Available on CMS information server

CMS NOTE 2005/000

The Compact Muon Solenoid Experiment

CMS Note

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

January 23, 2006

The CMS Simulation Validation Suite

S. Abdullin[1)], P. Arce[2)], S. Banerjee[3)], H. Cheung[1)], X. Ding[4,1)], D. Elvira[1)], X. Huang[5)], A. Karchilava[6)], L. Shabalina[7)], M. Strang[6)], J. Yarba[1)]

Abstract

We describe the characteristics and functionalities of the CMS simulation validation package. This package is designed to test new releases of the CMS simulation software against the current version.

1) Fermilab, Batavia, IL, USA

2) Ciemat, Madrid, Spain

3) Tata Institute, Mumbai, India

4) Nanjing University, China

5) Univ. de Puerto Rico, Mayaguez, Puerto Rico

6) SUNY-Buffalo, Buffalo, NY, USA

7) UIC, Chicago, IL, USA

1 Introduction

The purpose of the Simulation Validations Suite (SVS) is to validate each new version of the CMS simulation software, comparing values associated with low level simulation quantities, typically derived from hits, geometry, magnetic field or other Geant4 [1] objects, with reference values. An expanded or full validation for special releases will include higher level, physics, quantities. In the future, the scope of the project will be extended to include digis and fast simulation.

2 General Description

The validation process is divided into three stages to make the process clear, and the software easy to maintain:

- Individual detector system, geometry, or field packages do "on the fly" analysis on a set of validation samples, producing ROOT [2] output files. The initial packages include: SimG4TackerValidation, SimG4EcalValidation, SimG4HcalValidation, SimG4MuonValidation, SimG4FieldValidation, and SimG4GeomValidation. Each package consists of one or more tests associated with sub-detectors within a detector system, barrel/endcap and preshower in the case of the Ecal. Different tests have been defined and configured in some sub-detectors to use either low-level or high-level simulation information in the validation. The same test may be run many times under different conditions, or samples. ROOT trees with pre-processed information is written into ROOT browsable files. This information typically includes bare hit quantities or complex ones derived from hits or Geant4 objects.
- Macros are run on the ROOT trees and process information into validation objects, such as numbers in ASCII files and/or histograms.
- OVAL is the integration tool, used for launching the Suite from the SimG4Validation directory. Histogram comparison tools, such as the Statistical Tool Kit [5] or ROOT is used to compare the "current" histograms values with those stored in reference files. The differences are stored in ASCII files.
- OVAL is used to find differences between the ASCII files with current difference information and reference ASCII files.

3 Sub-detector Package Description

3.1 Tracker

3.1.1 Description

The `SimG4TrackerValidation` package is intended to test simulation of the tracking system comprised of the Silicon Strip Tracker and Pixel Detector System. The former consists of the Tracker Inner Barrel (TIB), Tracker Outer Barrel (TOB), Tracker Inner Disks (TID) and Tracker End Cap (TEC) while the latter includes the Pixel Barrel and Pixel End Cap. (The Pixel End Cap is also referred to as the Forward Pixel Detector.) The tests are performed for each subsystem separately due to the difference in their geometry.

The `SimG4TrackerValidation` package accesses collections of hits belonging to different subsystems through `TrackerHitsObject` and produces a ROOT tree with the simulated hit information.

Simple analyses are implemented in ROOT macro scripts to produce histograms associated with the final quantities to validate. In the `SimG4TrackerValidation` package, samples may be generated using the "Particle Gun", or read from an HBOOK file (HepEVT format) using the "Ntuple Reader". The two available options are controled from the OvalFile file.

3.1.2 ROOT Tree Content

One assistant class, `SimHitTrackerTree` provides the interface to ROOT. The information stored in the ROOT tree is divided into two branches:

1. Global Information

   Event Number
   Run Number

Total Number of Incident Particles
Incident Particle type
Incident Particle Energy
Incident Particle Momentum
Incident Particle η
Incident Particle ϕ
Number of vertices
Vertex coordinates
Total Number of Hits

2. Hit information

   Subsystem type
   ID of the detector unit (in Geant internal numbering)
   Hit position (entry and exit point coordinates in the local system, local position and direction)
   Deposited Energy of Hit
   ID of the track that produced a hit
   Process and particle momentum and type
   Time-of-flight

3.1.3 Validation Quantities

The comparison tests are performed on the following quantities, constructed from the ROOT tree leaves:

- Sample: Single muons, electrons or pions with p_T =15 GeV, 1500 events in 12 bins of η in the range $-3 < \eta < 3$:

- Quantities

```
Energy deposition
Distribution of track entry and exit points in x, y and z
Distribution of local x and y track entry point coordinates
Number of hits in each subsystem
```

Test variables are plotted separately for each subsysytem. Examples of Kolmogorov-Smirnov test output are presented in Figs.1-12.

3.1.4 Run Configurations

The validation is performed currently at one level for each tracker subsystem. But the package can easily accomodate running more tests with different input generated events. These possible tasks are controled via the OVAL tool. An OvalFile control file in the `test` sub-directory under `SimG4TrackerValidation` is used to configure the package.

The test using muons is started using the following command:

```
Ovalv1 run TrackerTestEtaBinMuon
```

3.1.5 Validation Tests

OVAL is also used to detect differencies between current and reference values stored in two separate histogram files. Statistical tools available in ROOT - `UnBinned KSTest` and `Binned Chi2Test` - are used to compare the sample and the reference histograms.

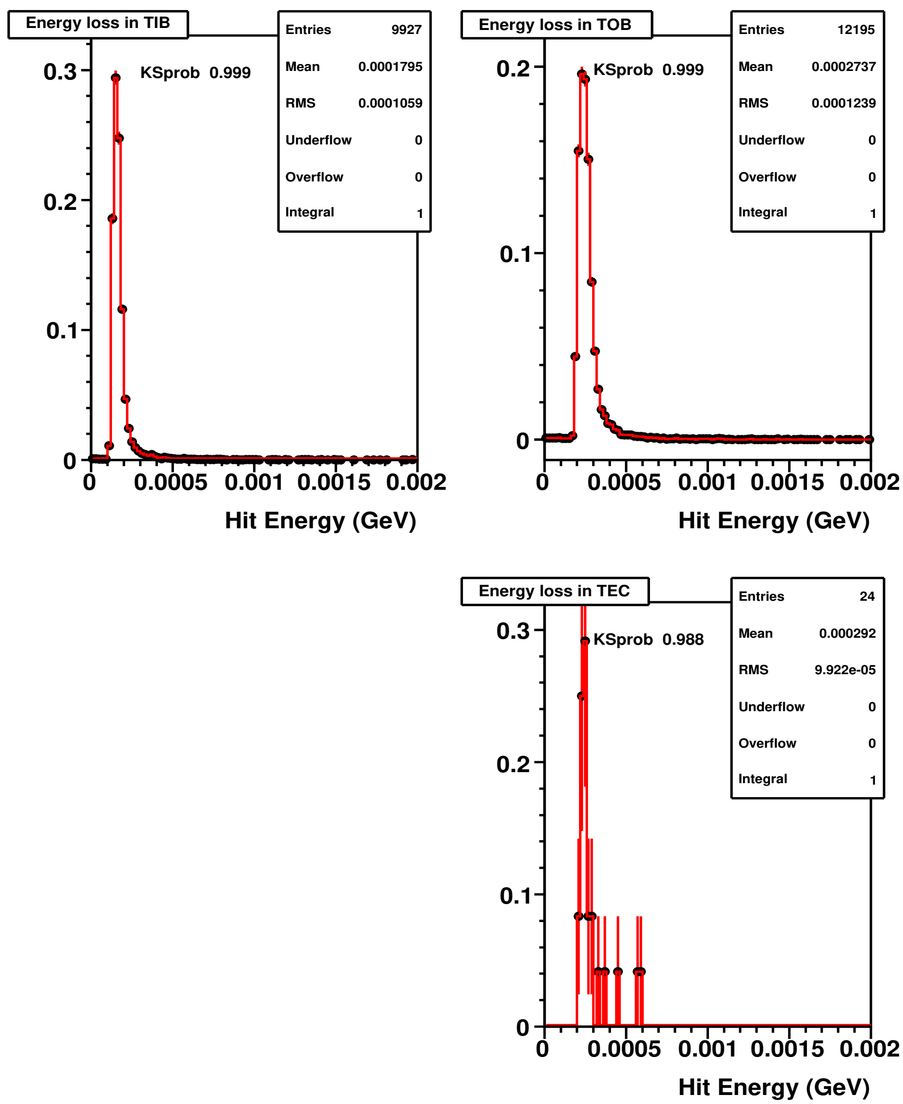

Figure 1: Kolmogorov-Smirnov test example to compare energy loss in different subsystems of the Silicon Tracker.

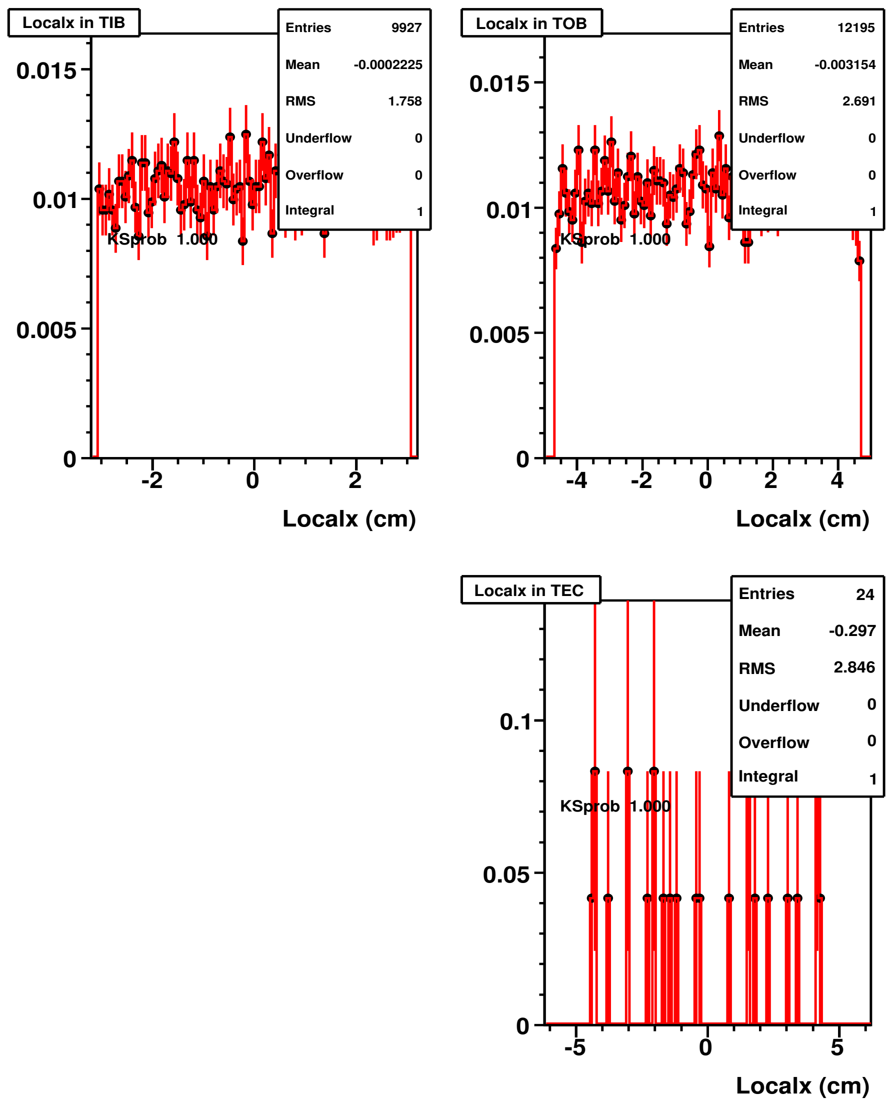

Figure 2: Kolmogorov-Smirnov test example to compare local x coordinates of the hits in different subsystems of the Silicon Tracker.

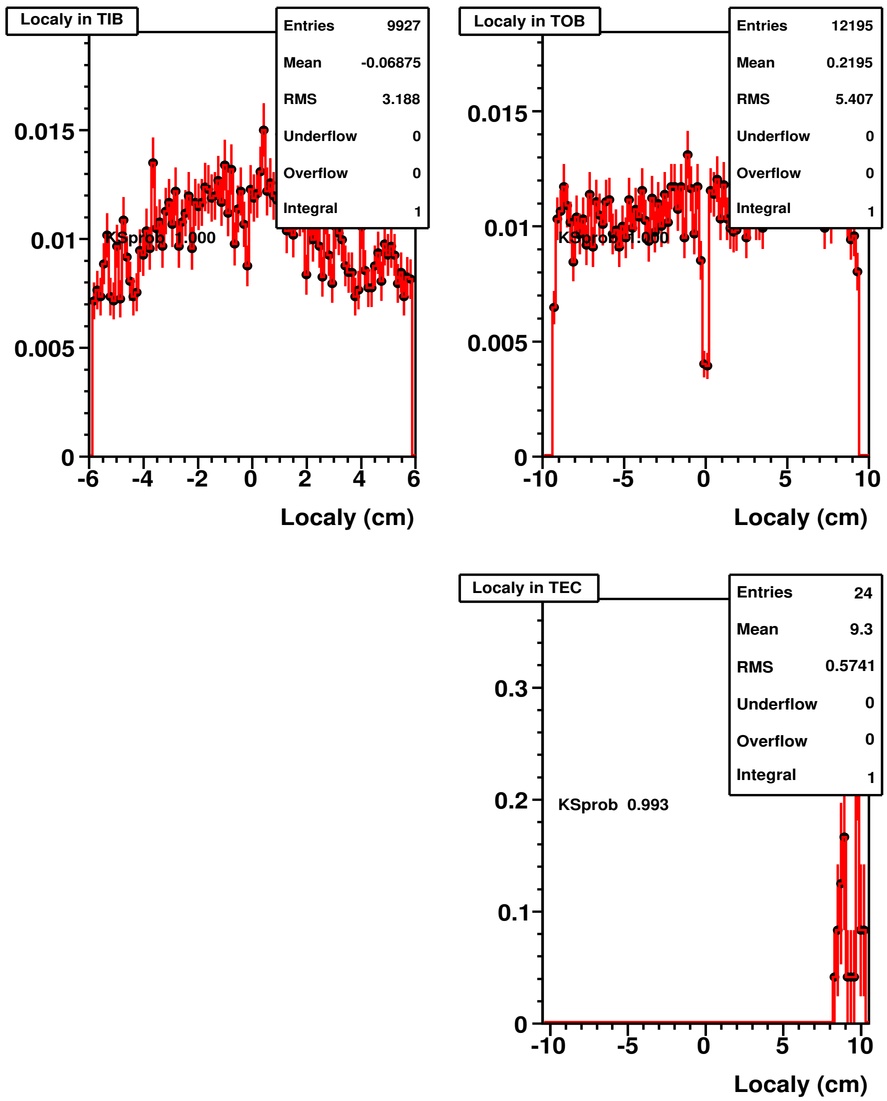

Figure 3: Kolmogorov-Smirnov test example to compare local y coordinates of the hits in different subsystems of the Silicon Tracker.

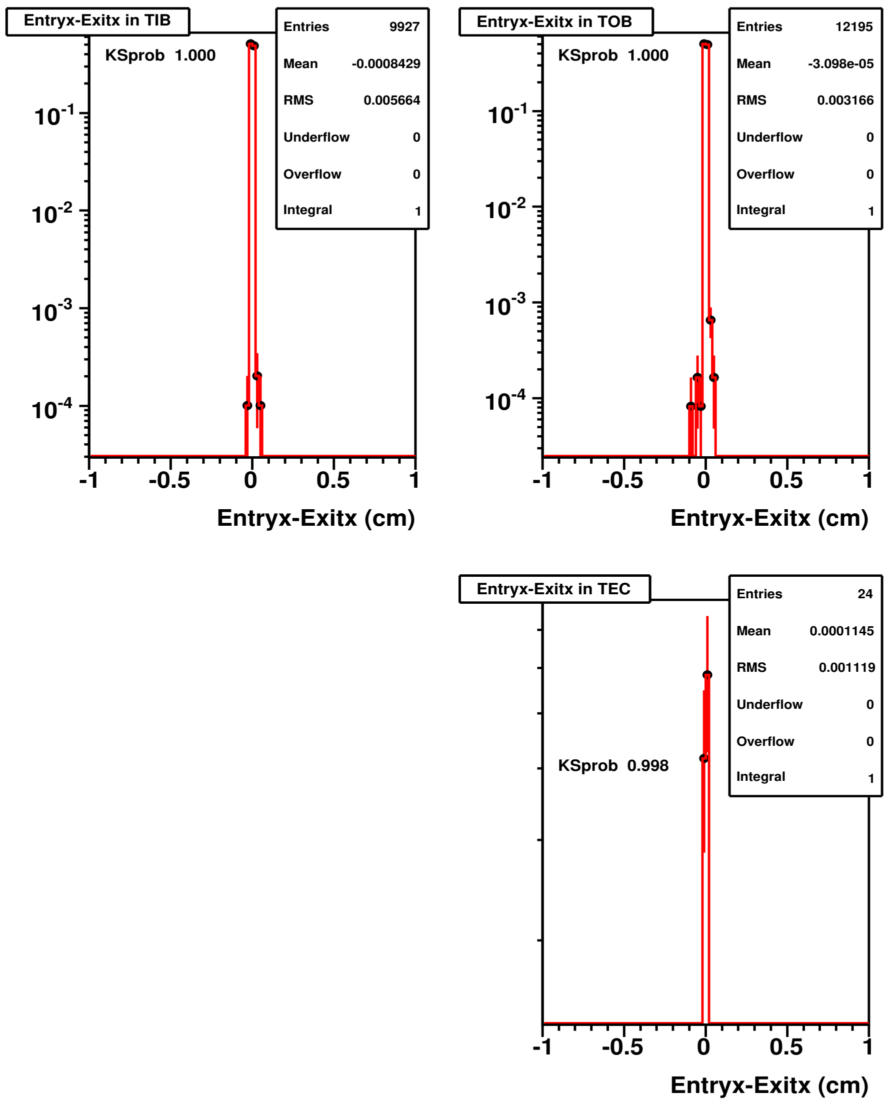

Figure 4: Kolmogorov-Smirnov test example to compare the difference in entry and exit points along the local x-axis of the tracks in different subsystems of the Silicon Tracker.

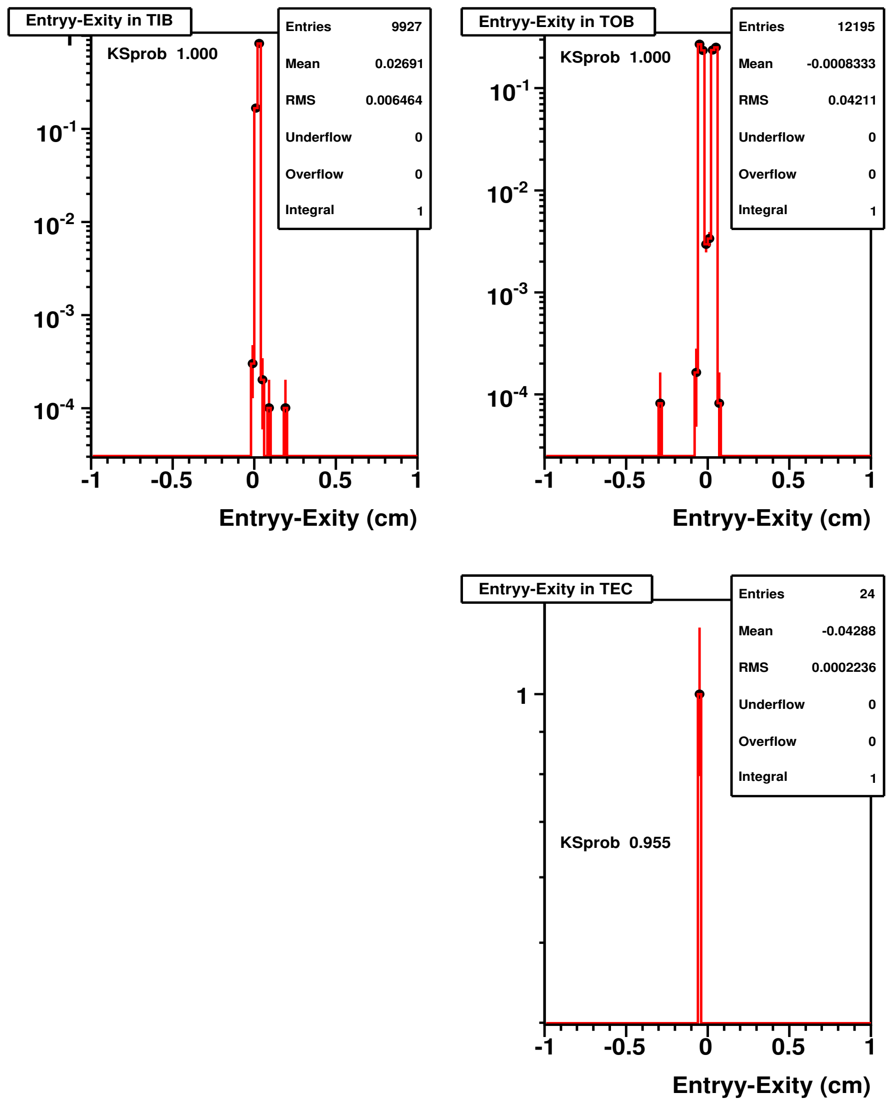

Figure 5: Kolmogorov-Smirnov test example to compare the difference in entry and exit points along the local y-axis of the tracks in different subsystems of the Silicon Tracker.

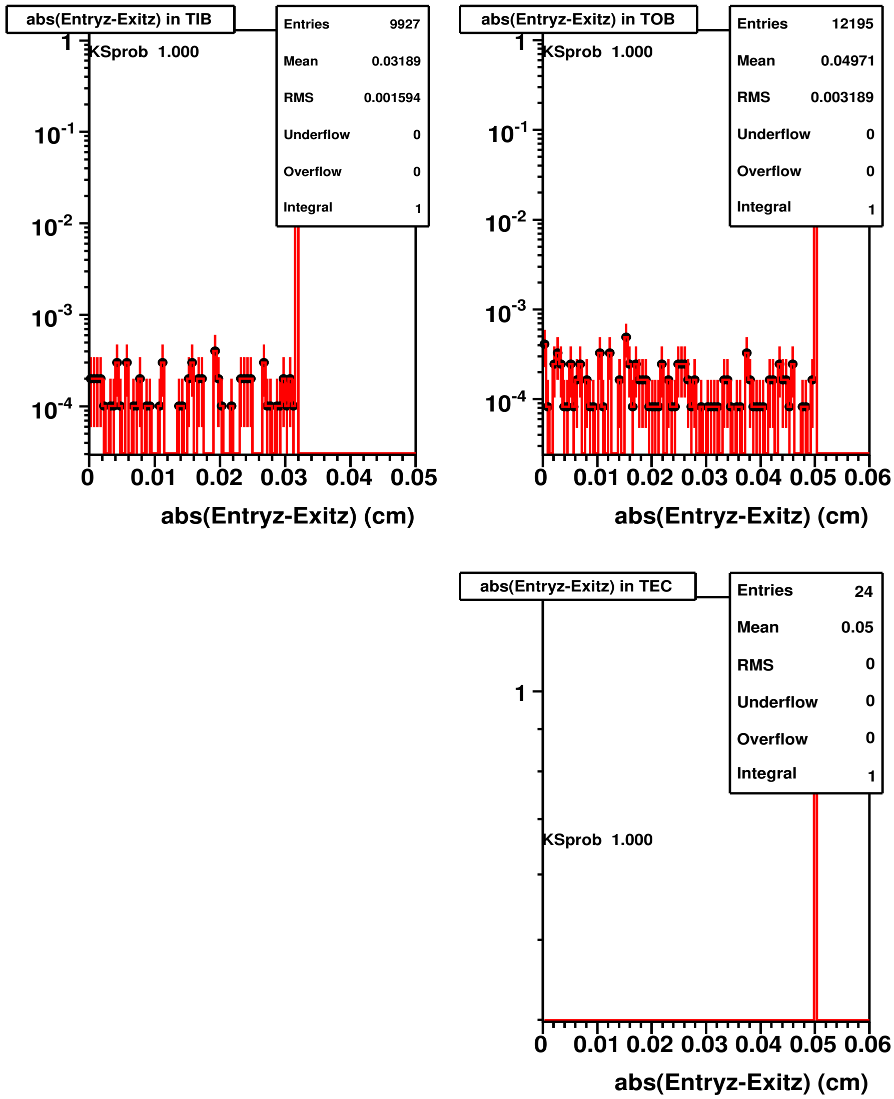

Figure 6: Kolmogorov-Smirnov test example to compare the difference in entry and exit points along the local z-axis of the tracks in different subsystems of the Silicon Tracker.

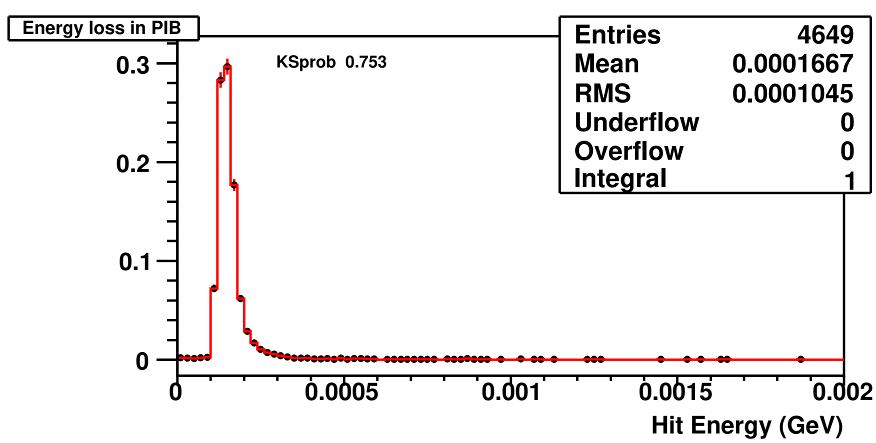

Figure 7: Kolmogorov-Smirnov test example to compare energy loss in different subsystems of the Pixel Tracker.

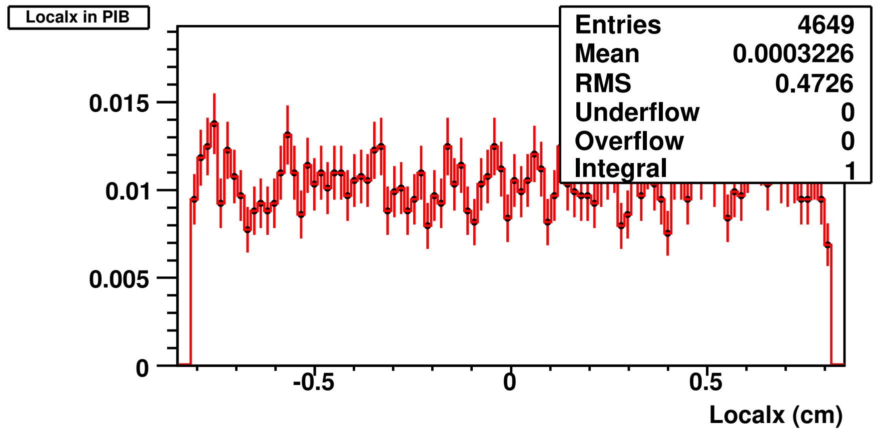

Figure 8: Kolmogorov-Smirnov test example to compare local x coordinates of the hits in different subsystems of the Pixel Tracker.

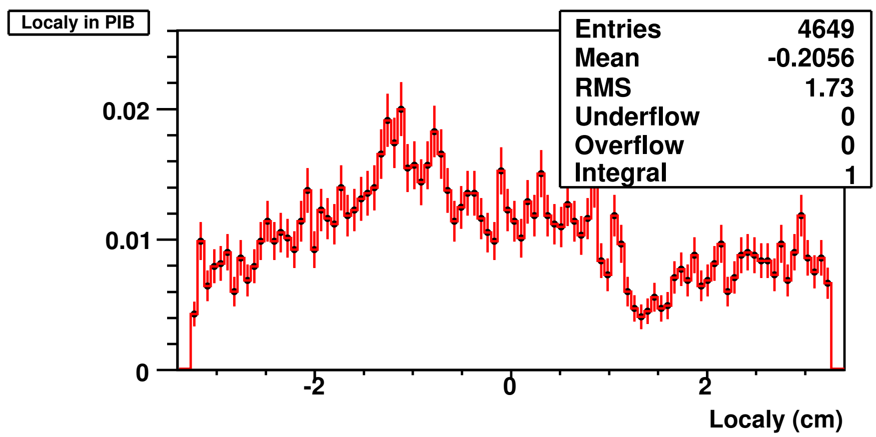

Figure 9: Kolmogorov-Smirnov test example to compare local y coordinates of the hits in different subsystems of the Pixel Tracker.

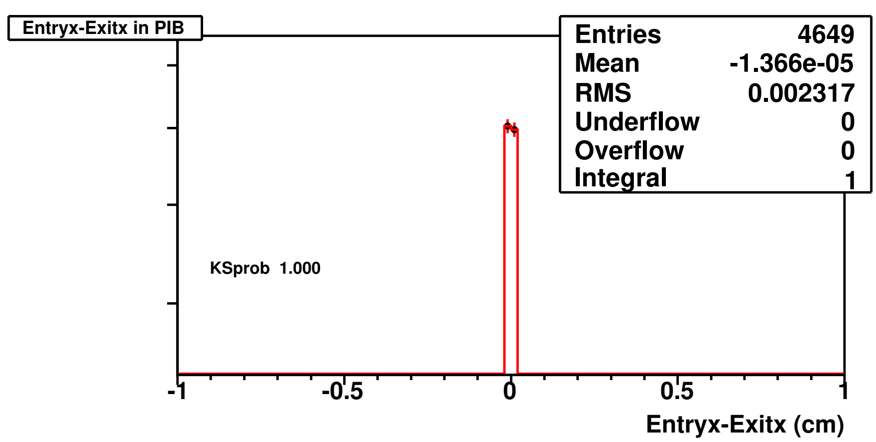

Figure 10: Kolmogorov-Smirnov test example to compare the difference in entry and exit points along the local x-axis of the tracks in different subsystems of the Pixel Tracker.

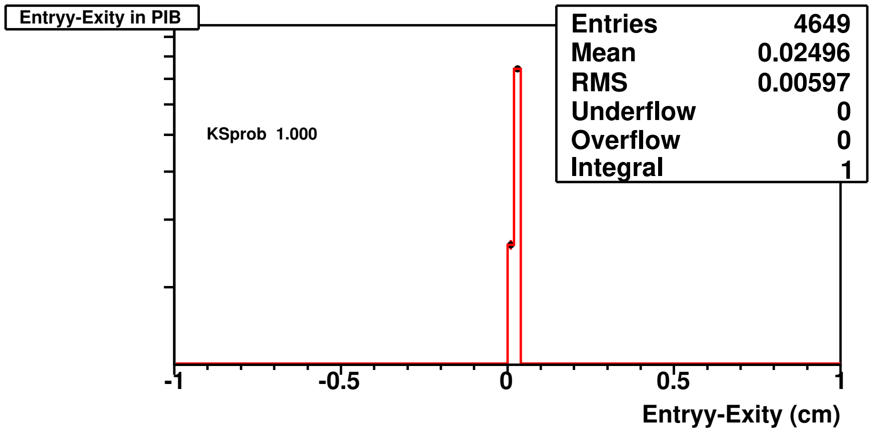

Figure 11: Kolmogorov-Smirnov test example to compare the difference in entry and exit points along the local y-axis of the tracks in different subsystems of the Pixel Tracker.

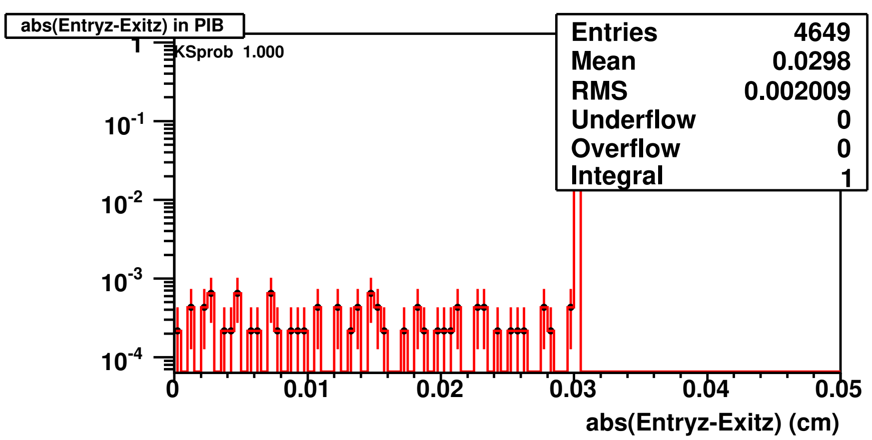

Figure 12: Kolmogorov-Smirnov test example to compare the difference in entry and exit points along the local z-axis of the tracks in different subsystems of the Pixel Tracker.

3.2 Electromagnetic Calorimeter

3.2.1 Description

The `SimG4EcalValidation` package is associated with the Electromagnetic Calorimeter (ECAL), both the barrel and the endcaps, as well as with the Pre-Shower sub-detector. The two tests implemented in `SimG4EcalValidation` are: `ECalorimeter`, and `PreShower`. Both tests perform a preliminary on-the-fly analysis of basic simulation quantities, hits and `G4Step` objects, and store the results in a ROOT tree contained in a ROOT file. Simple analyses are implemented in ROOT macro scripts to produce histograms associated with the final quantities to validate. In the `SimG4EcalValidation` package, samples may be generated using the "Particle Gun", or read from an HBOOK file (HepEVT format) using the "Ntuple Reader". The two available options are controlled from the C-Shell script file.

3.2.2 ROOT Tree Content

One assistant class, `SimHitEcalTree` provides the interface to ROOT. The information stored in the ROOT tree is divided into three branches:

1. Global Information

```
Event Number
Run Number
Total Number of Incident Particles
Particle Type of Every Incident Particle
Energy of Every Incident Particle
Momentum of Every Incident Particle
Vertex coordinates
Total Number of Hits in Barrel Calorimeter
Total Number of Hits in EndCap Calorimeter
Total Energy deposited in Barrel Calorimeter
Total Energy deposited in EndCap Calorimeter
Energy deposited in 1x1 crystal cluster
Energy deposited in 2x2 crystal cluster
Energy deposited in 3x3 crystal cluster
Energy deposited in 4x4 crystal cluster
Energy deposited in 5x5 crystal cluster
Energy deposited by EM Particles
Energy deposited by Hadrons Particles
```

2. Hit information

```
Hit position: etaHit, phiHit
Energy deposited of every Hit: energyEMHit, energyHadrHit
Global time
```

3. Step Information

```
Energy deposited in every X0 (radiation length): EX0[25].
```

3.2.3 Validation Quantities

The comparison tests are performed on the following quantities, constructed from the ROOT tree leaves:

- Lower-Level Validation for ECalorimeter
  - Sample: Photon; 30GeV; 2000 Events
  - Quantities

```
Plot the occupancy (eta versus phi)
Plot 1-D histogram of E1, E4, E9, E16, E25
Plot 1-D histogram of Ratio: E1/E4, E4/E9, E9/E16, E16/E25,
                             E1/E25, E9/E25
Plot the percentage of Energy deposited in Barrel and End-Cap
Plot the longitudinal development of Shower with a single Energy
```

- Higher-Level Validation for ECalorimeter
  - Sample: Photon; 10GeV, 20GeV, 30GeV, 40GeV, 50GeV; 2000 Events
  - Quantities

```
Plot the E25 resolution versus the energy of incident particle
Plot the longitudinal develpment of Shower with multi-Energy
```

- Lower-Level Validation for PreShower
  - Sample: Photon; 30GeV; 2000Evnets
  - Quantities

```
Plot the occupancy (eta versus phi)
Plot 1-D histogram of E1, E4, E9, E16, E25
Plot 1-D histogram of Ratio: E1/E4, E4/E9, E9/E16, E16/E25,
                             E1/E25, E9/E25
Plot the percentage of Energy deposited in Barrel, End-Cap
   and PreShower
Plot the longitudinal development of Shower with single Energy
```

- Higher-Level Validation for PreShower
  - Sample: Photon; 10GeV, 20GeV, 30GeV, 40GeV, 50GeV; 2000 Events
  - Quantities

```
Plot the E25 resolution versus the energy of incident particle
Plot the longitudinal development of Shower with multi-Energy
Plot the coefficient of Lead absorption versus the energy of
incident particle
```

3.2.4 Run Configurations

The validation is performed currently at two levels for each of the ECAL sub-systems: barrel/endcaps, and preshower detectors. The package can therefore be run in Low-Level or High-Level mode for the barrel/endcaps and for the pre-shower. These four possible tasks are controled via the OVAL tool. An OvalFile control file in the `test` sub-directory under `SimG4EcalValidation` is used to configure the package. The user can choose any individual validation level or all using one of the following commands:

```
Ovalv1 run EcalTest.csh.Lower-Level
Ovalv1 run EcalTest.csh.Higher-Level
Ovalv1 run PreShowerTest.csh.Lower-Level
Ovalv1 run PreShowerTest.csh.Higher-Level
Ovalv1 run
```

From above, it can be seen there are two C-Shell scripts: `EcalTest.csh` and `PreShowerTest.csh`. These define the concrete task for two levels of validation via setting the number of jobs, the location of input data, renaming the root files and making histogram comparison between current histograms and reference histograms respectively.

3.2.5 Validation Tests

In addition to the role in controling the tests, OVAL can also be used to detect differencies between current and reference values stored in four separate ASCII files below.

```
EcalTest.csh.Lower-Level.ref
EcalTest.csh.Higher-Level.ref
PreShowerTest.csh.Lower-Level.ref
PreShowerTest.csh.Higher-Level.ref
```

The Statistics Toolkit is used to test the consistency between the contents of current and reference histograms. The class `SmartTestHisto` is introduced to perform this task, by reading the data stored in the reference ROOT file and the sample ROOT file . If the test requires binned data, histograms are created from the ROOT tree information, otherwise, unbinned data are stored in vectors. In both cases, the Statistics Toolkit classes are used to perform the comparison and output the results to the oval log files. As currently implemented the `SmartTestHisto` class is used as an alternate option to the ROOT macro analysis. In the example provided under the `SimG4EcalValidation/test` directory, `Compare.cc`, two methods, `compareWithChi2Test()` and `compare2DWithChi2Test()`, are used to compare the sample histograms with the reference histograms.

Currently, all the quantities to validate at the lower-level (as given in 3.2.3) have been applied to the comparison with the Chi2Test method between the sample data and the reference data.

Finally, all the results will be saved to two postscript files. Examples of these are shown in Figs.13 and 14. The 1-D histograms from the sample and the reference are drawn in the same plot to aid in evaluating the differences between them, while for the 2-D histograms, such as the longitudianl profile and occupancy, are drawn in two plots separately. Blue is used for the reference data and red for the sample data. PV stands for the probability value calculated with the Chi2Test method. Because currently the sample histograms and the reference histograms are generated with the same version of OSCAR, the histograms of the two distributions are exactly same and all values of PV are `"1"`.In the future, the sample histograms will come from new versions of OSCAR (or CMSSW) and the PV values will give the differences between the results of the different versions.

3.3 Hadronic Calorimeter

3.3.1 Description

The `SimG4HcalValidation` package collects different type of information about HCAL simulated hits, using both `EndOfEvent` and `G4Step` G4-observers. It also performs some NxN cells cluster analysis and even cluster finding with cone algorithm. The collected information is stored in a ROOT tree. A csh script `job_ntuple.csh` performs the code running under OVAL, plots the histograms and compares some of them with the those in the reference file by means of a ROOT macro. The input data sample contains 1000 single pions with p=30 GeV shot at the center of a particular tower in the HCAL barrel with (η,φ)= (4,4).

3.3.2 ROOT Tree Content

The ROOT tree is divided into several branches. There is a switch which enables the information to be stored at three levels of complexity. These three levels is denoted as L1, L2, L3 and the description of each block of information will be denoted by a corresponding label.

The most essential primary information is collected in the block of CaloSimHit information (L1):

- nHits - number of hits;
- layerHits - number of HCAL layer to which the hit belongs;
- etaHits - hit η;
- phiHits - hit φ;
- eHits - hit energy;
- tHits - hit time;

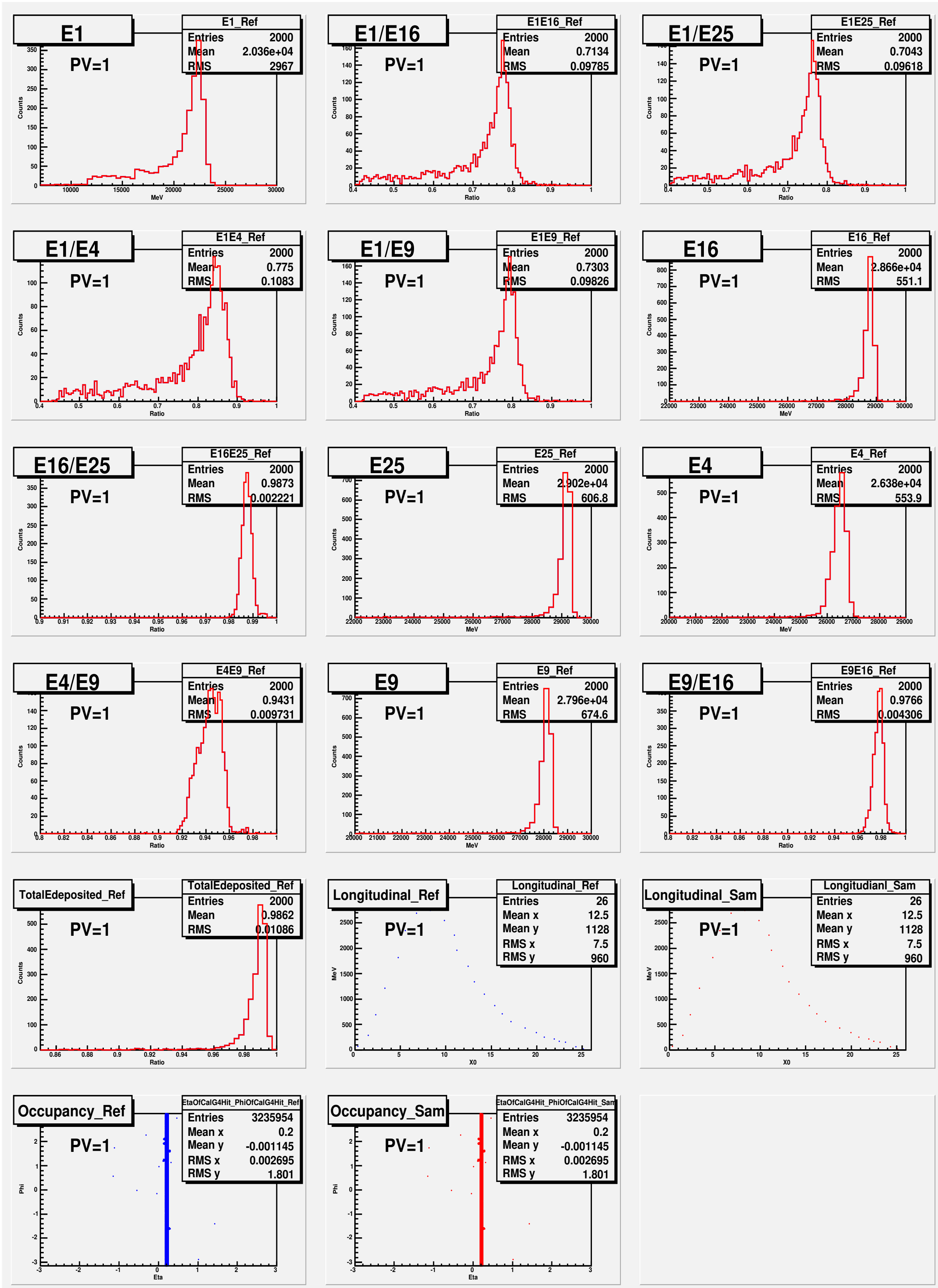

Figure 13: The Results of Comparison with the Chi-Square Test for ECAL.

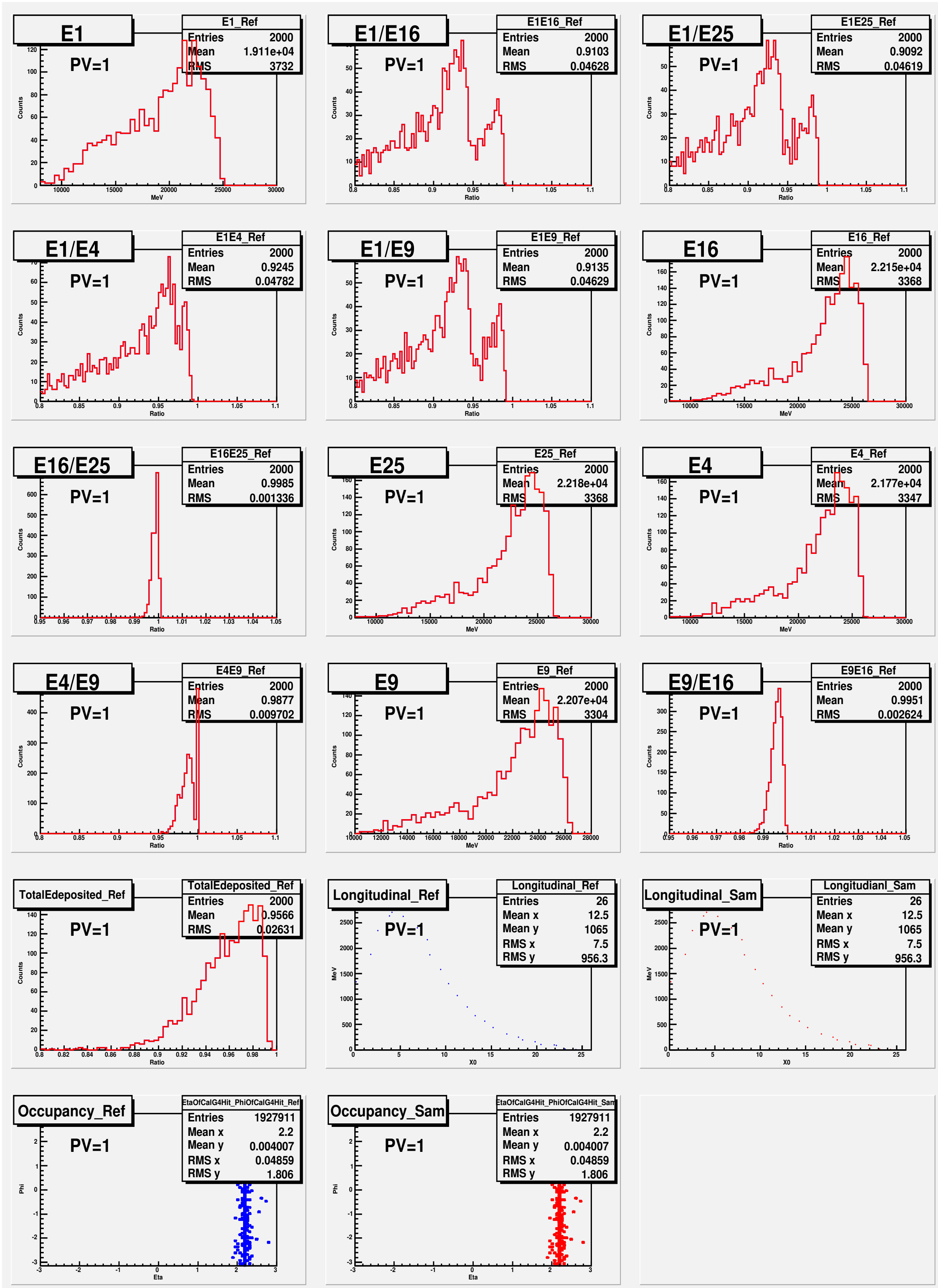

Figure 14: The Results of Comparison with the Chi-Square Test for Preshower.

- idHits - hit ID number;
- jitterHits - hit jitter with respect to the expected time of arrival.

There is a block of information dedicated to both cluster of cells within R=0.5 cone and 7x7 cells (hardwired number, works currently only for HCAL barrel) in the evaluated around user-defined η and φ direction, see Eta0 and Phi0 description in the next subsection, (L2):

- ecalNxNr - ECAL fraction in the cone within R= 0.5 around given direction;
- hcalNxNr - *idem* for HCAL;
- hoNxNr - *idem* for HO separately;
- etotNxNr - *idem* for the ECAL+HCAL (redundant information);
- ecalNxN - ECAL fraction in the 7x7 cluster around given direction;
- hcalNxN - *idem* for HCAL;
- hoNxN - *idem* for HO separately;
- etotNxN - *idem* for the ECAL+HCAL (redundant information).

There is a block of information about transverse NxN cluster profiles in the HCAL barrel (HCAL 1x1, 3x3, 5x5 and 7x7 cells hardwired) around user-defined η and φ direction (L2):

- nIxI - number of hits;
- iIxI - sequential number of the square to which the hit belongs;
- eIxI - energy of the hit;
- tIxI - arrival time of the hit.

There is a block containing information about the energy deposit in the HCAL scintillator layers and depths (groups of layers) as a sum over all η and φ cells (L2):

- nLayers - number of layers;
- eLayer - energy in each layer;
- nDepths - number of depths;
- eDepth - energy in each depth;
- eHO - energy in HO;
- eHBHE - sum of the energy in HB and HE.

Information about HF subdetector (L2) :

- elongHF - energy in the long fibers of HF;
- eshortHF - *idem* for short fibers;
- eEcalHF - energy collected in the entire ECAL;
- eHcalHF - *idem* for HCAL (barrel and endcap only).

Block of information about the highest-E_T jet (L3) :

- nJetHits - number of CaloSimHits in the jet (integer);

- rJetHits - distance (in $\eta - \varphi$ space) of each CaloSimHit from the center of the jet;
- tJetHits - arrival time of each CaloSimHit;
- eJetHits - energy of each CaloSimHit.

Then there are global variables for this highest-E_T jet (L3):

- ecalJet - ECAL fraction of the jet energy;
- hcalJet - *idem* for HCAL;
- hoJet - *idem* separately for outer calorimeter (HO);
- etotJet - total jet energy (redundant information);
- detaJet - η distance from "nominal value" (see Eta0 description in the next subsection);
- dphiJet - *idem* for φ;
- drJet - *idem* for distance.

The latter three variables are used in case of the single pion shooting at the fixed η and φ position to measure the cluster deviation from the "nominal" position.

There is a block containing information about all the jets above defined E_T threshold (L3) :

- nJets - number of jets;
- jetE - energy of each jet;
- jetEta - each jet η;
- jetPhi - each jet φ.

In addition, there is a special variable containing a mass of the two highest- E_T (if any) - dijetM.

3.3.3 Validation Quantities

The validation of the new version of the simulation is assumed to be done on the basis of a comparison of several histograms (among more than 70 plotted in total) with those previously stored in the reference file using the χ^2 method of ROOT. There are 6 histograms for energy deposition in the first 6 scintillator layers closest to the ECAL, time distribution of all SimHits and energy-weighted time distribution of SimHits in the 7x7 matrix of ECAL+HCAL around the pion entry point. Also included in the comparison are distributions of the number of SimHits in the ECAL and HCAL. All other histograms contain additional information which might be required for investigations in case of signifant discrepanceis revealed by the main comparison test. Figures 15 and 16 show examples of two distributions among the 10 selected for comparison tests: the deposited energy in the HCAL scintillator just behind ECAL and the energy-weighted distribuion of the time of hits in the 7x7 matrix of ECAL+HCAL towers around the pion direction.

3.3.4 Run Configurations

In the test area the default OSCARconfiguration.xml includes only ECAL and HCAL in the simulation. Also the magnetic field if by default switched off in the `.orcarc` file (actually named as hcalValidRC in the test area).

There are 11 external parameters which can be defined in the `.orcarc` file cards listed in Table 1.

In the cluster analysis there is a choice to take into account both ECAL and HCAL energy, or only the HCAL one (default). The corresponding control card is the first one in Table 1. The ECAL energy is always taken as it is, while the HCAL energy for cluster analysis can be taken as either original CaloSimHit energy (default) or the one multiplied by the corresponding sampling factors (same as in ORCA), card 2 in Table 1.

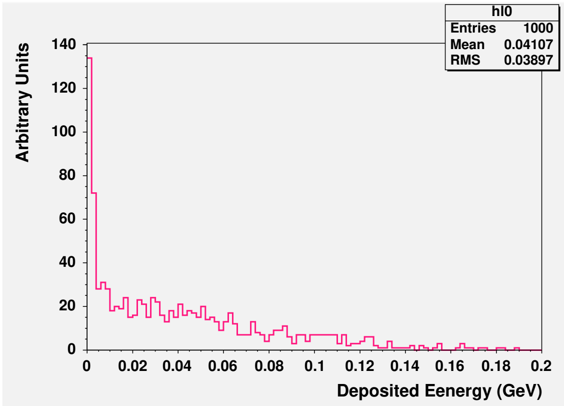

Figure 15: Energy deposition of the pion shower in the layer 0 (closest to ECAL).

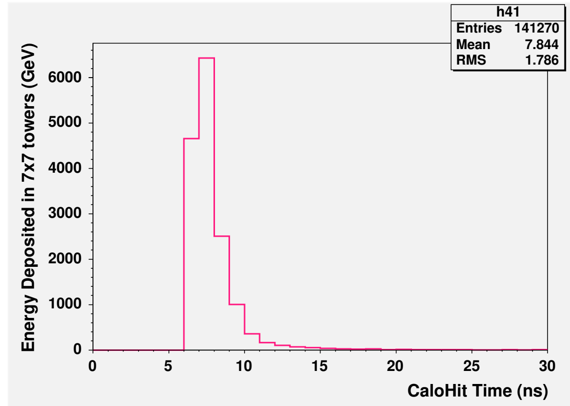

Figure 16: Energy-weighted timing of hits in the vicinity of pion direction.

Table 1: Set of SimpleConfigurable parameters defined as control cards

Card nubmer	Variable type	Defalut value	Key name in .orcarc file
1	bool	false	SimG4HcalValidation:HcalClusterOnly
2	bool	true	SimG4HcalValidation:HcalSampling
3	float	0.5	SimG4HcalValidation:ConeSize
4	float	1e-20	SimG4HcalValidation:EcalHitThreshold
5	float	1e-20	SimG4HcalValidation:HcalHitThreshold
6	float	0.	SimG4HcalValidation:TimeLowLimit
7	float	999.	SimG4HcalValidation:TimeUpLimit
8	float	5.0	SimG4HcalValidation:JetThreshold
9	float	0.0	SimG4HcalValidation:Eta0
10	float	0.0	SimG4HcalValidation:Phi0
11	int	2	SimG4HcalValidation:InfoLevel

Card 3 defines the cone size of the jetfinder. Cards 4 and 5 set non-zero minimal CaloSimHit energy for the ECAL and HCAL respectively. Cards 6 and 7 define the time window for the CaloSimHit arrival. Cut 8 sets a cut on minimal jet energy (the default value, 5 GeV, is set for HcalSampling = 1). Cards 9 and 10 stand for user-defined η and φ direction (for single pion shooting at the fixed position). Card 11 defines which blocks of information described in the previous subsection are stored in the ROOT tree.

3.3.5 Validation Tests

`OvalFile` control file in the `SimG4HcalValidation`test directory contains the configuration of the test job. It just defines the `job_ntuple.csh` as the main test script and assigns a default parameter `SimG4HcalValidation:InfoLevel=2` (from Table 1) to it.

3.4 Geometry

3.4.1 Description

The `SimG4GeomValidation` package writes out the number of volumes and materials for comparing them with a reference version using the OVAL tool. It also tracks particles with neither physics nor magnetic field and compares the total number of radiation lengths traversed until the end of the detector event-by-event.

If desired, it also constructs a few histograms and a ROOT tree with material budget information (number of radiation lengths), and a text file with accumulated material budget step-by-step.

3.4.2 Validation quantities

There are two different kinds of variables printed for testing with the OVAL tool:

- The geometry summary, that is the total number of different materials in the simulation, and the total numbers of logical volumes, physical volumes and touchables (copies of each volume).
- The number of radiation lengths after traversing all the detector. The events are read from an ntuple `/afs/cern.ch/cms/geant4rep/genntpl/murandom_disp-p10000.ntpl`, that contains one thousand muons of energy 10 TeV with random η in the range $-5.0 < \eta < 5.0$ and random ϕ, and starting at a displaced vertex $(x = 8.3, y = 1.5, z = 6.3)$ mm, to avoid as much as possible tracking along volume boundaries. In fact these muons behave as *geantinos*, because there is no magnetic field, and all the physics processes except transportation are switched off.

3.4.3 Material budget information

As well as running the test, the user may choose to produce some output in the form of histograms, ROOT tree or text file for help in understanding the differences with respect to the reference version.

Material Budget Histograms

If the parameter `TestGeometry:writeHistos` is set, a set of ROOT histograms is written into in a file whose name is defined by this parameter:

- 1D profile histogram of material budget vs η
- 1D histo η
- 1D profile histogram of material budget vs ϕ
- 1D histo ϕ
- 2D profile histogram of material budget vs eta and phi
- 1D histo η vs ϕ

Material budget tree

If the parameter `TestGeometry:writeMBTree` is set, material budget information will be written into a ROOT tree in a file whose name is defined by this parameter. The tree contains for each track:

- MB: Total material budget (sum of step length divided by radiation length for all steps)
- ETA: η direction at start of track
- PHI: ϕ direction at start of track

If `TestGeometry:TreeAllSteps` is set to 1, also the information for each step is stored (a maximum number of 5000 steps per track are stored):

- DELTAMB: accumulated material budget

- X: X position
- Y: Y position
- Z: Z position
- VOLUID: volume ID (constructed in alphabetical order)
- MATEID: material ID (constructed in alphabetical order)

Material budget Text File

If the parameter `TestGeometry:writeMBTxt` is set, an ASCII file is filled with step-by-step information about the volume traversed and the accumulated material budget until this step. The name of the file is defined by this parameter.

The file contains for each track a first line with

- Track number
- Eta direction at start of track
- Phi direction at start of track

then for each step:

- Accumulated track length (mm)
- Volume name [1)]
- Volume copy number
- Accumulated material budget (Number of radiation lengths)
- Material radiation length (rad. length / mm)

and at the end of track:

- Accumulated material budget

3.4.4 Run configuration

The `oval prod` command will run the test and compare the current values with those in a reference file, `testGeometry.ref`. Several parameters control the extra output in the form of histograms, ROOT tree or text file as explained above.

3.5 Field

3.5.1 Description

The `SimG4FieldValidation` package checks the tracking in the CMS magnetic field. The test compares the deviation at the end of the track in position and direction.

[1)] The volume that is printed corresponds to the volume traversed in this step, which is different to the volume printed by "/tracking/verbose" that is the volume the track is going to enter after this step.

3.5.2 Validation quantities

The ntuple /afs/cern.ch/cms/geant4rep/genntpl/test_field.ntpl is used for this test. It contains 400 single muon events in four groups of 100 events with different energies: 1, 10, 100, 1000 GeV. The muons are run as charged geantinos, given that the physics processes are switched off.

The deviation of these tracks after traversing all CMS is compared with a reference version. This deviation is calculated in two directions: one perpendicular to the initial track direction and to z, ϕ direction, and the other one perpendicular to this first one and the initial track direction. Also the change in momentum projected in the above mentioned two directions is checked, as well as the change in kinetic energy (that should be zero, except minimal non-conservation by the field tracking).

3.5.3 Run configuration

The oval prod command runs the test and compare the current values with those in a reference file. The reference file is testFieldProp.ref.

3.6 Muon System

3.6.1 Description

The SimG4MuonValidation test checks the muon physics in GEANT4, by using 100 GeV muons. It writes out ROOT histograms of the relevant quantities and compare these histograms with those from a reference file using the OVAL tool.

3.6.2 Validation quantities

The test compares all the quantities that are relevant on the high energy muon physics in GEANT4.

The ntuple /afs/cern.ch/cms/geant4rep/genntpl/single_muon.ntpl is used to run 1000 muons of 100 GeV. The secondary particles are killed before being tracked.

For each muon it fills the following histograms:

- Energy lost
- Energy deposited
- Deviation in position
- Deviation in angle (deg)
- Number of tracking steps

And for each type of muon process, i.e. ionization, bremsstrahlung, e^+e^- production, muon nuclear interaction, decay and capture at rest, it makes histograms of the quantities:

- Total energy taken by secondaries
- Energy of secondary particles
- Angle of secondary track with respect to primary track (muon)

3.6.3 Run configuration

The oval prod command will run the job producing the histograms and will compare them with those in a reference file, testmuon_ref.root.

3.6.4 Validation tests

The histograms are compared with the reference ones at the end of job using the LCG PI Statistics Testing toolkit. This tool makes a χ^2 comparison and prints the probability to obtain that chi2 value or higher. This probability value is compared with the value of a reference file, `testMuonPhysics.ref` using the OVAL tool.

The following plots show an example of the histograms and the difference between the reference sample and another one simulated with different random numbers. One plot uses a linear scale while the other uses a log scale.

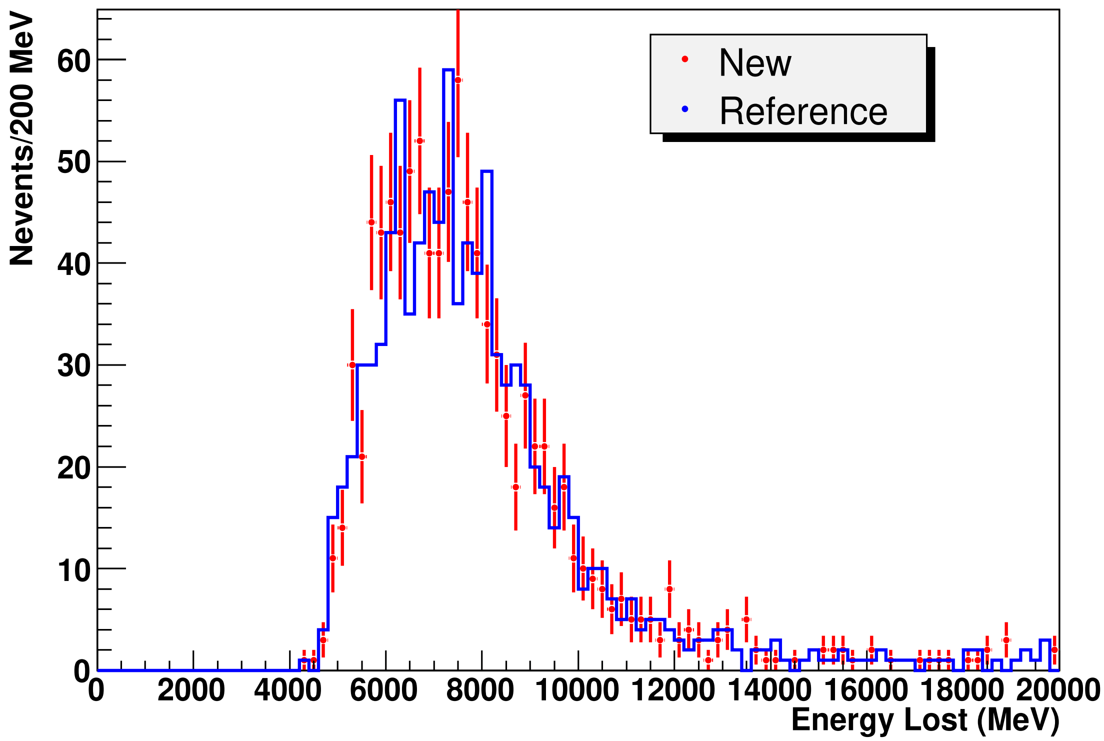

Figure 17: Energy lost for 100 GeV muons traversing all the CMS detector

3.7 Global Validation

3.7.1 Description

The `SimG4ValidationGlobal` package is currently under construction within the CMSSW framework and is intended to test simulation of the entire detector (all sub-detectors in parallel) for expected types of physics events with an active magnetic field. It is designed to allow reproduction of the plots from the ORCA package `ExSimHitsStatistics` within the new framework.

The package makes use of the `GlobalValidation` class which acts as an event producer within the framework adding the relevant simulated information from each sub-detector to the `PGlobalSimHit` object. This object is then exported as a branch to the POOL output file, `simevent.root`.

ROOT macro scripts will then be used to access the root file and create the relevant histograms. Validation comparisons with respect to reference plots will then be performed in a manner similar to the existing validation packages.

3.7.2 ROOT Tree Content

In addition to the standard CMSSW output, the `PGlobalSimHit` object is stored in a unique branch of the `simevent.root` output file. This object contains:

1. Monte Carlo

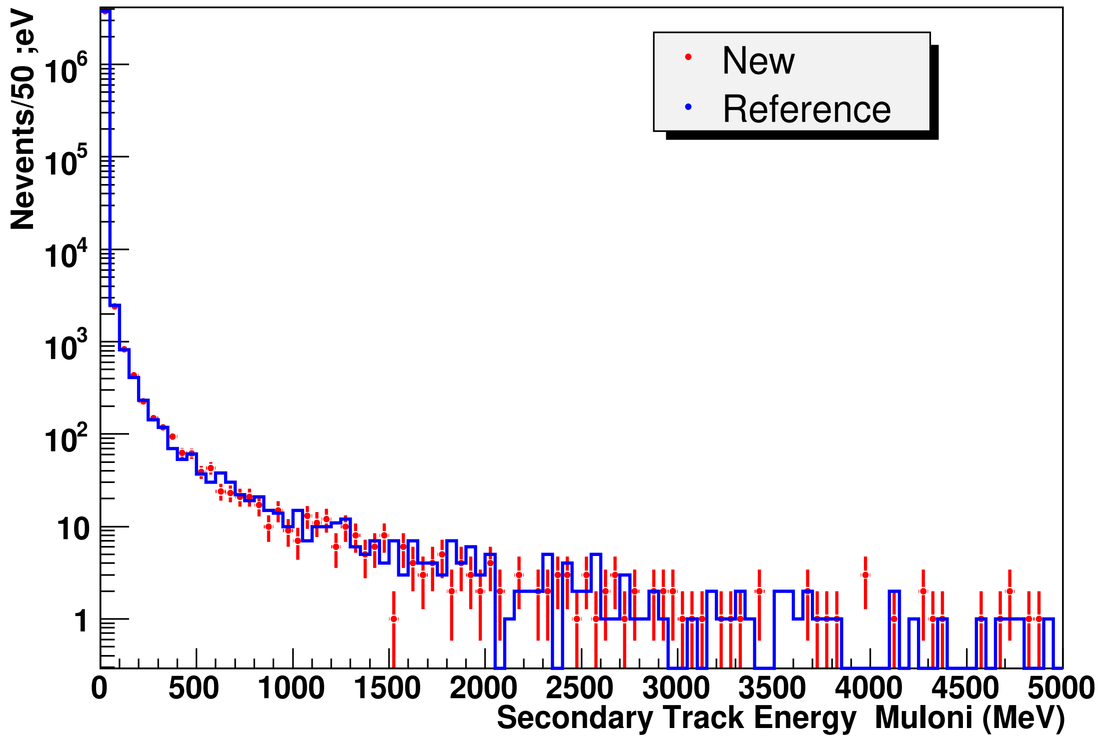

Figure 18: Energy lost for 100 GeV muons traversing all the CMS detector

Number of vertices from Geant4
Number of tracks from Geant4
Number of raw generated particles
Position (x,y,z) of each Geant4 vertex
p_T of each Geant4 track
Energy of each Geant4 track

2. Electromagnetic Calorimeter

   Number of Ecal hits
   Energy of each Ecal hit
   Time of flight for each Ecal hit
   Global ϕ of each Ecal hit
   Global η of each Ecal hit
   Number of preshower Hits
   Energy of each preshower hit
   Time of flight for each preshower hit
   Global ϕ of each preshower hit
   Global η of each preshower hit

3. Hadronic Calorimeter

   Number of hits
   Energy of each hit
   Time of flight for each hit
   Global ϕ of each hit
   Global η of each hit

4. Tracker

   Number of Pixel hits
   Global ϕ of each Pixel hit
   Global η of each Pixel hit
   Time of flight of forward Pixel hits
   Time of flight of barrel Pixel hits
   Global R of barrel Pixel hits
   Global Z of forward Pixel hits
   Number of Silicon hits
   Global ϕ of each Silicon hit
   Global η of each Silicon hit
   Time of flight of forward Silicon hits
   Time of flight of barrel Silicon hits
   Global R of barrel Silicon hits
   Global Z of forward Silicon hits

5. Muon

   Number of hits
   Global ϕ of hits
   Global η of hits
   Time of flight for DT hits
   Global R of DT hits
   Time of flight for CSC hits
   Global Z of CSC hits
   Time of flight for RPC barrel hits
   Global R of RPC barrel hits
   Time of flight for RPC forward hits
   Global Z of RPC barrel hits

3.7.3 Validation Quantities

Validation studies will be performed on a subset of these variables as best determined by future studies.

3.7.4 Run configuration

The creation of the initial root file is handled as a typical CMSSW job, using the `runP.cfg` configuration file to set the initial data source and other running conditions for the simulation. The file used for this package can be found in the `SimG4Validation/Global/test` subdirectory of the CMSSW package. The code is run using the `cmsRun -p runP.cfg` command.

ROOT macros will be written which create the full suite of plots as found in the old `ExSimHitsStatistics` package as well as the plots of interest for the current validation.

In the future, the validation will be performed using the OVAL tool as with the existing simulation packages.

3.7.5 Validation tests

The validation will be performed using statistical tools available in ROOT including the Kolmogorov-Smirnov test and the χ^2 test.

4 Integration and Optimization

The tool of choice for the integration of the SVS is OVAL, version 3_5_0. OVAL is a testing tool created to help developers to detect unexpected changes in the behavior of their software. Detailed information on the OVAL project may be found in Ref. [3].

OVAL includes the following features:

- Executes shell commands and scripts.
- Compiles and links validation programs.
- Runs the test programs and scripts within a specified Unix environment.
- Creates on the fly the needed configuration parameter sets or uses pre-defined external configuration files.
- Executes recursively a user-specified set of targets, including a hierarchy of subdirectories.
- Allows modular, configurable sub-tests within one subdirectory.

The SVS is executed from the `/SimG4Validation` base directory, using the following set of commands:

```
eval `scramv1 runtime -csh`
ovalv1 prod -nb [target1, target2..]
```

Here the `-nb` key indicates the no-build mode, since the suite libraries will be included in the simulation software distribution from CVS. It is up to the user to execute the build commands in case a modification is made to one or more package. If no specific target is given, every test in all packages included in the Suite will be executed.

Although OVAL 3_5_0 has suited the needs of the SVS project at the current stage, some improvements would allow more flexibility. For example:

- OVAL cannot extract and execute a particular test from a given target/subdirectory, if ran at a higher level in the hierarchy of subdirectories. For example, it is not possible to run single tests within a sub-detector package, such as ECalorimeter, Preshower, low-level mode, or high-level mode. OVAL will go down the directory hierarchy, identify all sources of executables and scripts, and run all tests. This feature imposes a severe limitation to the use of the modular structure of some packages, such as SimG4TrackerValidation and SimG4EcalValidation. In addition to reduced flexibility, there is a toll on the CPU performance.
- OVAL creates a run control file on the fly from a set of control data cards specified in the OvalFile. Optionally, separate environments may be defined to change the run conditions of the test. For example, SimG4TrackerValidation performs the same test sequentially for different pseudorapidity ranges. At the beginning of each run of the test, the data cards associated with the pseudorapidity range are modified, and the new values appended to the on-the-fly control file. Alternativelly, it is possible to use a run control file resident on disk by providing its name as an argument to the executable. A different file may be passed to the executable for each environment, that is each time the same test is run under different conditions. As an extension of the current features, it would be nice if OVAL allowed inclusions, that is the possibility of including disk resident run control files, as well as adding data cards or overriding their values. Since the format of run control files are different in CMSSW, with a nested structure which defines modules in between brackets, it is not clear how it will be possible to override data cards defined in a previous module.
- OVAL always requires a reference file in order to execute the "prod" command, which includes the "diff" step. Otherwise, the tool skips even the "run" step for the whole target.

We are aware that OVAL 5_9_8 has been released recently. The new release of the tool features important improvements over version 3_5_0, including several bug fixes that were of a concern to us. We plan to switch to OVAL 5_9_8 in the near future. We also plan to continue the dialog with the developers team for further upgrades.

5 Comparison Test Optimization Studies

A validation of a simulation against real physics data or with another simulation version relies on comparing two sets of data. There are a number of choices in this comparison, not including what type of data or distributions to compare as listed in the preceding sections. These choices are due to the statistical nature of the comparison. For example, the number of events to generate, the type of statistical test (*e.g.* χ^2 or Kolmogorov-Smirnov) to use, and whether to use binned distributions (histograms) or unbinned (event-by-event) data. In this section we present a summary of the results of our studies. More details can be found in a separate CMS Note [4].

Studies were done using the Kolmogorov-Smirnov (KS) and χ^2 tests. For the Kolmogorov-Smirnov test, we performed the studies both for unbinned and binned data. Several software methods were used for the tests, these are listed below.

1. TH1:: KolmogorovTest from ROOT, which is for 1-dimensional histograms (binned data).
2. TMath::KolmogorovTest from ROOT, which can be used for both binned and unbinned data.
3. StatisticsTesting::KolmogorovSmirnovComparisonAlgorithm from the GEANT 4 HEP Statistics Project, which is an external tool included in the LCG Physics Interface (PI) project [5]. This can be used for both binned and unbinned data.
4. Our own user code to implement the KS test for both binned and unbinned data, since the TMath implementation had some problems as explained in the separate CMS Note [4].

5.1 Optimization study procedure

The evaluation consisted of the following steps:

1. Create 500 pairs of data sample and reference data. Each sample and reference data set has 1000 data points taken randomly from the same parent distribution. The parent distribution shapes were chosen to represent some of the distributions that might typically exist in the subdetector test data. The three parent distribution shapes and variations used are listed below:
   (a) *Gaussian*: A Gaussian distribution with a mean(μ)= 0 and $\sigma = 1$, 0.9, 0.8 and 0.5. The reference always has $\sigma = 1$.
   (b) *Exponential*: An exponentially decaying function $e^{-x/\lambda}$ where the reference has $\lambda = 10$ and the variations have $\lambda = 9$, 8, and 5.
   (c) *Linear*: A linear distribution $mx + (1 - \frac{m}{2})$, from $x = 0$ to $x = 1$, where the reference has slope $m = 0$, and the variations have slopes of 0.1, 0.2 and 0.5.

   Either the unbinned data is used or, in case the binned data, a 1-dimensional ROOT histogram with 100 or 10 bins is used. The lower and upper limits for the histograms are $(-5, 5)$ for the Gaussian function; $(0, 100)$ for the exponential function; and $(0, 1)$ for the linear function.
2. Compare each of the 500 samples with the corresponding reference, by performing the KS test using the four methods given above, and recording and plotting the probability values (p-values) calculated with the different methods.
3. Repeat the test using either 100, 500, 10000, or 100000 events. Where we keep the number of events in the samples always equal to the number of events in the reference data that are compared to. The KS test p-values for these are again recorded and plotted.

When two data sets each with N events are compared using the KS test, the two cumulative distributions are compared and the maximum difference $D_N^{max}(x)$ is determined.[2)] One then determines, if the two data sets are drawn from the same parent distribution, with what probability one would get a value of $D_N^{max}(x)$ as large as, or larger than that observed. We call this the p-value for the KS test.

Figure 19 shows the $D_N^{max}(x)$, $\sqrt{N/2}D_N^{max}(x)$ and p-value distributions for comparing, using an unbinned KS test, 500 pairs of reference and sample data, for different numbers of events. Both the reference and sample data are taken randomly from the same parent Gaussian distribution with a width of one. It can be seen that the correct high statistics limit is being reached, *i.e.* for high statistics the $\sqrt{N/2}D_N^{max}(x)$ distribution is of the expected shape and thus the p-value distribution is approaching being flat.

5.2 Discriminating power of comparison tests

We define two quantities to express the power of a given test for discriminating between two distributions that are different, while correctly identifying that two distributions are actually the same (statistically). When two distributions are drawn from the same parent function, the fraction of the time that our test (wrongly) determines

[2)] The cumulative distribution is defined as follows: If a data set consists of N values (x_i, $i = 1$, N), and these are all ordered so that $x_i \leq x_j$ where $i < j$, the cumulative function is $S_N(x_i) = n(x_i)/N$ where $n(x_i)$ is the number of values less than or equal to x_i.

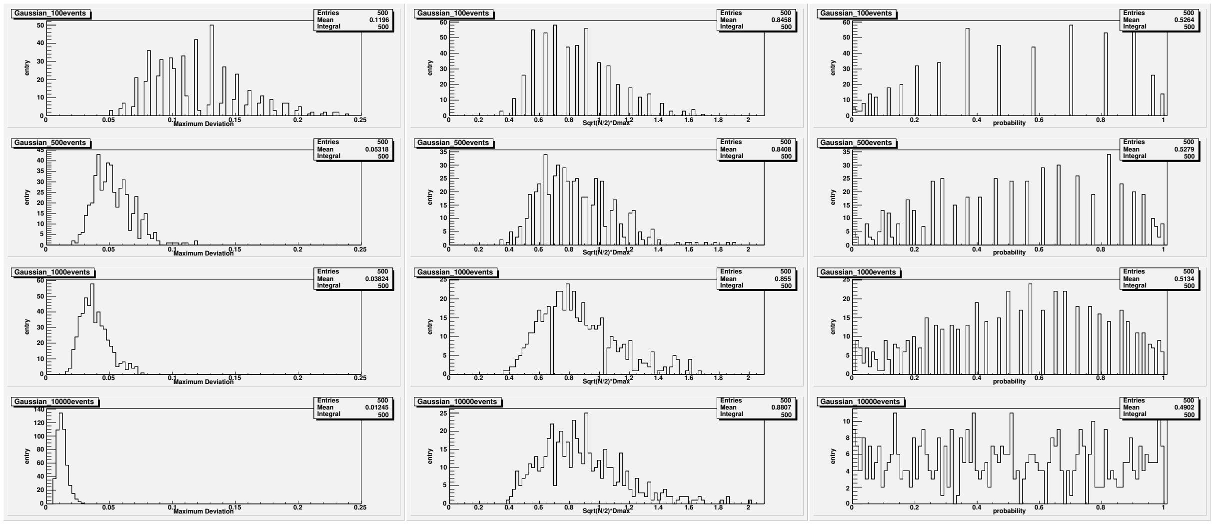

Figure 19: Results of the KS tests for unbinned data comparing reference data sets to samples where both reference and samples have 100 (1st row), 500 (2nd row), 1000 (3rd row) or 10000 (last row) events. All distributions are taken randomly from a Gaussian with a width of 1. (left) D_N^{max}; (middle) $\sqrt{N/2}D_N^{max}$; and (right) p-value.

they are different is called the *false-positive* fraction. The other quantity we define is called the *discrimination efficiency* and is defined further below.

In the high statistics limit where all p-value distributions are flat and independent of N, one can select an N-independent p-value to set the false-positive fraction. For example, if we accept that tests with a p-value less than 0.01 (1%) signifies that the two distributions are different, then the false-postive fraction is expected to be 0.01. Similarly, if we take tests with a p-value less than 0.1 to say that two distributions are different, the false-positive fraction would be 0.1. The larger the false-positive fraction the more histograms will be flagged as different when they could very well be the same (statistically). The false-positive fraction therefore determines how many histograms we have to look at manually. We want this to be as small as possible, though this is less important than having a high discrimination efficiency for a validation task.

One cannot appreciate the discriminating power by just looking at the p-value in figure 19. For example, one cannot see how much better a comparison using 1000 events is than one using only 100 events. To do that we should look at comparisons where the two distributions being compared are taken from different parent distributions.

Figure 20 shows the p-value distributions for comparing Gaussian reference data sets with $\sigma = 1$ with Gaussian sample distributions with varying values of σ. Results for comparisons using 1000 events per data set are shown as well as those from comparisons of data sets with 100 events. It can be seen that the mean p-value decreases more rapidly for the 1000 event sample comparisons, when the Gaussian width decreases. This means the discrimination is far more effective when using 1000 event samples compared to using 100 event samples. We need to express this quantitatively.

When comparing two distributions that are actually different, the *discrimination efficiency* is defined as the fraction of time our test will find them different. For example, compare a reference Gaussian with $\sigma = 1$ with a sample Gaussian distribution with a $\sigma = 0.8$ and each with 1000 events. If we set the *critical p-value* equal to 0.02 as the p-value below which a test will identify the two to be different, then from Fig. 20 we see that the discrimination efficiency is about 0.5. I.e. in only about half the time will we actually say the two distributions are different. The false-positive fraction for this p-value would be about 0.02 (if we were in the limiting high statistics case.) Although this is a small false-positive fraction, the discrimination efficiency is low. If we set the critical p-value to 0.2, the discrimination efficiency would be about 0.998, while the false-positive fraction would be about 0.2, this is much more preferable for our task at hand. It can be seen that if we scan the critical p-value from 0 to 1, we can obtain a curve for the discrimination efficiency versus false-postive fraction. Rather than rely on the limiting case to get the false-positive fraction, we integrate the p-value distributions (like those in Fig. 19) to obtain more accurate results.

The discriminating power of a test is best shown by the discrimination efficiency versus false-positive fraction curve. We obtained these curves for a number of different conditions.

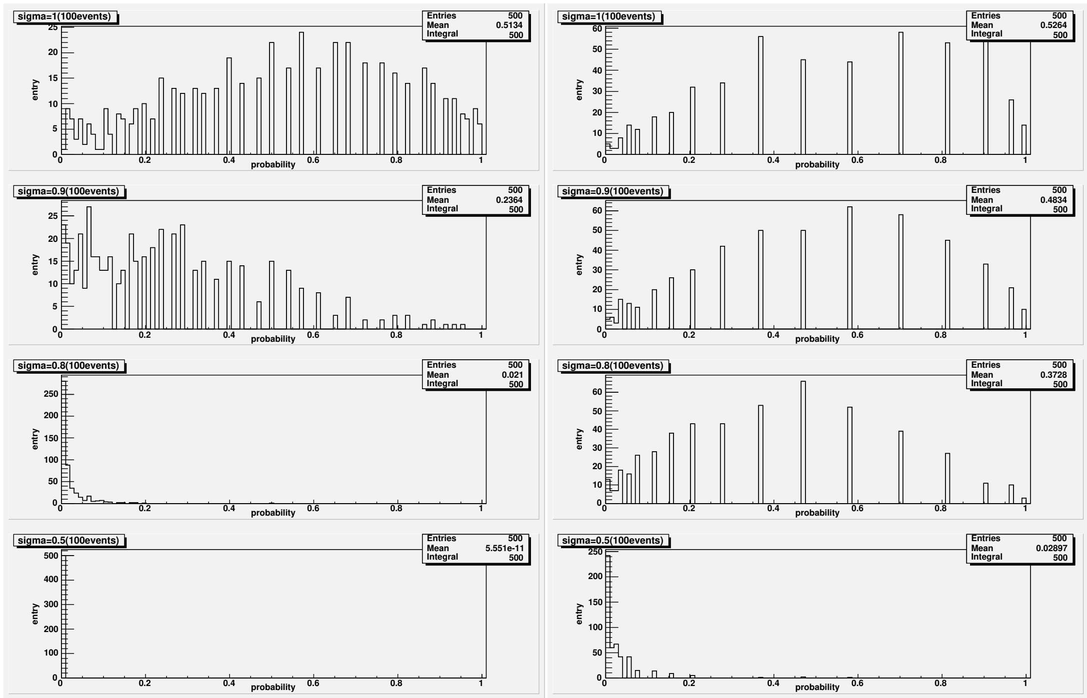

Figure 20: Results for p-values of KS tests for unbinned data comparing Gaussian ($\sigma = 1$) reference data sets with samples with different Gaussian widths: (top) $\sigma = 1$, (2nd row) $\sigma = 0.9$, (3rd row) $\sigma = 0.8$, and (bottom) $\sigma = 0.5$. (left) Each of the 500 sample and reference data sets have 1000 events each; (right) Each of the 500 sample and reference data sets have 100 events each.

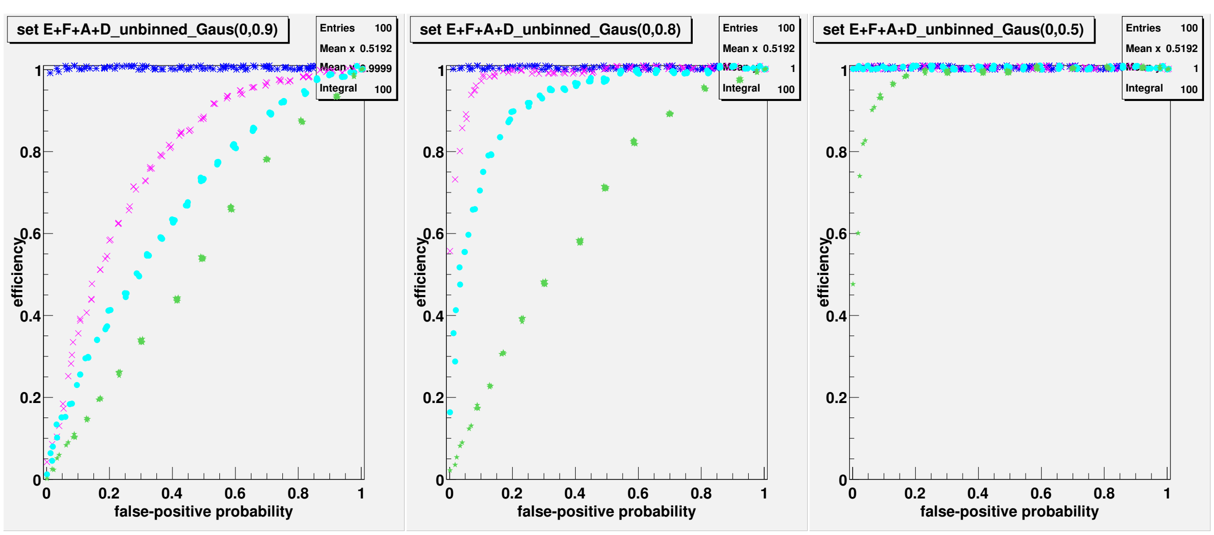

Figure 21: Efficiency in discriminating Gaussian distributions with different widths plotted against the false-positive fraction as explained in the text. The results were obtained from KS tests for unbinned data comparing reference data sets with N events and $\sigma = 1$ to samples with N events and different Gaussian widths: (left) $\sigma = 0.9$, (middle) $\sigma = 0.8$, (right) $\sigma = 0.5$. (green pentagrams) $N = 100$, (cyan dots) $N = 500$, (magenta crosses) $N = 1000$, and (blue double-crosses) $N = 10000$.

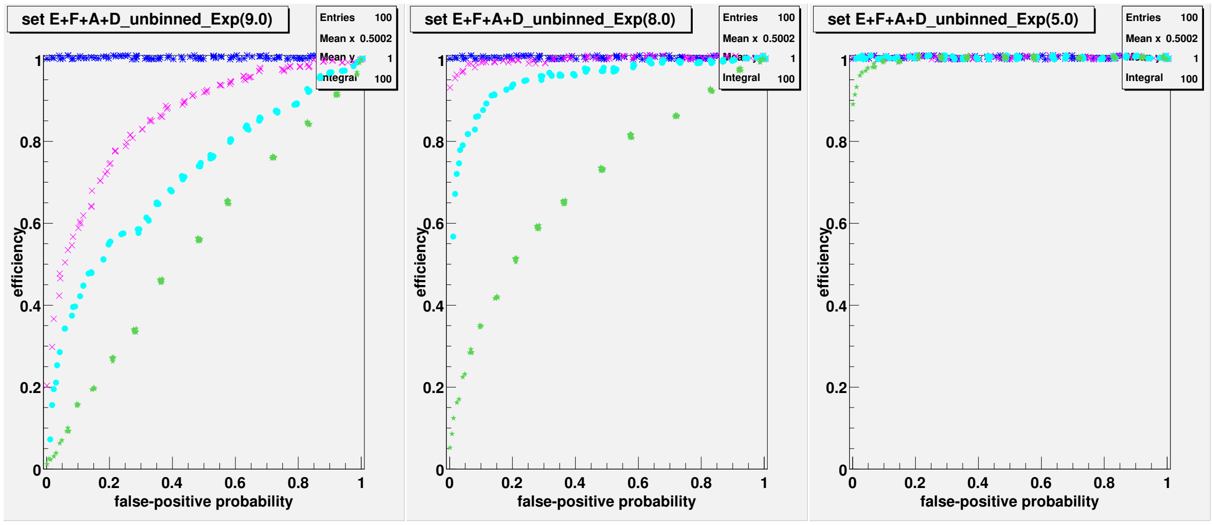

Figure 22: Efficiency in discriminating Exponential distributions with different decay constants plotted against the false-positive fraction as explained in the text. The results were obtained from KS tests for unbinned data comparing reference data sets with N events and $\lambda = 10$ to samples with N events and different decay constants: (left) $\lambda = 0.9$, (middle) $\lambda = 0.8$, (right) $\lambda = 0.5$. (green pentagrams) $N = 100$, (cyan dots) $N = 500$, (magenta crosses) $N = 1000$, and (blue double-crosses) $N = 10000$.

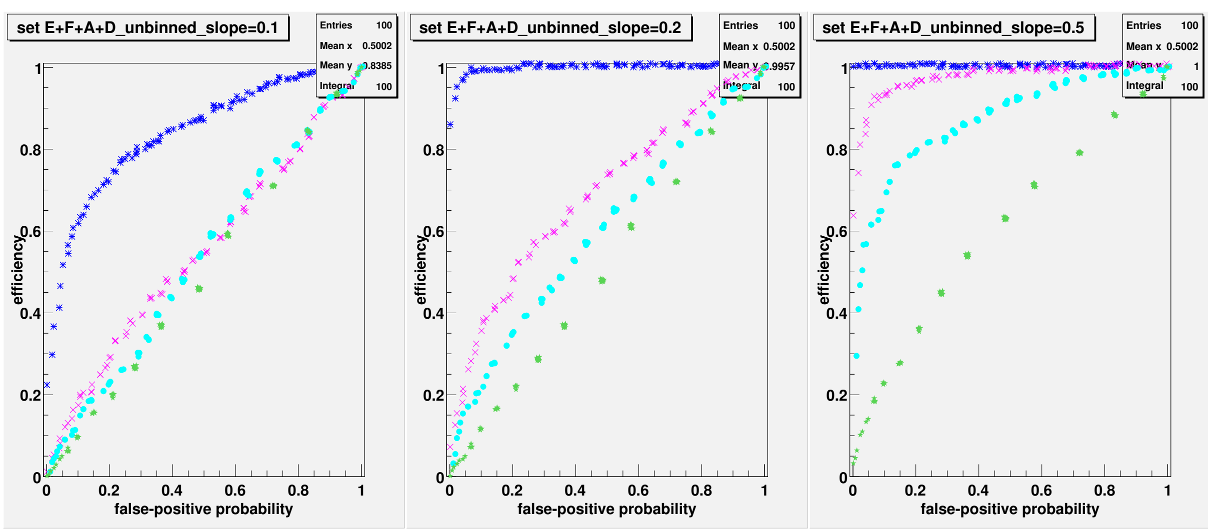

Figure 23: Efficiency in discriminating linear distributions with different slopes plotted against the false-positive fraction as explained in the text. The results were obtained from KS tests for unbinned data comparing reference data sets with N events and slope$= 0$ to samples with N events and different decay constants: (left) slope$= 0.1$, (middle) slope$= 0.2$, (right) slope$= 0.5$. (green pentagrams) $N = 100$, (cyan dots) $N = 500$, (magenta crosses) $N = 1000$, and (blue double-crosses) $N = 10000$.

Figures 21, 22, and 23, show the results for comparing Gaussian, exponential, and linear distributions, respectively. Each discrimination efficiency versus false-positive plot contains the results for comparisons using 100, 500, 1000 or 10000 event samples.

As expected, there is a compromise between discrimination efficiency and false-postive fraction. Higher discrimination efficiencies leads to higher false-positive fractions. Also with 10000 events, a near 100% discrimination efficiency is possible at small false-positive fractions, even for smaller (10%) differences between the reference and sample distributions. Conversely, acceptable discrimination efficiencies using 100 event samples are only possible if one accepts false-positive fractions of almost 100%, unless the reference and sample distributions are very different. The actual choice of the number of events to use is dictated by the size of the difference one wants to detect. For detecting ∼20% differences in Gaussian or exponential distributions, 1000 event samples give reasonable discrimination efficiency without too high a false-positive fraction. Linear distributions, (probably distributions that change more slowly) seem to require larger event samples to obtain the same discrimination power.

5.3 Using unbinned and binned data for KS test comparisons

The intent of the validation package is to use histogrammed variables for comparisons rather than unbinned data, as histograms take up much less storage and are easier to handle and visualize. In this section we compared the results of using the KS test for binned and unbinned data, to see the effect on the discrimination power when the data are binned.

The Kolmogorov-Smirnov test is intended for unbinned data, however, if one understands the issues, there is no real problem with using it for binned data. The basic issue is that the deviation D_N^{max} will not be distributed as expected so that the probability distribution will not be flat, even in the high statistics limit. This means that the calculated p-value using the standard (unbinned) algorithm is not "correct". We can still use the KS test for binned data, we just cannot rely on the p-value returned by the test to follow the expected distribution. Since we can obtain the p-value distribution ourselves in a Monte Carlo study, we do not have to make this assumption.

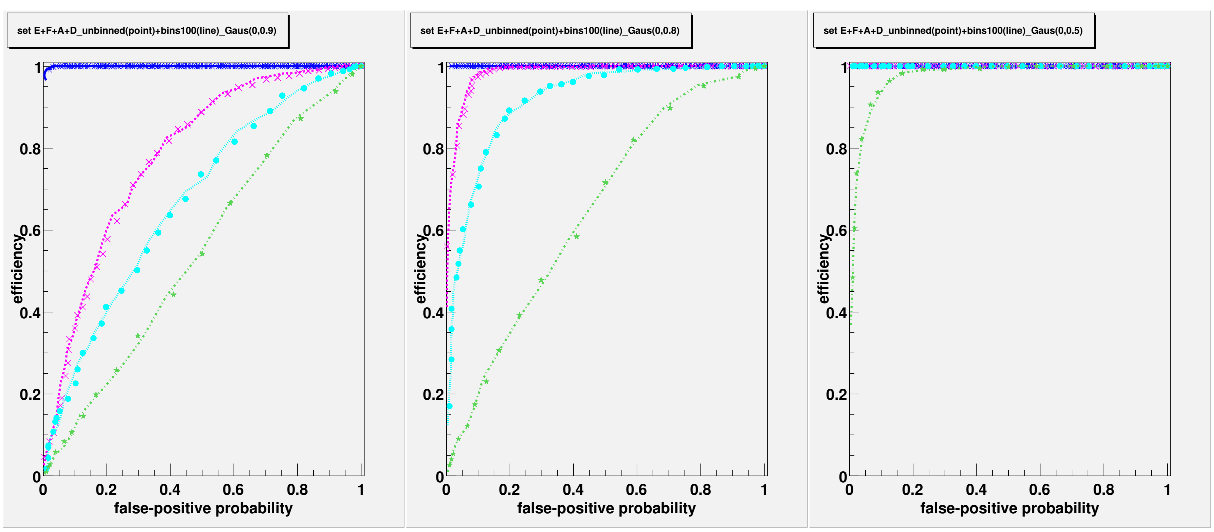

Figure 24: Efficiency in discriminating Gaussian distributions with different widths plotted against the false-positive fraction as explained in the text. The results were obtained from KS tests for unbinned data (points) and binned data using 100 bins (line). Comparisons are for pairs of reference data with N events and $\sigma = 1$ with samples containing N events and different Gaussian widths: (left) $\sigma = 0.9$, (middle) $\sigma = 0.8$, (right) $\sigma = 0.5$. (green pentagrams, dot-dash line) $N = 100$, (cyan dots, dotted line) $N = 500$, (magenta crosses, dashed line) $N = 1000$, and (blue double-crosses, solid line) $N = 10000$.

Figure 24 shows the results for comparing the different distributions using the KS test with unbinned data with the results using binned data. Each discrimination efficiency versus false-positive plot contains the results for comparisons using 100, 500, 1000 or 10000 event samples.

One would have expected that the KS test using unbinned data to be a more powerful discrimination tool than when using binned data. This would be expected to be more noticeable as the number of bins used is decreased. These results and those for comparing exponential and linear distributions show surprisingly that the KS test when using binned data can be equally as powerful as when using unbinned data. However the same performance in discrimination efficiency at a given false-positive fraction is achieved at a higher p-value when using binned data. This is shown in Figs. 25.

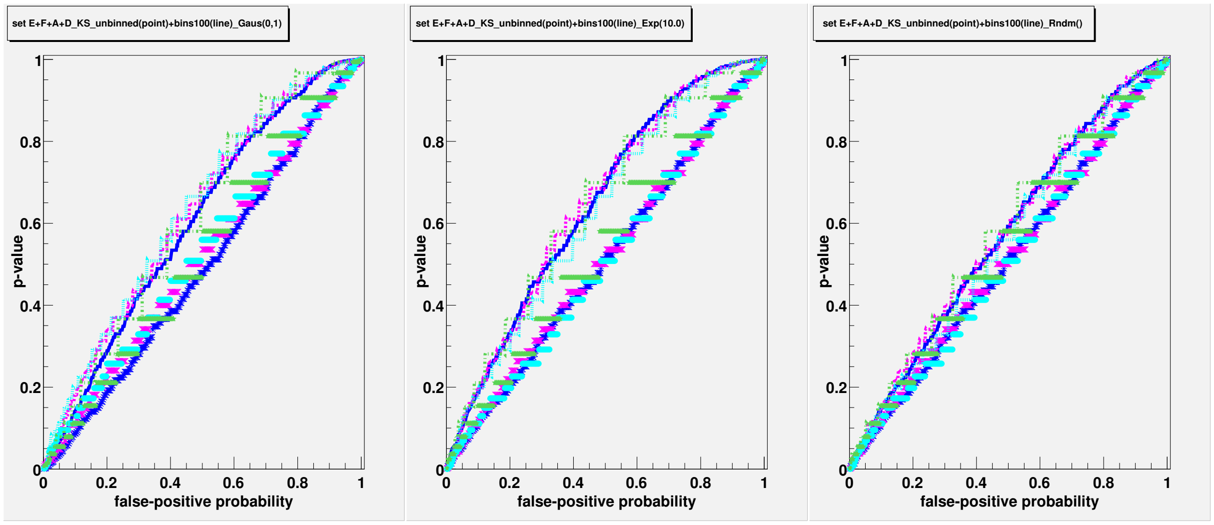

Figure 25: The p-value needed to obtain a given false-positive fraction, comparing results from KS tests for unbinned data (points) and binned data using 100 bins (line). Comparisons are for pairs of reference data with N events with samples containing N events, and for the different distributions (left) Gaussian, (middle) exponential, (right) linear. (green pentagrams, dot-dash line) $N = 100$, (cyan dots, dotted line) $N = 500$, (magenta crosses, dashed line) $N = 1000$, and (blue double-crosses, solid line) $N = 10000$.

5.4 Comparison of KS tests with χ^2-tests

The χ^2 test is a commonly used statistic for comparing if two distributions are from the same parent function (*i.e.* if they *match*).

We used a χ^2 test statistic coded by us and is defined as follows. For two histograms, one with entries n_i and the other m_i, where $i = 1$ to 100 (number of bins), and $\sum n_i = \sum m_i = N$ (same total number of entries), the χ^2 is given by:

$$\chi^2 = \sum_{i=1}^{N} \frac{(n_i - m_i)^2}{n_i + m_i}, \quad \text{for } i \text{ where } (n_i + m_i) \neq 0 \tag{1}$$

The number of degrees of freedom *ndf*$= N_{nzb} - 1$. Where N_{nzb} is the total number of bins where n_i and m_i are not both zero. If all bins are populated then $N_{nzb} = N$. The *ndf* is one less than N_{nzb} because we effectively constrain the total number of entries in the two histograms to be the same. This counts as one constraint. *I.e.* one does not test the agreement of the total number of events in the two histograms. If all n_i and m_i are large enough (*e.g.* larger than 5), it can be shown that statistic of Eq.(1) follows a χ^2 distribution with $N-1$ degrees of freedom. Thus for large enough N the p-value distribution will be flat.

Figure 26 shows the distributions of χ^2, χ^2/ndf, and p-value using the the statistic of Eq.(1) for different total number of events. The two histograms were drawn from the same parent Gaussian function with a mean of zero and $\sigma = 1$. It can be seen that the p-value distribution approaches being flat for large N.

The χ^2 statistic given by the ROOT TH1 class, TH1::Chi2Test, in version 4.04.02f of ROOT it is defined slightly differently, but is the same when the two data samples have the same number of entries. However the number of degrees of freedom by default is *ndf*$= N_{nzb}$. Version 4.04.02f has an incorrect calculation of the p-value as the wrong χ^2 and *ndf* is used in the call to TMath::Prob to calculate the p-value.[3] This leads to a statistic that does not

[3] The parameters passed, from inside TH1::Chi2Test, to TMath::Prob is $\chi^2/2$ and *ndf/2*, whereas it should just be χ^2 and *ndf*. This may be because of an earlier implementation of TMath::Prob. Since the incomplete Gamma function is used, the parameters passed to that function should be half the χ^2 and *ndf* values. However inside TMath::Prob itself, the passed

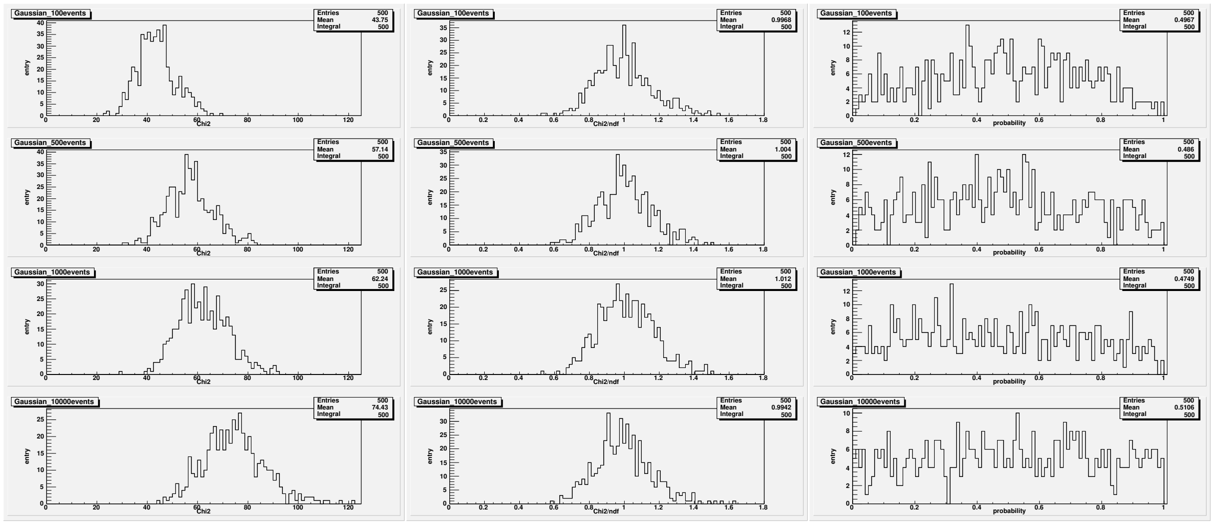

Figure 26: Results of χ^2 tests for binned data using 100 bins comparing reference data to samples where both reference and sample data sets have 100 (1st row), 500 (2nd row), 1000 (3rd row) or 10000 (last row) events. All data are taken randomly from a Gaussian with a width of 1. The user-defined definition of χ^2 is used as explained in the text. (left) χ^2; (middle) χ^2/ndf; and (right) p-value.

follow an actual χ^2 distribution and the probability is not flat. Nevertheless we also tested using this this statistic "as-is", since this is used in the validation packages, and we get similar results to using our user-coded statistic.

Figures 27– 29 show the results for comparing Gaussian, exponential, and linear distributions using the KS test with binned data with the results of a χ^2 test with 100 bins. Each discrimination efficiency versus false-positive plot contains the results for comparisons using 100, 500, 1000 or 10000 event samples. Results using both the user-defined χ^2 statistic are shown.

These results seem to show that the χ^2 test performs more poorly compared to the KS test.

5.5 Conclusions from Comparison Test Studies

The first preliminary conclusion is that the KS test using binned data can be just as powerful in a discrimination test as using unbinned data, but the p-value must be set lower when using binned data.

The second preliminary conclusion is that the KS test using binned data is preferred over a χ^2 test as it is a more power discrimination tool.

These conclusions are preliminary because we are still trying to understand two unexpected results:

1. The discrimination power when using the KS test is almost the same using binned data as with unbinned, and moreover, seems to be unaffected when the number of bins is reduced from 100 to 10.
2. The discrimination power in the χ^2 test is better with 10 bins than 100 bins, and for the Gaussian distributions, the 10 bins discrimination is even better than the KS test discrimination.

Results for comparisons to a (fixed) function is being done to further check and better understand these results. A more detailed and full report is under preparation [4].

6 Exernal Tools

OVAL [3] is the main tool used in SVS to control its execution and perform comparisons. This tool enables one to compare values in an ASCII file with those in a reference file. The user defines a set of tolerances for the quantities of interest, and OVAL returns a "DIFF" message in an output file when the difference is outside the tolerance range. The evaluation of the usage of OVAL in the SVS project is included in Section 4. The SVS team

parameters are already divided by two.

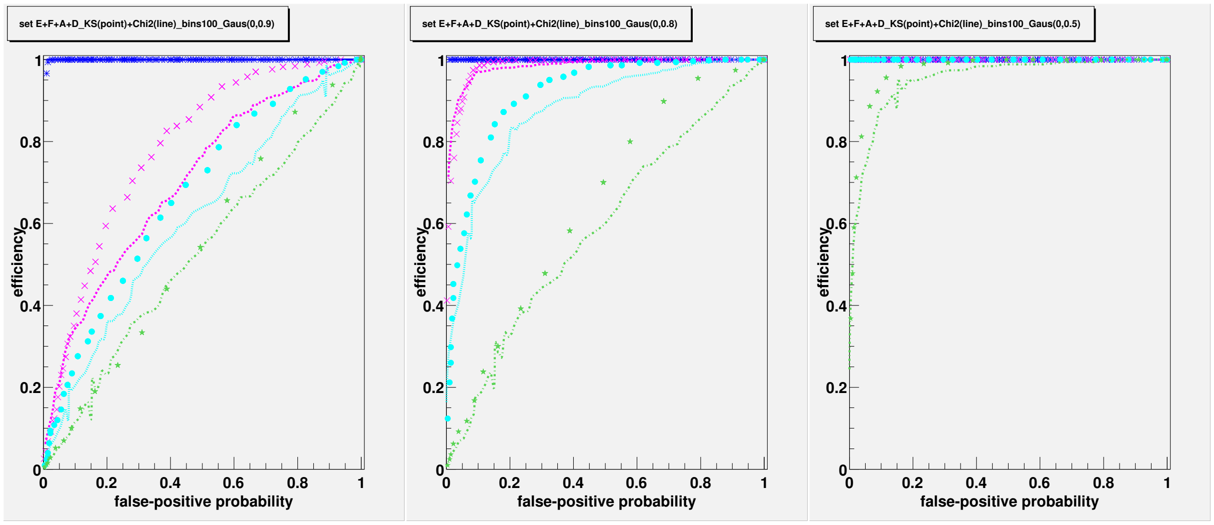

Figure 27: Efficiency in discriminating Gaussian distributions with different widths plotted against the false-positive fraction as explained in the text. The results were obtained from KS tests for binned data (points) and from the user-defined χ^2-tests for binned data (line) using 100 bins. Comparisons are for pairs of reference data containing N events having $\sigma = 1$ with samples containing N events and different Gaussian widths: (left) $\sigma = 0.9$, (middle) $\sigma = 0.8$, (right) $\sigma = 0.5$. (green pentagrams, dot-dash line) $N = 100$, (cyan dots, dotted line) $N = 500$, (magenta crosses, dashed line) $N = 1000$, and (blue double-crosses, solid line) $N = 10000$.

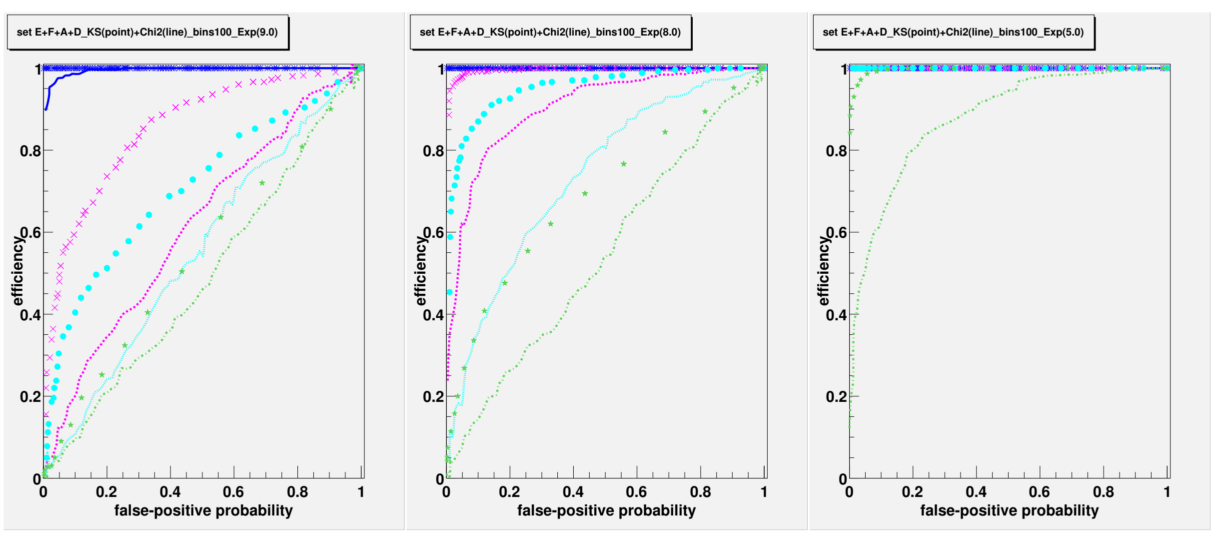

Figure 28: Efficiency in discriminating Exponential distributions with different decay constants plotted against the false-positive fraction as explained in the text. The results were obtained from KS tests for binned data (points) and from the user-defined χ^2-tests for binned data (line) using 100 bins. Comparisons are for pairs of reference data containing N events having $\lambda = 10$ with samples containing N events and different decay constants: (left) $\lambda = 0.9$, (middle) $\lambda = 0.8$, (right) $\lambda = 0.5$. (green pentagrams, dot-dash line) $N = 100$, (cyan dots, dotted line) $N = 500$, (magenta crosses, dashed line) $N = 1000$, and (blue double-crosses, solid line) $N = 10000$.

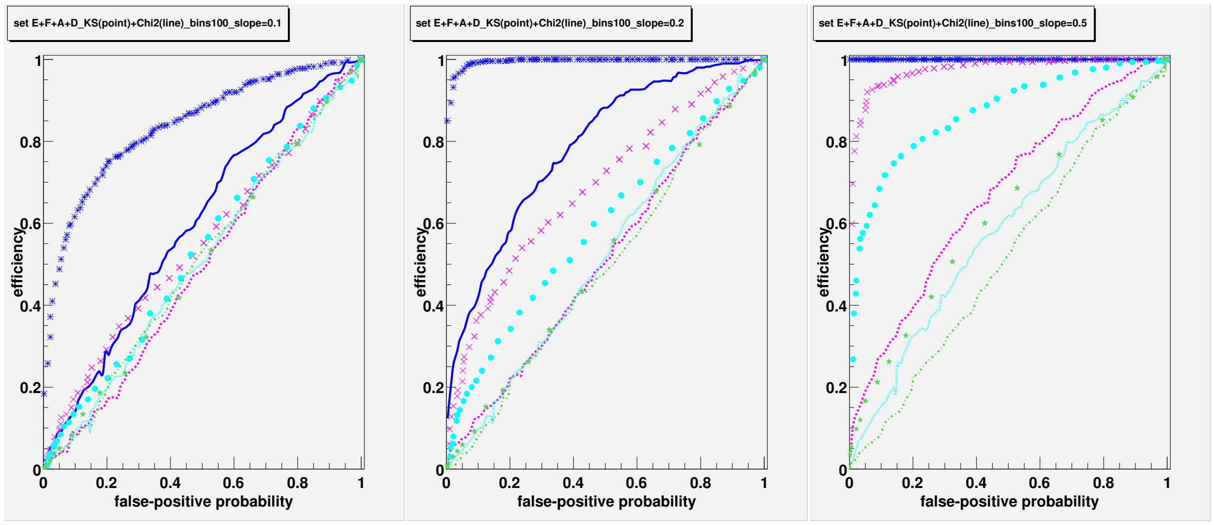

Figure 29: Efficiency in discriminating linear distributions with different slopes plotted against the false-positive fraction as explained in the text. The results were obtained from KS tests for binned data (points) and from the user-defined χ^2-tests for binned data (line) using 100 bins. Comparisons are for pairs of reference data containing N events having slope$= 0$ with samples containing N events and different slopes: (left) slope$= 0.1$, (middle) slope$= 0.2$, (right) slope$= 0.5$. (green pentagrams, dot-dash line) $N = 100$, (cyan dots, dotted line) $N = 500$, (magenta crosses, dashed line) $N = 1000$, and (blue double-crosses, solid line) $N = 10000$.

is in contact with the OVAL developers, who are open to suggestions. Requests are coming as decisions are made during the migration process to CMSSW.

Tools other than OVAL are being tested for histogram comparision. The two obvious options are the Statistical Tool Kit [5], and the tests available within ROOT [2].

The Statistical Tool Kit is a general purpose tool for statistical analysis, which follows the tool kit philosophy, and is designed to be flexible and extensible. It is component-based, allowing re-use and integration in different frameworks, and has no dependence on any specific analysis tool. The user layer bridges the core statistical component and the user's analysis, which may be based on different frameworks such as AIDA or ROOT. The user layer shields the user from the underlying algorithms and design, and only deal with the user's analysis objects and the choice of comparison algorithm. The kit currently supports a very complete statistics software suite for comparison of two binned and unbinned distributions, including Anderson-Darling, Chi-squared, Fisz-Cramer-von Mises, Tiku tests (binned), Creamer-von Mises, Goodman, Kolmogorov-Smirnov, Kuiper, and Tiku tests (unbinned). The Statistical Tool Kit is currently being used by the Geant4 physics validation group within the LCG Simulation Validation project. The current version of SVS uses the Statistical Tool Kit in the Muon and Ecal packages.

ROOT is an analysis framework widely used in running experiments. ROOT is not a software project that specializes in statistical tools, but the Chi-square and Kolmogorov-Smirnov tests are made available through methods of its histogram class.

While the Statistical Tool Kit is a project specializing on statistical tools and provides a large variety of goodness-of-fit tests, ROOT satisfies the current needs of the SVS, and allows one to perform the tests within the same macro file used to analyze the ROOT trees produced by the validation packages. The most complete solution in the long term could be to use the Statistical Tool Kit as an external library loaded into ROOT. This option would allow the Simulation Validation Group, and CMS in general, to benefit from the large variety of tests in the tool kit, while keeping the analysis within the ROOT framework.

The SVS team performed studies to verify the consistency between the Chi-squared and Kolmogorov-Smirnov test results obtained from the Statistical Tool Kit and from ROOT.

6.1 Comparing Statistics Toolkit with ROOT Statistics Methods

As an initial step towards evaluating software tools for statistical comparison of binned and unbinned samples, we have tested the Chi-squared methods available in ROOT and in PI/StatisticsTesting package. ROOT v4.04.02

and PI v1.3.3 were used in this test. It needs to be pointed out that in this processes we used modified class StatisticsComparator, with the interface to ROOT histograms replacing the default interface to AIDA histograms. The modified source code was provided to us by the StatisticsTesting developers, via private communication.

The test is based on a series of 5,000 trials. Each trial contains 4 steps :

- creating two 1-dimentional ROOT histograms (TH1F), with 100 bins in the range from 0.5 to 1.5 (the actual number of bins is 102, with 0-bin to hold underflow, and bin 101 to hold overflow)
- filling each histogram with 10,000 Gaussian distributed random seeds, of the same MEAN=0.5 and different σ's, σ_1=0.3 and σ_2=0.31
- performing χ^2 test using ROOT TH1::Chi2Test() method, for the two histograms; underflow and overflow of the distributions were included.
- performing χ^2 test using the PI/StatisticsTesting Chi2ComparisonAlgorithm class; underflow and overflow of the distributions were included

In each trial we looked at the two values of the probability and pvalue, calculated with the use of these two different software packages. The difference between the two pvalues, pvalue(ROOT)-pvalue(StatisticsTesting), is presented in Fig. 30.

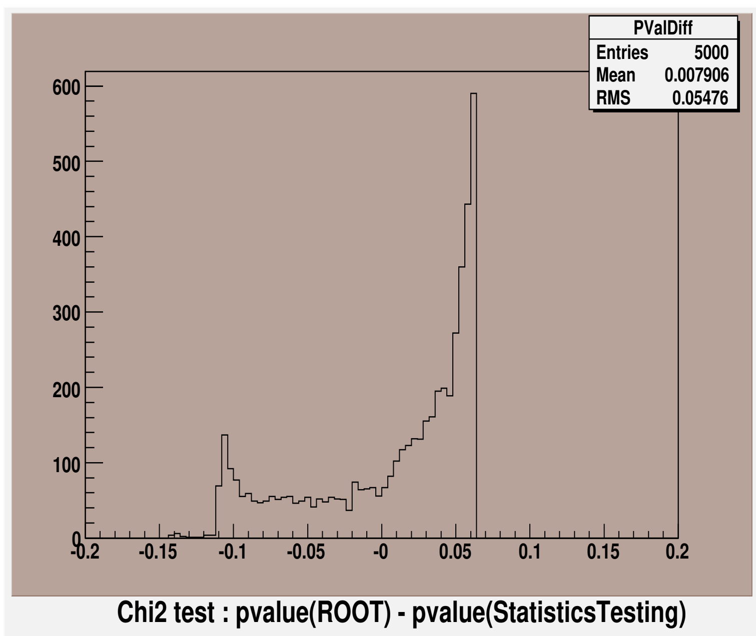

Figure 30: Comparison of pvalues obtained using ROOT and using the Statistics Toolkit.

We have noticed that there were no difference in computing the χ^2 of the two binned samples. However, there is a difference in computing the NDF. Since the number of entries both histograms was always the same, the NDF is typically equal the number of bins minus 1. However, the ROOT TH1::Chi2Test() methods checks if a specific i-th (j-th, k-th, etc.) bin is empty in both histograms and, in this case, recursively reduces NDF by 1. In the Chi2ComparisonAlgorithm of the PI/StatisticsTesting the NDF seems to be always equal the number of bins minus 1. Further differences are likely to lay in the use of different software (sub)packages to perform the necessary mathematics involved in the calculation of the probability, in particular, the use of different implementations of the Gamma-function.

As a cross-check, in each trial we have taken the χ^2 and the NDF computed by the Chi2ComparisonAlgorithm (provided cases where NDF was equal to the one computed by ROOT TH1::Chi2Test()) and recalculated probability using ROOT TMath::Prob() method. The result was always practically identical to the one computed by the ROOT TH1::Chi2Test() method, as shown in Fig. 31.

In the case of comparing two identical histograms ($\chi^2 = 0$) both packages always calculate the probability to be 1.

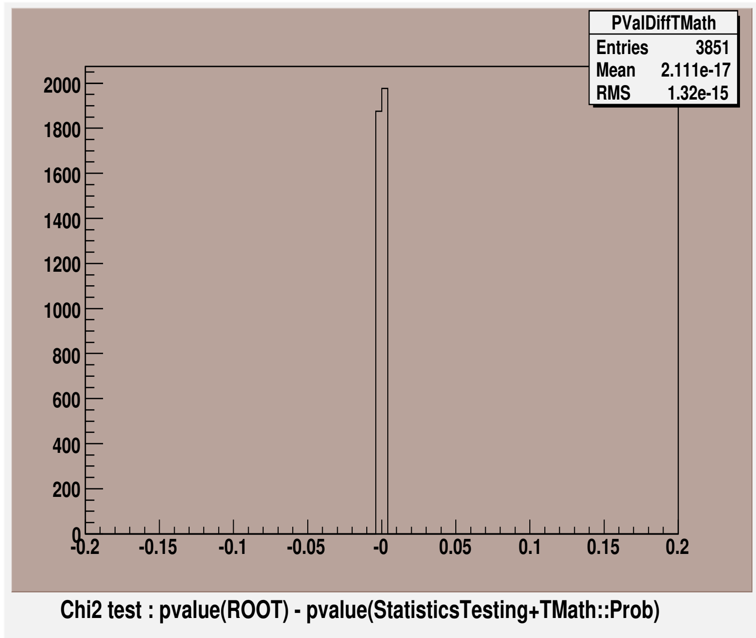

Figure 31: Comparison of pvalues obtained using ROOT and using the Statistics Toolkit when NDF is the same and using the same function in both cases to calculate the pvalue from the χ^2 and NDF value.

We have also performed timing tests for these two implementations of the Chi-squared test, based on the 5,000 trials. In the timing test each trial included

- at the beginning of each trial, creating the two ROOT TH1F histograms
- filling each of these two ROOT histograms with 10,000 random Gaus-distributed seeds.
- performing the "Chi2" test by either the ROOT TH1::Chi2Test method or the Chi2ComparisonAlgoritm.
- printing the resulting χ^2, NDF and pvalue.
- deleting the histogram at the end of each trial.

ROOT TStopwatch timer was used to estimate the required CPU. As a result, we found that the all-ROOT based test required 53.93 sec to perform 5,000 trials. It is further estimated that most of the CPU went into creating, filling and deleting the histograms and into printing the resulting information, while the amount of CPU required to perform the "Chi2" test itself seemed to be negligible. The test which employed the Chi2ComparisonAlgorithm of the PI/StatisticsTesting required 5235.41 sec to perform 5,000 trials. It has to be stressed that even in the case of comparing two identical histograms ($\chi^2 = 0$),the PI/StatisticsTesting still required ~ 0.7 sec to process one trial. The test was done on the cmsuaf.fnal.gov cluster (Intel CPU 2.4GHz).

We would like to point out that it is not the goal or our study to investigate whether this striking difference in the amount of required CPU was due to details of the coding technique (including heavy use of STL containers, often nested, in the StatisticsTesting) or in the details of mathematical algorithms involved in the computations. For the SVS task this timing difference is not so significant. It could be more significant though for similar tasks done in the online, such as might be done in the online Data Quality Monitoring (DQM).

7 Maintenance and Operation

Each package developer/expert will maintain his/her own package. Either the production group or the operator on duty will run the integrated Suite before each release and consult with the corresponding experts in case there are differences with respect to reference values. Experts will take corrective measures if necessary.

8 Extensions

The current Suite validates simulation quantities from Geant4 or hit information. A natural extension is to incorporate digis to the validation process. This is the next stage of the project. Another extension would check the fast simulation with respect to full simulation reference data.

A longer term extension would be to include physics quantities to compare. This will enable not only a comparison of one simulation version with another, but also a comparison of the simulation with real data. Although a physics validation of the simulation is complicated and likely to be done separately, once the correct distributions are determined, they could be included in the SVS package so that when a new version of either the simulation code or the reconstruction code is created, the suite could be used to automatically compare with previous versions for both simulation and real data. These could be done as part of a nightly build of the code. More generally, some of the distributions used in the SVS package would be the same ones used also in the DQM, for example distributions that monitor the alignment of the tracking detectors.

9 Acknowledgments

The authors wish to thank the various developers of the packages used in the SVS, for their help during the development and testing of the SVS package. We also wish to thank members of the PRS groups for helping to define the various distributions to use in the SVS package.

References

[1] http://geant4.web.cern.ch/geant4/".

[2] http://root.cern.ch/".

[3] http://polywww.in2p3.fr/cms/software/oval", http://savannah.cern.ch/projects/oval".

[4] Harry W. K. Cheung and Xiaoping Ding, "Comparison Test Optimization Studies for Validation Tests", CMS Note in preparation.

[5] http://www.ge.infn.it/geant4/analysis/HEPstatistics"