "End-to-end" simulation:

production of pseudodata + simulation of data sizes;

Analysis of pseudodata to test analysis chain
* Requirements:

- \rightarrow *modular* (ease configuration changes);
- \rightarrow accessible (ease handling for new people);
- \rightarrow *flexible* (ease inclusion of new configurations);
- * Also need:
- Well defined IO formats and standards
- Flexible database to accomodate both MC and data (SQL ?);

Strong requirement:

Online and offline analysis both need to be ready and tested, and pseudo-data sets have to be analyzed before data taking (Summer/Fall 2021).

=> critical given high luminosities / high detectors and DAQ rates.





Simulation: G4SBS

Status: up and running, in continued development (see Andrew's review talk on software yesterday)

* Geant4-based package (doc at https://hallaweb.jlab.org/wiki/index.php/Documentation of g4sbs)

* Source code is maintained in a github repository controlled by Jefferson Lab: <u>https://github.com/JeffersonLab/g4sbs</u>

=> Master branch is most stable/full-featured (active development on uconn_dev branch)

* CMake build (requires CMake \geq 3.9)

* Prerequires root ≥ 6 and Geant4 with CLHEP;

* Contains most of the detector geometries for the "core" SBS experiments: G_{M}^{n} , G_{F}^{n} , G_{F}^{p} , SIDIS, G_{F}^{n} -RP; Active development for TDIS;

* The g4sbs output is stored as series of stl vectors.

Note: for all detectors, the elementary "hits" are consolidated as long as they are within a certain time window defined individually for the detectors (typically around 100 ns). Hits are consolidated:

- per "module" (Calorimeters/Scintillators type detectors);
- per PMT (RICH type detectors);
- per plane/particle ID (GEMs) ;

Simulation: G4SBS

G4SBS produces the following information for the following detector types:



- plane ID;
- particle ID, mother ID, track ID;
- sum of energy deposit;
- average time;
- average position in plane +in global coordinates
- position at the entrance and exit of the GEM drift
- momentum, slope of of particle

- ...

- * "Calorimeters" (+Scintillators) type detectors:
- module/cell ID;
- sum of energy deposit;
- average time;
- average position in the module + in global coordinates;
- * "RICH (cherenkov) type detectors:
- PMT ID;
- number of photoelectrons;
- average time;

(Note: only the most useful info for the digitization is reported here. The full information for the output of each detector type is available in the g4sbs documentation) Feb. 18, 2021 4



Simulation digitization: libsbsdig

* C++/root based program (requiring g4sbs as a dependency) which uses files produced by G4SBS to produce a realistic digital detector response (ADC, TDC). This information is added to the original G4SBS root tree.

- * Repository: <u>https://github.com/JeffersonLab/libsbsdig/tree/sbsdig_lw</u>
- * Documentation: <u>https://redmine.jlab.org/projects/sbs-software/wiki/Documentation_of_libsbsdig</u>



Feb. 18, 2021

Simulation digitization: libsbsdig

* C++/root based program which uses files produced by G4SBS to produce a realistic digital detector response (ADC, TDC). This information is added to the original G4SBS root tree.

* Repository: <u>https://github.com/JeffersonLab/libsbsdig/tree/sbsdig_lw</u>

* Documentation: https://redmine.jlab.org/projects/sbs-software/wiki/Documentation of libsbsdig

Features:

- * Standalone program (no need to rely on a root script)
- * Cmake build (requires CMake \geq 3.9)
- * Prerequires root \geq 6 and g4sbs;
- * Uses one unique configuration file (instead of one database per detector like the old libsbsdig code) + one text file to list the g4sbs input files.

(NB: the necessary input to add for each detector is in the documentation);

* similarly to g4sbs, the output is stored as series of stl vectors

Status:

- successfully digitizes G4SBS produced files for the GMn experiment at the rate of ~300 Hz
- includes GMn and GEp subsystems;
- basic sanity check of GMn detectors at

https://sbs.jlab.org/DocDB/0000/000065/001/gmndet_dig.pdf

- GEn-RP subsystems are being incorporated;

- background addition feature still under development/debugging (see slides 11, 12)



Simulation digitization: libsbsdig

sbsdig produces the following information for the following detector types:

* GEMs:

- strip number ;
- module number (a "module" being here defined as one readout direction of a GEM chamber)
- adc0, adc1, ... adc5
- * Cherenkov and hodoscope detectors:
 - channel number;
 - adc;
 - tdc_l, tdc_t;
- * HCal
 - channel number;
 - adc0, adc1, ... adc19
 - tdc;
- * other calorimeters:
 - channel number;
 - adc;

Note: This information is also available in the g4sbs documentation;



Simulation digitization: digitization algorithm

GEMs digitization:

1st step: ionization: * Use the energy deposit to estimate the number of ions generated; * distribute these ions uniformly between x, y, z_{in} and x, y, z_{out};



2nd step: avalanche: simulate the avalanche for each of these ions, according to a Cauchy-Lorentz function (which width will depend on the speed of diffusion of the ions in the gas, etc, and the normalization will depend on the gain);

 3^{rd} step: numerically integrate the avalanche on each of the strips: the integrated charge normalizes a pulse function $f(t) = C (t-t_0)/\tau^2 \exp(-(t-t_0)/\tau)$. (where $\tau = 56$ ns and t_0 depends on the time of the hit registered by g4sbs) This function is then integrated on the six 25ns samples and converted into ADC values;

 4^{th} step: add a pedestal (σ = 20 ADC) on each of the samples and cap samples ADC values to 4096; Note: In the full background case, steps 1, 2, 3 are repeated for the background hits before applying the pedestal

Feb. 18, 2021



Simulation digitization: digitization algorithm

PMT detectors digitization:

1st step: estimation of number of photoelectrons from the energy deposit. This estimation depends on the detector e.g. for the timing hodoscope or the HCal, there is a dependence of the light yield on the hit position; The number of photoelectrons multiplied by the PMT gain is used to normalize a pulse function $f(t) = C (t-t_0)/\sigma^2 \exp(-(t-t_0)/\sigma)$ where σ is the FWHM of the PMT.

2nd step: if the DAQ involves TDCs, the pulse is compared to the threshold (provided in database); if threshold is crossed, the leading and trailing times are recorded

3rd step: the integral of the pulse is converted into ADC values; the times are converted into TDC values; In the case of HCal (readout by FADC 250), the pulse is integrated on 4ns ADC samples

 4^{th} step: apply a pedestal (μ , σ , in database) on ADC value(s); cap samples ADC values to 2^{ADC_bits} (ADC_bits provided in db)



Note: In the full background case, steps 1, 2, 3 are repeated for the background hits before applying the pedestal



Simulation digitization: digitization algorithm

Examples below. More plots at https://sbs.jlab.org/DocDB/0000/000065/001/gmndet_dig.pdf



A = sum(ADC samples)



Simulation digitization: Background addition

Background addition:

Feb. 18, 2021

Processing of beam-on-target g4sbs files to generate "background" histograms (e.g. below) for hit multiplicity (in a certain time window), and distribution of hit position, energy deposit, number of photoelectrons when applicable, etc.

These histograms are fed as an input to sbsdig, which uses them to regenerate hits by sampling those histograms; the time distributions of those hits is generated uniformly within the detector time window.

Once those hits are generated, they are processed and superimposed to the signal hits.

11

Simulation digitization: Background addition

Background addition:

Processing of beam-on-target g4sbs files to generate "background" histograms for hit multiplicity (in a certain time window), and distribution of hit position, energy deposit, number of photoelectrons when applicable, etc.

These histograms are fed as an input to sbsdig, which uses them to regenerate hits by sampling those histograms; the time distributions of those hits is generated uniformly within the detector time window.

Once those hits are generated, they are processed and superimposed to the signal hits.

Status: under development/debugging

* Main issues:

- some information from the background seems to be lost somewhere in the program;

- having the original ADC/TDC values from the signal would be a very "nice-to-have" feature, that is currently not implemented;

- as is, digitization with full background painfully slow (a few Hz for GMn...);

Interface with SBS-offline: SBSSimDecoder

SBS-offline status: under development (see Andrew's review talk on software yesterday)

- * Repository: <u>https://github.com/JeffersonLab/SBS-offline</u>
- * Documentation: <u>https://redmine.jlab.org/projects/sbs-software/wiki/Documentation_of_SBS-offline</u>
- * Prerequires root ≥ 6 and Podd ≥ 1.6 ;

* Main challenge:

- the usual spectrometer classes from Podd do the tracking first and processes other detectors next; SBS analysis will basically have to be done the other way around (calorimeter clustering first, then tracking).

SBSSimDecoder status: Functional

(a few "nice to have" features may be added)

The role of SBSSimDecoder is to provide an interface between the digitized simulation and the SBSoffline analysis.

SBSSimDecoder unfolds the digitized simulation tree and loads the ADC and TDC values to the corresponding "THaSlotData" crate/slot, according to the simulation crate map that is provided to SBS-offline => functions correctly

The main "nice to have" feature to add is the transmission of MC information from the digitized simulation to the analysis classes.

Interface with SBS-offline: SBSSimDecoder

SBSSimDecoder status (2): processing speed study.

One concern that came up during the development of the SBSSimDecoder is the speed of execution of the SimDecoder itself (we do **not** want this decoder to be a speed bottleneck).

The table below lists the time needed to perform different set of actions over 48000 events :

Action	Time (s)	Speed (kHz)
Traverse the digitized tree, looping on digitized tree vectors	14	3.5
+ load the ADC/TDC values in the THaSlotData objects	18	2.6
+ decode all detectors	28	1.7
+ save output for all detectors	41	1.2

More details in: <u>https://sbs.jlab.org/DocDB/0000/000073/001/SpeedDiagnose_sbsoffline.pdf</u>

Summary

* libsbsdig and SBSSimDecoder are now functional, and can be utilized to develop SBS-offline

* a few key features are still in the works for libsbsdig (namely the background addition, which is my highest priority)

* beyond that, a few subsystems still need to be incorporated into libsbsdig (namely for GEn-RP, SIDIS)

* An article about this work is in preparation : <u>https://github.com/efuchey/SBSdig_writeup</u>

Thank you !

Earm.BBGEM.dighit.module:Earm.BBGEM.dighit.strip

17