Disclaimer:

Apologies for the somewhat eclectic feel of this talk. Its content is liberally lifted out of several sources:

- Hall C12 GeV Soft. Progress, Software Meeting, 11/05/12
- Research Management Plan for the Hall C 12 GeV Detector Software, 03/03/12
- Hall C++ Analyzer progress: Calorimeter, 12/09/12
- Hall C 12 GeV Software Review, 06/05/12
- Hall C++ Analyzer progress: Hodoscope, 12/12/12

Contributors:

S. Wood, M. Jones, B. Sawatzki, S. Zhamkochyan, V. Tadevosyan, GN
Outline

- Motivation
- Existing Hall C analyzer (engine)
- 12 GeV-era Hall C analyzer (hcana)
- Summary/Outlook
  - (Tons of stuff I will skip
    - Analysis algorithms (tracking, pid, etc.)
    - Workflow/Calibration/Monitoring
    - Hall C Simulation (simc))
Rationale:

- As part of the 12 GeV upgrade Hall C is building a new spectrometer: SHMS. It needs software.
- Existing Hall C software engine: F77/Cernlib
- ~100,000 Lines of Code
- C++ is a better match for the skillset of younger collaborators

“das Kind mit dem Bade ausschüttent”- Thomas Murner, 1512, Narrenbeschwörung

Keep all the solid, documented, and proven analysis algorithms from “engine”
Management Infrastructure:

- **Software manager:** Mark Jones, JLab
  - **C++/ROOT Analyzer:** Gabriel Niculescu, JMU
  - **Fortran Analyzer:** Ed Brash, CNU
  - **Simulation (SIMC):** Dave Gaskell, JLab
  - **Calibrations:** John Arrington, ANL
  - **Online histogramming:** Pete Markowitz, FIU

"Bureaucracy is the price we pay for impartiality"
**Milestones**

**Present**
- Set-up Management structure
- Monte Carlo simulation is ready
- Decided on Git for code management of C++ analyzer

**2012**
- July: Define reference HMS data for testing code
- Sep: Documented non-tracking HMS detectors code in Fortran Analyzer
- Oct: Make DAQ decoding in C++ Analyzer object-oriented
- Oct: Ability to analyze Hall C data at the raw data level in C++ Analyzer
- Dec: Documented the drift chambers and tracking code in Fortran Analyzer
- Dec: Verify HMS hodoscope analysis in C++ Analyzer
Milestones (Part 2)

<table>
<thead>
<tr>
<th>Year</th>
<th>Milestones</th>
</tr>
</thead>
</table>
| 2013 | Jun: SHMS code added to Fortran Analyzer.  
      | July: Full analysis of HMS data with C++ Analyzer ready.  
      | Sep: C++ Analyzer ready for SHMS calorimeter tests.  
      | Dec: Full analysis of HMS data with C++ Analyzer verified by comparison to Fortran analyzer. |
| 2014 | Jan: Scalar and BPM analysis code in C++ analyzer.  
      | Feb: Calibration codes ready.  
      | Jul: Analyze cosmic ray data in SHMS with both Analyzers.  
      | Sep: First beam, analyze data with both Analyzers. |

*With these milestones, and the infrastructure shown earlier a few of us went in front of a Software review Committee in June 2012...*
Approach

- Keep all analysis algorithms from **engine**
- Rewrite in ROOT/C++
- Built on top of Hall A’s **PODD** software
- ...In publicly readable **git** repository

“If you know your history
Then you would know where you coming from”
**GIT**

- “Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion (SVN), CVS, Perforce, and ClearCase with features like **cheap local branching**, **convenient staging areas**, and **multiple workflows**."
- **http://git-scm.com/**
- **Effort spearheaded by one L. Torvalds…**
Current Status

- Reads Hall C style parameter files
- Reads Hall C style hardware (detector mapping)
- Builds engine-style raw hit lists
- Extracts hodoscope and drift chamber hit lists from HMS CODA files
- Hodoscope reconstruction/rest of milestones to follow
**Engine Overview**

- **hcana** will follow existing **engine** approach...

- As that doesn’t really mean much for most of you I better take a step (or 3) back!
HMS Detector Array
motivation  engine  hcana  summary

HMS Vacuum pipe exit

HMS

SHMS

(för π/e at high E' only, otherwise vacuum)

THomas Jefferson National Accelerator Facility

Gabriel Niculescu, Hall A & C Software Workshop
Engine Overview:

- F77/Cernlib based. ~100k LOC
- Code “identical” for spectrometers (HMS/SOS, HMS/SHMS)
- Some parts of it (CTP) in C (S.W.)
  - pipeline data input/output (run on the farm way b4 farm)
  - Run-time histogram definition, cuts, filling, weighting
  - Custom scaler report(s)
  - Detailed detector layout defined by position/geometry parameter files (which look different than Hall A!)
- Standalone and integrated calibration and diagnostic tools
Engine Overview (II)

- Raw hit processing/decoding
  - [updates for new ADC/TDCs in progress] – Brad S.
  - Option to dump info needed for detector calibration
- Track-independent detector quantities
- Track-dependent reconstruction
  - Calculate efficiencies for each detector
    - Robust algorithms which yield reliable measure of performance.
    - Not tuned to specific experiment (e.g. extreme rates or backgrounds may require modified approaches)
  - Calculate ‘basic’ physics quantities for each event
- Heavy emphasis on experiment-independent issues
Engine Overview (III)

- Each experiment...
  - Must provide higher-level physics reconstruction
  - Must decide if they want to use more specialized efficiency calculations
  - Must determine efficiency of experiment-specific cuts
  - For most cuts/calculations there are multiple predefined (“standard”) options
Calibration (time-of-flight)

**Initial calibration**
- Stand-alone code, significant user intervention
  1. Uses special engine (txt) output file
  2. Correct for velocity of light propagation along paddle
  3. Apply pulse-height correction
  4. Determine offset for each PMT

**Update calibrations**
- Stand-alone code; both full and simplified version
- Keep propagation, pulse-height corrections fixed and refit offsets
Calibration (time-of-flight)

Figure 4-3: Difference between corrected time and the average corrected time of all other hits on a track. This quantity measures the hodoscope timing resolution, with all corrections applied, after calibration.
motivation  engine  host

Monitoring

- Same version for on/offline replay
- Replay Strategy: reconstruct events, dump info in histograms, ntuples, text files
- Variety of monitoring tools to help users assess the quality of the data and/or identify problems
On/Offline Analysis:

- Same version for on/offline replay
- Replay Strategy: reconstruct events, dump info in histograms, ntuples, text files
- Variety of monitoring tools to help users assess the quality of the data and/or identify problems
Software review
recommendations
Review Hall C Recommendations

- With the somewhat aggressive schedule leading up to December 2013, make sure to engage a reasonable number of early adopters to stress test the new framework.
- Re-use existing efforts from Hall A to decode CODA-formatted data in ROOT.
- If resources are limited, the Fortran-based SHMS reconstruction should be a low priority.
- While we encourage the move to git as a code management system, be sure not to underestimate the extent of the paradigm shift. Identify a workflow model for your use of git. Communicate clearly the new paradigm (easy branching, no central repository, etc) Set up (or link to) tutorials for users with a mapping of routine CVS tasks to their git equivalents (such as cvs diff, etc). Document or link to documentation for standard git tasks without obvious equivalent in CVS or SVN, such as git rebase, or bisect.

Actions:
- Biweekly software meetings
- Definitely using the existing Hall A software to decode CODA
- To promote Git usage:
  - Arranged a seminar on Git
  - Encourage use in managing other projects
Review General Recommendations (2)

- Nightly builds are performed by some; we recommend them for all.
- Evaluate standard code evaluation tools, such as valgrind, clang’s scan-build, cppcheck, Gooda, ... for inclusion in the software development cycle. We suggest looking at an Insure++ license as well.
- Run a code validation suite such as valgrind as part of the routine software release procedure.
- Give full and early consideration to file management, cataloging and data discovery by physicists doing analysis. Report on this area in future reviews.

Actions:

• Started to investigate code evaluation tools
Selected HMS run for analysis (50017)

HMS documentation
- Calorimeter done.
- Scintillator planes has started.
- Aerogel, Cerenkov and Drift chambers not started.

HMS coding in “hcana”
- Can analyze data at the raw CODA level. Read-in Hall C detector maps and parameter files.
- Calorimeter and scintillator has made comparison to Fortran analyzer with “slightly processed” data.
- Aerogel detector has started
Hall C++ Analyzer progress: Calorimeter

Simon Zhamkochyan,
Vardan Tadevosyan
Shower detector (HMS)

- The Shower detector consists of 52 lead glass blocks, arranged in 4 layers and 13 rows.
- In the 1st and 2nd layers PMTs are attached to both right and left sides ("positive" and "negative" ADC channels).
- In the 3rd and 4th layers PMTs are on the right side only ("positive" ADC channels).
Analysis process: Init, Read DB

- **Run initialization**
  - Determining run number
  - Determining database
  - Determining detectors

- **Reading data from database:**
  - Geometrical parameters
  - Pedestal values
  - ADC channel to Shower block number mapping
  - Calibration constants
Analysis process: Decode

- **Decoding data**
  - Distinguishing pedestal events
  - Calculating pedestals mean values and sigmas
  - Calculating signal thresholds: \( Thr = \text{Max}(50, \text{Min}(10, 3*\sigma)) \)
  - Reading raw ADC value
  - Calculating pedestal subtracted values

- **Output**
  - Determining basic variables
  - Filling the tree
  - Filling and saving histograms
Comparing \texttt{hcana} results with Fortran engine

- ADC hit maps; MDUALITY experiment, run # 50017
- engine histograms are deliberately shifted to be seen together with the C++ analyzer histograms

Results match!!
Comparing hcana and engine results
same run # 50017...
Results match!!!
HMS Hodoscope:

- GN
- Similar progress as the Yerevan group
- Can read/decode/fill raw scintillator data (S.W.)
- Can read position, calibration parameters* (S.W.)
- **If you think this is not a big deal consider**
  - (these are valid for all of hcana, not the hodoscope alone):
  - Need to keep existing Hall C database format
  - Need to keep existing Hall C variable names “available”. (Try keep users content)
  - Need to NOT re-write Ole’s code (to keep JLab management content)
  - **Strive to write C++ code and not F77 translated into C++!**
general.param file

- #real raddeg
- raddeg=3.14159265/180
- ; hms/sosflags.param include spectrometer offsets and options.
- #include "PARAM/genflags.param"
- #include "PARAM/hmsflags.param"
- #include "PARAM/gtarget.param"
- #include "PARAM/hdc_offsets.param"
- #include "PARAM/hdc.pos"
- #include "PARAM/hhodo.pos"
- #include "PARAM/hcal.pos"
- #include "PARAM/sdc_offsets.param"
- #include "PARAM/sdc.pos"
- #include "PARAM/shodo.pos"
- #include "PARAM/scal.pos"

I deleted some lines so that it fits on the page
hhodo.pos file

hpathlength_central = 2500

; Z positions of hodoscopes
hscin_1x_zpos = (89.14-11.31)  
-48.750
hscin_1y_zpos = (108.83-11.31)  
-41.250
hscin_2x_zpos = (310.13-11.31)  
-33.750
hscin_2y_zpos = (329.82-11.31)  
-26.250
hscin_1x_size = 8.0  
-18.750
hscin_1y_size = 8.0  
-11.250
hscin_2x_size = 8.0  
-3.750
hscin_2y_size = 8.0  
3.750

; Number of hodoscope paddles per layer
hscin_1x_nr = 16  
11.250
hscin_1y_nr = 10  
18.750
hscin_2x_nr = 16  
26.250
hscin_2y_nr = 10  
33.750

; X,Y positions of hodoscope paddles
hscin_1x_left = 37.750  
48.750
hscin_1x_right = -37.750  
56.250
hscin_1x_offset = -1.3
**motivation engine**  
**hcana**  
**summary**  

**hhodo.param file**

```plaintext
; hstart_time_center  center of allowed time window
  hstart_time_center = 35.

; hstart_time_slop    1/2 width of time window
  hstart_time_slop = 20.

; ...
  hhodo_slop = 2., 2., 4., 4.

; hhodo_pos_sigma = .3,.3,.3,.3,.3,.3,.3,.3
  .3,.3,.3,.3,.3,.3,.3,.3
  .3,.3,.3,.3,.3,.3,.3,.3
  .3,.3,.3,.3,.3,.3,.3,.3
  .3,.3,.3,.3,.3,.3,.3,.3
  .3,.3,.3,.3,.3,.3,.3,.3
  .3,.3,.3,.3,.3,.3,.3,.3
```

Thomas Jefferson National Accelerator Facility  
**Gabriel Niculescu, Hall A & C Software Workshop**
Hodoscope: Engine

- HMS s1x+ ADC hits t=user
- HMS s1x+ TDC hits t=user
- HMS s1y+ ADC hits t=user
- HMS s1y+ TDC hits t=user
- HMS s2x+ ADC hits t=user
- HMS s2x+ TDC hits t=user
- HMS s2y+ ADC hits t=user
- HMS s2y+ TDC hits t=user
Monitoring

- Same HMS hodoscope raw ADC & TDC hits as before!
- Done in hcana!
Summary/Outlook:

- Management Structure, Milestones in place
- HMS documentation
  - Calorimeter done. Hodoscope in progress.
- HMS coding in “hcana”
  - Can analyze data at the raw CODA level.
  - Read-in Hall C detector maps and parameter files.
  - Calorimeter and Hodoscope have made comparison to Fortran analyzer with “slightly processed” data.
- Aerogel detector has started
- Seeking more manpower for this effort
  - Cross-checks, validation, deployment (E.C.?, I.N.?)
- Confident that we’ll meet milestones
Back-up from here on
Analysis Algorithm

- **For each detector:**
  - **Calibration:**
    - Options to dump information for diagnostic/detector calibration
    - Some detectors have built-in calibration options
  - “Detector reconstruction”
    - Tracking, TOF, npes, shower energy, etc
  - **Efficiency Calculation**
    - Default efficiency calculation built in
    - Options for experiment-specific calculations
For each spectrometer:

- Track and particle identification (at the target)
  - Use all detector information to obtain focal plane quantities
  - Use matrix elements to obtain target quantities
  - Options to apply corrections (e.g., eloss, pointing, etc.)

\[
T_{\text{tar}} = \sum_{i,j,k,l} R_{i,j,k,l}^y (X_{fp})^i (Y_{fp})^j (X'_{fp})^k (Y'_{fp})^l , \quad 1 \leq i+j+k+l \leq N ,
\]
Analysis Algorithm (III)

- **If a coincidence experiment:**
- **Coincidence event reconstruction:**
  - Very Experiment-specific!
  - The Fortran code provides subroutines for this (c_reconstruction, c_physics, etc.).
  - The default code only provides calculation for very basic quantities
- Historically some/most users just produce a coincidence ntuple and use their own coincidence reconstruction software (.f, .kumac, or both)
(S)HMS Tracking

- Two drift chambers / spectrometer
- 6 wire planes/chamber
- || wires shifted by ½ cell
- Need scintillator hodoscope signal (common stop)
moti(S)HMS Tracking

Tracking algorithm:
- **Identify pairs**: hits in overlapping wires (non-parallel planes)
- **Identify combos**: group pairs that are within $R \sim 1.2$ cm
- **Generate spacepoints**: combine combos within $R \sim 1.2$ cm
- **Generate stubs** (single-chamber tracks) for all spacepoints
- **Find tracks**: link stubs between two chambers
- **Apply drift time offset, determine L/R**
  - Between planes where offset
  - Best chi-squared for unmatched planes
- **Calculate track-dependent quantities** for all surviving tracks
- **Select ‘final’ (focal plane) track**

Various places where cuts can be applied. By default, only raw timing on DC TDC hits.
Figure 4-5: Example event display of results of HMS tracking. Hit wires in chamber 1 (above, left) and chamber 2 (above, right), with X(black) and X’(red) horizontal wires, Y(black) and Y’(red) vertical wires, and U(green) and V(blue) wires. The projection of the fitted track to the middle of each chamber is marked by the magenta x.
(S)HMS Tracking

\[ \chi^2 / \text{ndf} \quad 1774 / 454 \]
\[ p_0 \quad 0.984 \pm 4.937 \times 10^{-5} \]
\[ p_1 \quad -0.0001 \pm 8.432 \times 10^{-7} \]
Beyond single event reconstruction

- Sample workflow for typical Hall C experiment
- Programs & scripts available to the user
- Most scripts work “out of the box”
- Some will require user customization
Drift Chamber Time-to-Distance Calibration

![Drift Chamber Time-to-Distance Calibration Graphs](image-url)
Calibration (electron PID)

**Cerenkov:**
- Gain match PMTs
- Check position dependence of response

**Calorimeter**
- Correct attenuation along blocks
- Gain match individual blocks with clean e- samples at multiple (S)HMS momenta
- Match left, right PMTs

- Identify e- inefficiency and pion contamination, using combination of Cerenkov and calorimeter cuts
Calibration (hadron PID)

- **Aerogel**
  - Gain match PMTs
  - Velocity vs. Aerogel
    - Separate pions (and e+) from heavier
  - e-h coincidence time
    - Separate $\pi/K/p$...
- **Velocity vs. $dE/dx$**
  - Separate $\pi/p/d$ at lower momentum
Hall C Matrix Element Optimization Package (CMOP)

Kétévi A. Assamagan, Dipangkar Dutta, Pat Welch

May 2, 1997

Abstract

We present a fitting code to optimize the transport matrices of the Hall C spectrometers.
**Figure 4.10:** The focal plane quantities.
COSY matrix element optimization

\[ T_{\text{tar}} = \sum_{i,j,k,l} R_{i,j,k,l}^\beta (X_{fp})^i (Y_{fp})^j (X'_{fp})^k (Y'_{fp})^l , \quad 1 \leq i+j+k+l \leq N , \]
Figure 4.12: Reconstruction of the vertical (left) and horizontal (right) sieve slit hole patterns.
**COSY matrix element**

**Mean yptar +/- RMS diff (mrad)**

- \( z_{tar} (cm) \)

**Mean xptar +/- RMS diff (mrad)**

\( z_{tar} (cm) \)

_A. Puckett, PhD Dissertation_
**SIMC Overview**

**Initialization**
- Choose reaction, final state (if appropriate)
- Disable/enable implementation of (or correction for) raster, eloss ...

**Event generation**
- Select vertex based on target size, position, raster size, beam spot size
- Determine energy, angle generation that will populate 100% of the acceptance (accounting for radiation, energy loss, ...)

**Physics Processes**
- Event-by-event multiple scattering, radiative corrections, particle decay, coulomb corrections
Acceptance

Can apply geometric cuts or spectrometer model. Default spec. models include target/spec. offsets, model of magnetic elements, apertures at front, back, middle of magnets, collimators, detector active area

Event Reconstruction

Tracks are fitted in the focal plane and reconstructed to the target. Apply average energy loss, fast raster corrections (consistent with data analysis). Calculate physics quantities for Ntuple.
Beyond single event reconstruction

- Internal efficiency calculations (detector, tracking...)
  - Experiment-specific modifications (e.g. tracking efficiency vs. particle type, high-background data, singles vs. (rare) coincidences)

- Scalar analysis
  - Defaults
  - Time charts (dump of E/x/y vs. t)
  - beam on/off

- Diagnostics
  - raw detector dumps
  - online monitoring

- Calibrations
  - Internal (including continuous)
  - Data dump for external calibrations
Beyond single event reconstruction

- Corrections (beam drift, FR, special [coin/cer block...])
- Cuts, normalized yields in arbitrary bins.
- Simulation: ratio method, include all ‘physics’ event-by-event, apply global corrections (mean eloss) as in analyzer

- At present, Engine has robust default estimates for efficiencies, calibrations, etc... with options for specific conditions (or user-provided experiment-specific changes).
- **New framework**: may want multiple (or selectable) versions to allow simpler selection of optimized approach. Fortunately this capability already exists in PODD.
Calibration (electron PID)

Electron PID
**Calibration (electron PID)**

- **Cerenkov:**
  - Gain match PMTs
  - Check position dependence of response

- **Calorimeter**
  - Correct attenuation along blocks
  - Gain match individual blocks with clean e- samples at multiple (S)HMS momenta
  - Match left, right PMTs

- **Identify e- inefficiency and pion contamination, using combination of cerenkov and calorimeter cuts**
Drift Chamber Calibration

Drift Chamber Time-to-Distance Calibration

V. Tvaskis, PhD Dissertation
To convert from drift times to drift distances, it is assumed that the drift distance is a monotonically increasing function of the drift time $d_{\text{drift}} = d(t_{\text{drift}})$. Since the size of a drift cell in the HMS is small compared to the envelope of tracks populating the active area of the chambers, the change in the relative number of incident tracks within any given drift cell can be assumed to be small. It is a very good approximation to assume a uniform distribution of drift distances within a cell. Even given small cell-to-cell variations in the relative non-uniformity of the distribution of incident tracks as a function of drift distance, when averaged over all the drift cells in a plane of wires and/or all planes within the chambers, the assumption of uniform drift distance is quite robust. This assumption greatly simplifies the time-to-distance conversion, since it is straightforward to map the observed drift time spectrum onto a uniform distance distribution. The drift time distribution $n(t_{\text{drift}})$ is defined as the probability density of events as a function of drift time within the allowed time window.

$$\int_{t_{\text{min}}}^{t_{\text{max}}} n(t)dt = 1 \quad (4.12)$$

For a uniform distribution of events, then, the drift distance for a given drift time is simply given by the integral of the observed drift time spectrum up to the measured $t_{\text{drift}}$:

$$\frac{1}{d_{\text{max}}} \int_{0}^{d_{\text{drift}}} dx = \frac{d_{\text{drift}}}{d_{\text{max}}} = \int_{t_{\text{min}}}^{t_{\text{drift}}} n(t)dt$$

$$d_{\text{drift}}(t_{\text{drift}}) = d_{\text{max}} \int_{t_{\text{min}}}^{t_{\text{drift}}} n(t)dt \quad (4.13)$$
Figure 4-1: Profile histogram of $t_{raw}$ as a function of the pulse height measured by the ADC for the PMT at the + end of S1X paddle 8, with fitted walk correction shown.
Time to distance calibration
Drift chamber wire maps
Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, **convenient staging areas**, and **multiple workflows**.

; Setup file for engine
begin parm engine

;CODA EVENT FILE.
gen_run_number = 1

; g_data_source_filename = '/home/cdaq/daq03/coda2/runlist/daq03_%d.log'

; g_data_source_filename = '/cache/mss/hallc/e93021/raw/oct97_%d.log'

; g_data_source_filename = '/cache/mss/hallc/daq03/raw/daq03_%d.log'
g_data_source_filename = '/group/hallc/gabriel/hcana/work/daq04_%d.log.0'

; g_data_source_filename = '/w/work2802/fpi2-data/daq03_%d.log'

  g_ctp_database_filename = 'standardfpi2.database' ; pointers to map/param files.
; g_ctp_test_filename = 'pion.test' ; ctp test definitions.
; g_ctp_hist_filename = 'pion.hist' ; ctp histogram definitions.

Sample Parameter Files:
REPLAY.PARM
Sample Parameter Files:

- general.param

#real raddeg
raddeg=3.14159265/180

; hms/sosflags.param include spectrometer offsets and options.
#include "PARAM/genflags.param"
#include "PARAM/hmsflags.param"
#include "PARAM/sosflags.param"

#include "PARAM/gdebug.param"
#include "PARAM/hdebug.param"
#include "PARAM/sdebug.param"
#include "PARAM/htracking.param"
#include "PARAM/stracking.param"

#include "PARAM/gtarget.param"
#include "PARAM/hdc_offsets.param"
#include "PARAM/hdc.pos"
#include "PARAM/hhodo.pos"
#include "PARAM/hcal.pos"
#include "PARAM/sdc_offsets.param"
Sample Parameter Files:

```
hhodo.pos
```

- `hpathlength_central = 2500`
- `; Z positions of hodoscopes`
- `hscin_1x_zpos = (89.14-11.31)`
- `hscin_1y_zpos = (108.83-11.31)`
- `hscin_2x_zpos = (310.13-11.31)`
- `hscin_2y_zpos = (329.82-11.31)`
- `hscin_1x_dzpos = 2.12`
- `hscin_1y_dzpos = 2.12`
- `hscin_2x_dzpos = 2.12`
- `hscin_2y_dzpos = 2.12`
- `hscin_1x_size = 8.0`
- `hscin_1y_size = 8.0`
- `hscin_2x_size = 8.0`
- `hscin_2y_size = 8.0`
- `hscin_1x_spacing = 7.5`
- `hscin_1y_spacing = 7.5`
- `hscin_2x_spacing = 7.5`
- `hscin_2y_spacing = 7.5`
- `; Number of hodoscope paddles per layer`
- `hscin_1x_nr = 16`
Optics
Fast Physics Monte Carlo: SIMC

- Standard Hall C (or A!) MC for coincidence reactions
  - HMS/SOS/HRS/SHMS optics models (COSY)
  - Aperture checks, detector FID cuts yield acceptance
  - Detailed implementation of radiative corrections, multiple scattering, ionization energy loss, particle decay
- Simple prescriptions for some FSIs, Coulomb corrections

Reactions implemented:
- Elastic and quasielastic $H(e,e'p)$, $A(e,e'p)$
- Exclusive pion and kaon production
  - $H(e,e'\pi^+)n$, $A(e,e'\pi^+/\pi^-)$ [quasifree or coherent]
  - $H(e,e'K^+)\Sigma$, $A(e,e'K^+/\pi^-)$ [quasifree or coherent]
  - $H(e,e'\pi^+/\pi^-)X$, $D(e,e'\pi^+/\pi^-)X$ [semi-inclusive]
  - $H(e,e'K^+/\pi^-)X$, $D(e,e'K^+/\pi^-)X$ [semi-inclusive]
  - $H(e,e'\rho\to\pi^+\pi^-)p$, $D(e,e'\rho\to\pi^+\pi^-)$ [diffractive $\rho$]
Fast Physics Monte Carlo: SIMC

- **SIMC is NOT a full detector response simulation a la GEANT**
  - Detectors implemented via impact on acceptance
  - Efficiency corrections (generally) applied to data

- **SIMC does NOT simulate a large class of processes simultaneously to generate backgrounds (e.g. like Pythia)**

- **SIMC is NOT a generic event generator:**
  - Processes generated over a limited phase space, based on fully populating spectrometer acceptances

- **SIMC is NOT hard to modify**
  - Many non-standard or experiment-specific tests can easily be performed based on simulated information
SHMS simulation in SIMC

T. Horn, Jlab/CUA
Hydrogen elastic:
Radiative correction + fit to form factors as input
NO other free parameters!
Kaon electroproduction:

Hydrogen hyperon cross section model + $^2\text{H}$ (also $^3,^4\text{He}$) spectral functions + FSI + Kaon decay + RC + Norm. Factor for $\Lambda, \Sigma$ peaks

$p(e,e'K^+)\Lambda$  
$d(e,e'K^+)\Sigma^0$
Examining decay-at-target events

Simulated decay distance (cm) for accepted $\pi \rightarrow \mu \nu$ decays

Coincidence time (ns)

Data

SIMC
Documentation

- Continuously “in-progress”/improving
- Moving to C++/ROOT will help (doxygen, etc.)
- Hall C users/staff actively maintain electronic/online resources (wiki, DocDb, elog, newsletter, etc.)
- Provide the “Why?”, not only the “How?”
- Hall C wiki (general Hall C entry point)
- Hall C 12 GeV Software wiki
- Hall C software documentation
- Hall C PhD Theses