svtsim status

Bill Ashmanskas, CDF simulation meeting, 2001-11-15

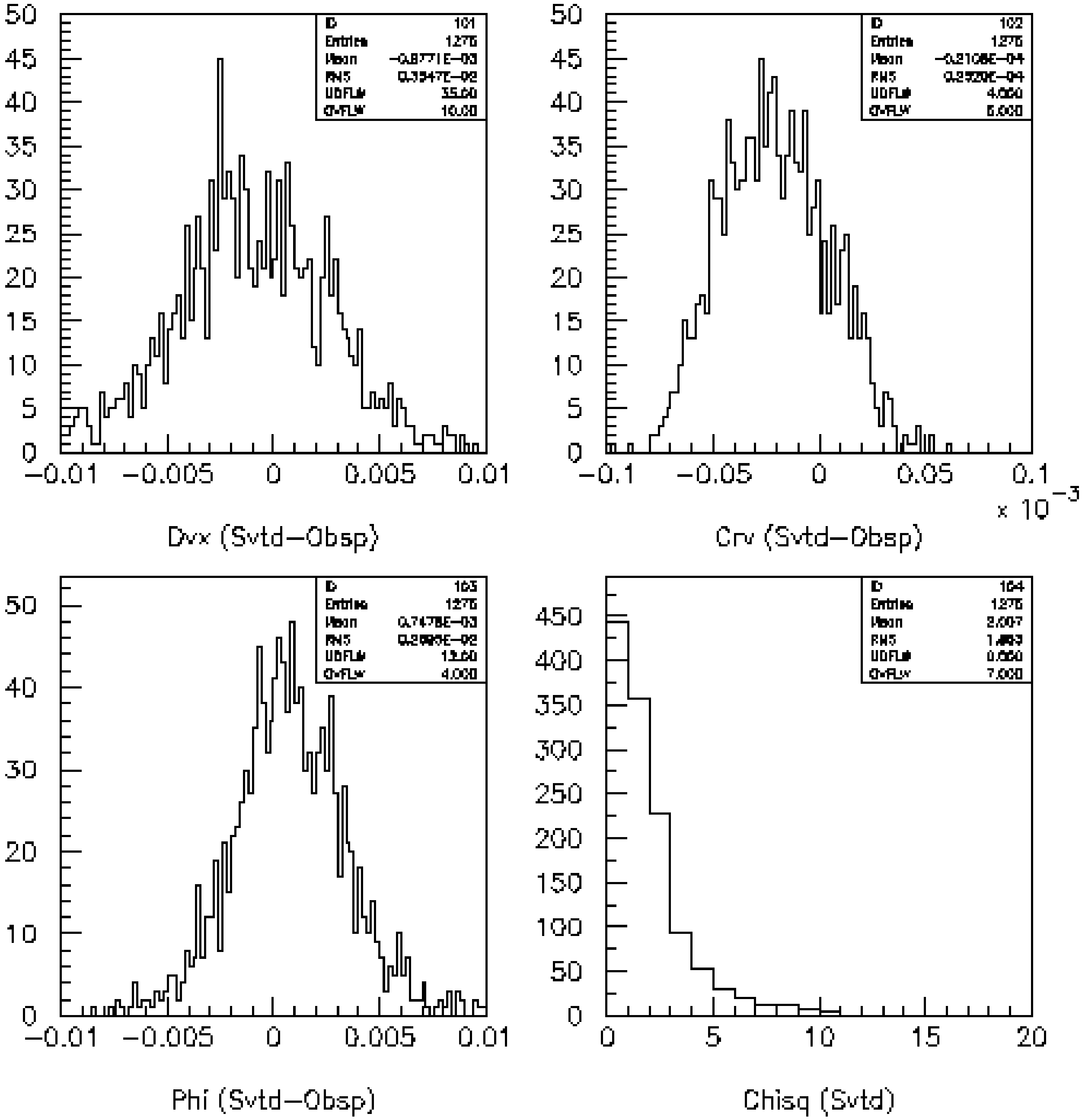
- Main authors: Ashmanskas, Belforte, Cerri, Nakaya, Punzi
- Design goals/features:
  - main goal is perfect emulation of plausible SVT configurations
  - core library is modular “object-like” ANSI C
    - simple, portable
    - svtsim_hf_t *myhf = svtsim_hf_new(); etc.
    - run online in VME crates, stand-alone in test stand, and offline in AC++
  - code is organized to reflect structure of hardware
  - thin layer of C++ to interface to AC++
  - straightforward to have/use multiple configurations
    - now driven by ASCII files, selectable via talk-to
    - database implementation is in the works
  - efficient memory use: don’t store what you can easily compute
  - efficient CPU use: e.g. choose appropriate data structures and algorithms
    - ~50ms/event on recent 2-track data, including clustering, road search, and fitting
    - could be faster: last optimization was 1-2 years ago

- Current use:
  - on-crate conversion of high-level configuration data (patterns, fit coefficients, etc) into RAM images at run initialization
  - validation of new configuration constants on old or simulated data before first online use
  - offline validation of hardware performance
  - offline analysis of algorithm performance
  - “random test” of data flow in test stand
    - both boards and emulators have Python interface for rapid development of tests
- Current non-use:
  - on-crate board emulation has been run with some test data, but is not yet ready for real-time validation
    - most of this comes for free once we can run offline with perfect emulation, robust error handling
  - management of configuration files is still too intricate for casual users
    - database fixes this
  - don’t know anyone using it to predict trigger efficiencies, rates, etc.
    - probably a less detailed simulation is more appropriate for most of these studies

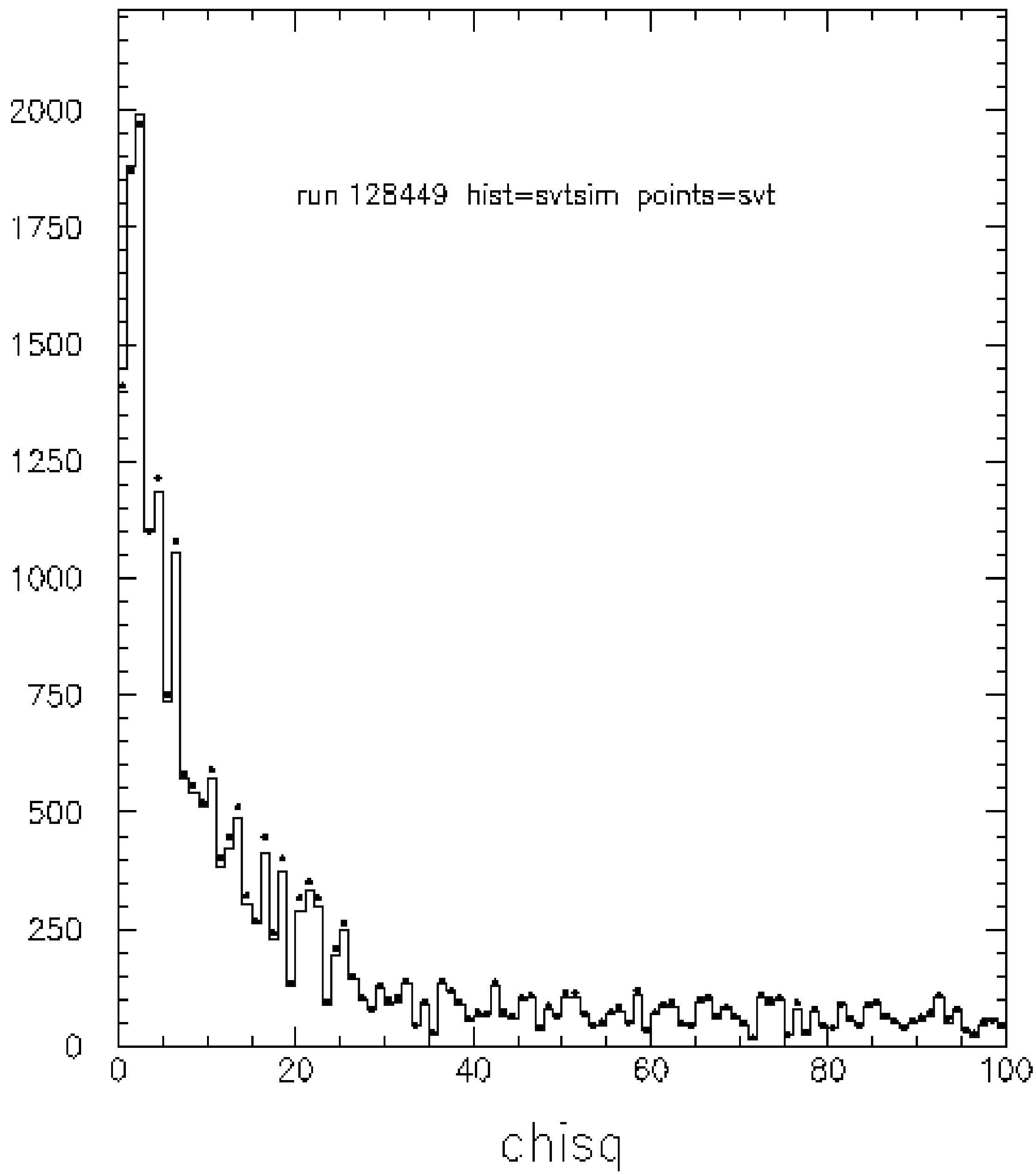
- Recent progress:
  - quite good emulation of run 128449 (latest run)
    - along the way, we found 4 unplugged fibers, 2 swapped fibers
    - now use SVT's own readout of XTRP cable data
    - there are a few board features that we still don't emulate perfectly; need to decide whether it's easier to model what the board does or to change what the board does
    - found only 1 of the 36 Hit Finders in which we're not able to predict the hits perfectly; no explanation yet
  - working to extend back to other important September/October runs
    - main reason to do this is to exercise book-keeping
    - also want to have good emulation of some unbiased (by SVT) runs, to study efficiency
  - database
    - SVT_DATA table exists, is replicated
    - some configuration data has been stored
    - working on reading it from offline
      - has been done from stand-alone program that links to libraries that are distributed with the offline code
      - official support expected in December; we have what we need to survive until then

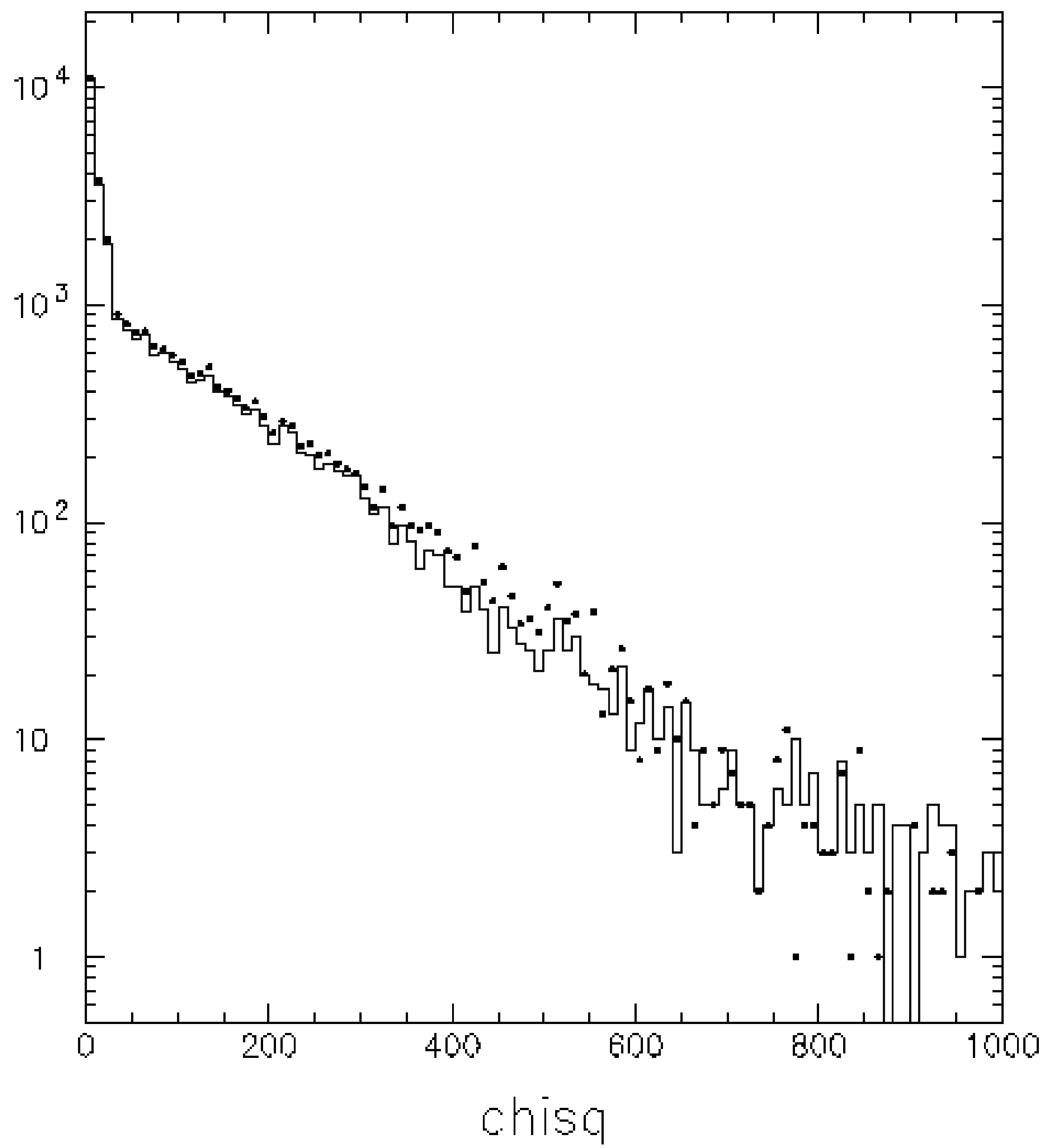
- Significant to-do items:
  - finish database implementation (3 parallel parts):
    - tracing/fixing any remaining sim/data discrepancies
    - driving online configuration from database
      - enforces bookkeeping mechanism
    - driving offline emulation from database
      - most important part is access to small quantity of run-dependent bookkeeping data (~100 bytes/run)
      - it is also convenient to use DB to archive and to serve larger quantity of configuration data (~10MB/config)
  - emulate online beam subtraction
    - software is trivial; some firmware work is needed to make every event's output perfectly predictable
  - run svtsim on a huge volume of raw data to ensure that emulator and hardware are doing the same thing
    - perfect emulation of all "good" runs is necessary and sufficient criterion that bookkeeping is adequate
  - extend functionality (both svt and svtsim in parallel):
    - add non-linear fit terms (computed by G. Punzi)
    - allow layers other than 0123 (trivial software change)
    - implement "4/5" tracking (allows one SVX miss)
    - include L00
  - better interface to intermediate results
    - e.g. Xin Wu's HitFinder→SiHitSet conversion

- How to run it
  - svtsim reads SIXD and XFLD (or equivalent XTRP cable output recorded by SVT) and writes SVTD
  - I try to keep some up-to-date instructions at `http://hep.uchicago.edu/~ashmansk/sundry/running_svtsim.html`
  - There is a script (in svtsim/test) to throw single muons, which works in 3.18.0, but it doesn't work with the 4.0.0 cdfSim executable, for reasons I haven't had time to debug. (Volunteers?)
  - The example/instructions also show how to build TrigSim++ and use it to run XFTsim and svtsim; this works with both 3.18.0 and 4.0.0. Packages needed are TriggerMods and svtsim.
  - I think recent data are not readable with 3.18.0, so 4.0.0 is needed if you want to run on data. There is an example script for that, too.
    - The hard part is that the configuration files for emulating SVT's performance on data are in b0dap30:/cdf/code_common/cdfonline/svt_config
    - This is fixed by database implementation, which is now days, not weeks, away
  - The example scripts fill a simple ASCII ntuple and use it to make some postscript plots; more complete ntuples are in the SVTmon package

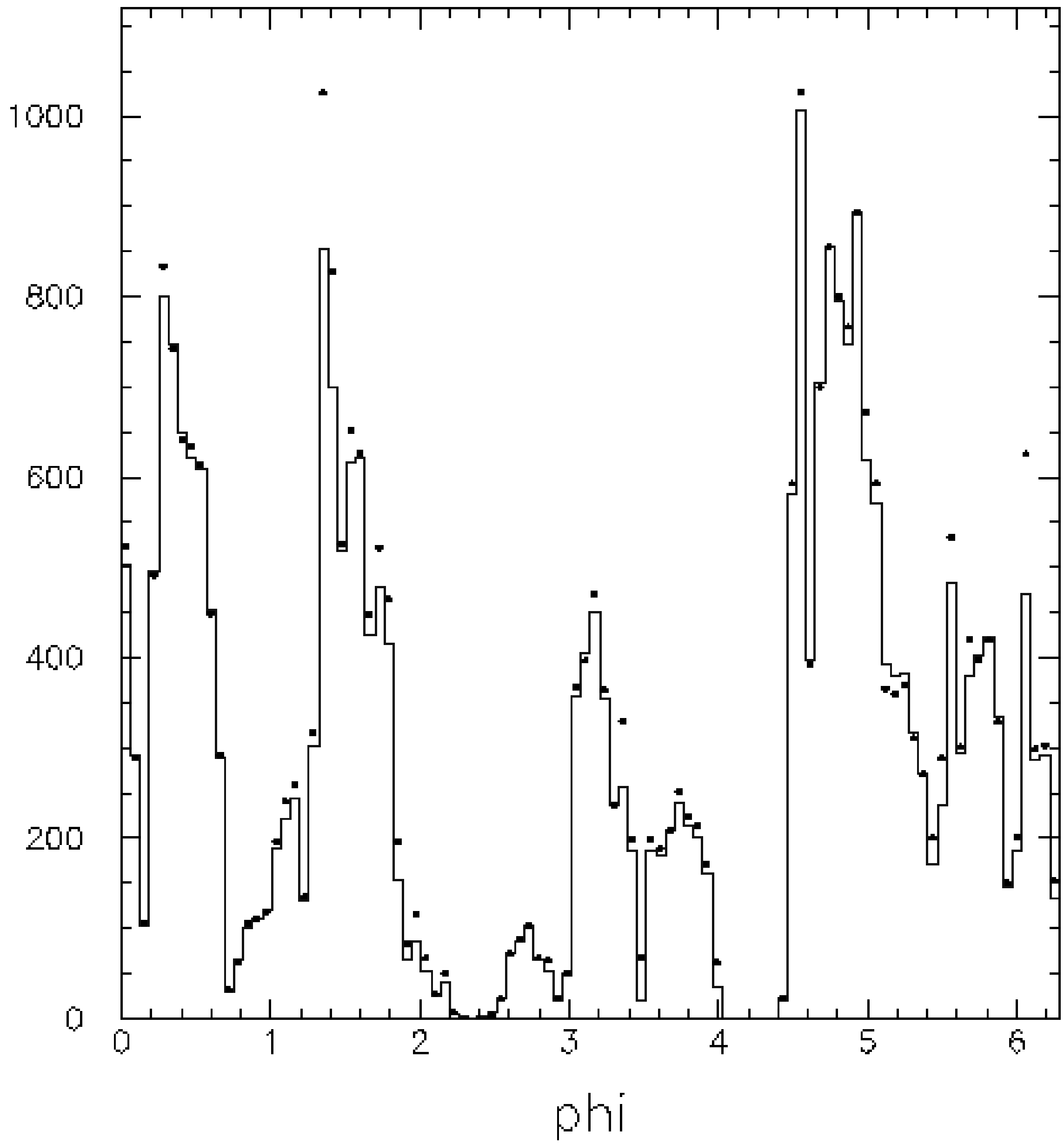


ID 101
Entries 1276
Mean -0.8771E-03
RMS 0.3547E-02
UDFLW 35.00
OVFLW 10.00
Dvx (Svtd−Obsp)
ID 102
Entries 1276
Mean -0.2108E-04
RMS 0.2920E-04
UDFLW 4.000
OVFLW 6.000
x 10^-3
Crv (Svtd−Obsp)
ID 103
Entries 1276
Mean 0.7478E-03
RMS
UDFLW 13.00
OVFLW 4.000
Phi (Svtd−Obsp)
ID 104
Entries 1276
Mean 2.007
RMS
UDFLW 0.000
OVFLW 7.000
Chisq (Svtd)

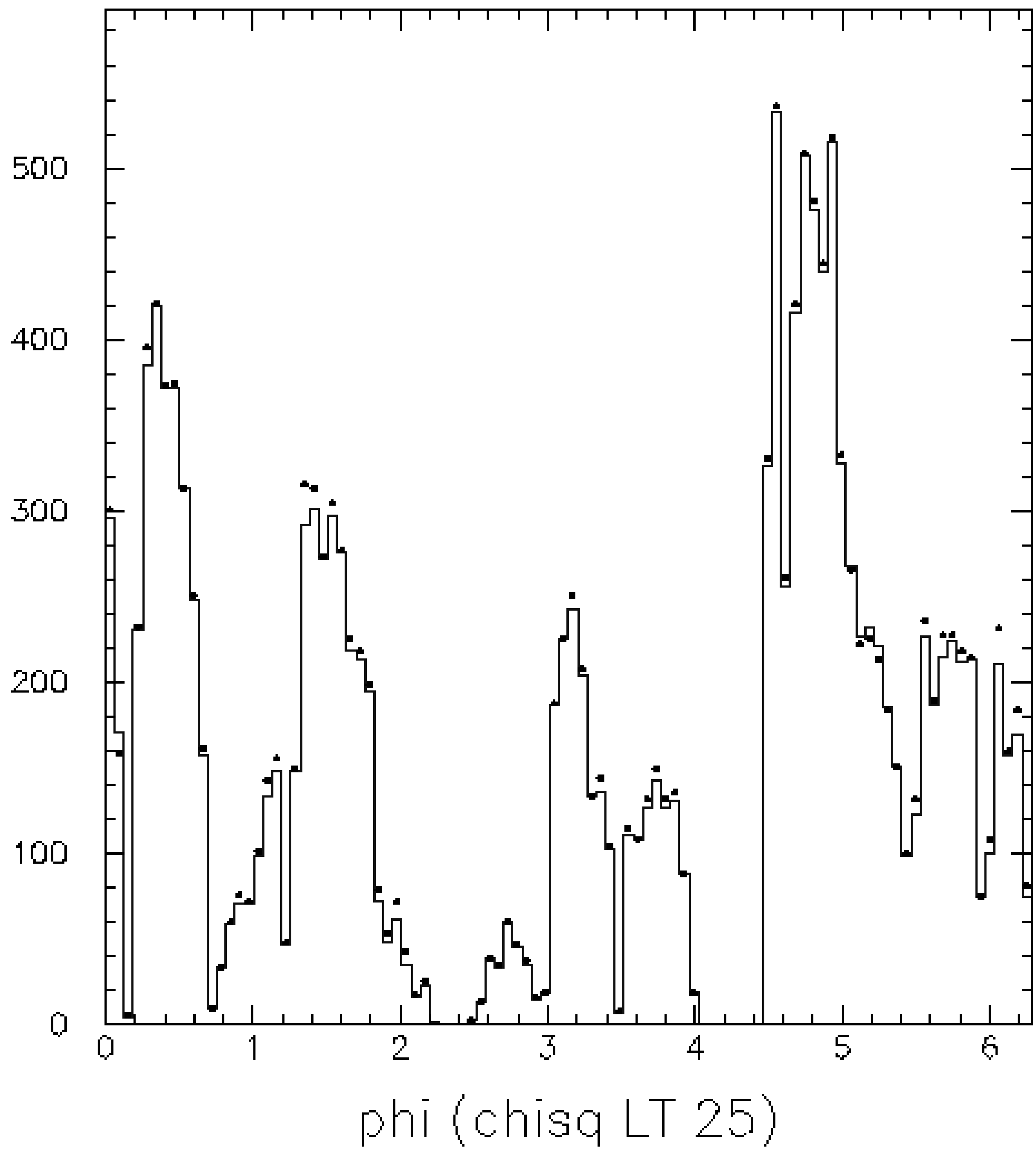

run 128449 hist=svtsim points=svt
2000
1750
1500
1250
1000
750
500
250
0
0
20
40
60
80
100
chisq



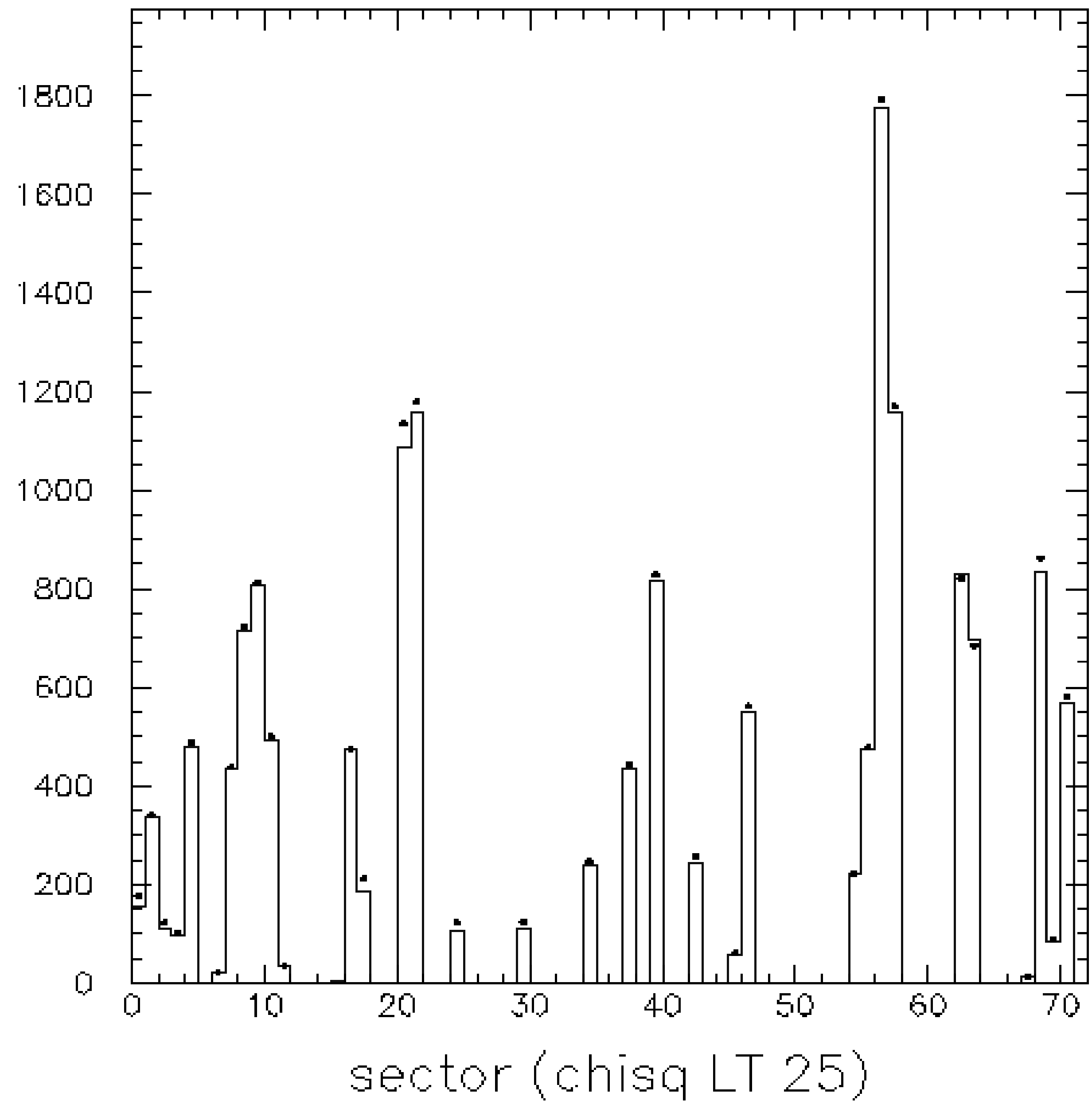
10^4
10^3
10^2
10
1
0
200
400
600
800
1000
chisq



1000
800
600
400
200
0
0
1
2
3
4
5
6
phi



0
100
200
300
400
500
0
1
2
3
4
5
6
phi (chisq LT 25)


1800
1600
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50
60
70
sector (chisq LT 25)