Status of Fast Detector Simulation

Mikael Berggren[1]

[1]DESY, Hamburg

Contribution to the LCFORUM Meeting, Hamburg, June 2010

Outline

The need for fast simulation

- We have very good full simulation now.
- So why bother about full simulation ?
- Answer:
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

The need for fast simulation

- We have very good full simulation now.
- So why bother about full simulation ?
- Answer:
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

The need for fast simulation

- We have very good full simulation now.
- So why bother about full simulation ?
- Answer:
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

The need for fast simulation

- We have very good full simulation now.
- So why bother about full simulation ?
- Answer:
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

The need for fast simulation

- We have very good full simulation now.
- So why bother about full simulation ?
- Answer:
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

Cross-section and event-generation time

PYTHIA obtains a total cross-section for $e^+e^- \rightarrow \gamma\gamma e^+e^- \rightarrow q\bar{q}e^+e^-$ at $E_{CMS} = 500$ GeV of 28371 pb
(+ another 7170 pb if the diffractive and elastic components are included, but these classes do not contribute to high $P_{T\ miss}$-events)

- $\int \mathcal{L}dt = 500$ fb^{-1} $\rightarrow$ 14 $\star 10^9$ events are expected.
- 10 ms to generate one event.
- 10 ms to fastsim (SGV) one event.

10^8 s of CPU time is needed, ie more than 3 years. This goes to 3000 years with full simulation.

Clearly, there is need to reduce this number by one or two orders of magnitude, by using generator level cuts.

Cross-section and event-generation time

PYTHIA obtains a total cross-section for $e^+e^- \rightarrow \gamma\gamma e^+e^- \rightarrow q\bar{q}e^+e^-$ at $E_{CMS} = 500$ GeV of 28371 pb
(+ another 7170 pb if the diffractive and elastic components are included, but these classes do not contribute to high $P_{T\ miss}$-events)

- $\int \mathcal{L} dt$ = 500 fb^{-1} $\rightarrow$ 14 $\star 10^9$ events are expected.
- 10 ms to generate one event.
- 10 ms to fastsim (SGV) one event.

10^8 s of CPU time is needed, ie more than 3 years. This goes to 3000 years with full simulation.

Clearly, there is need to reduce this number by one or two orders of magnitude, by using generator level cuts.

Cross-section and event-generation time

PYTHIA obtains a total cross-section for $e^+e^- \rightarrow \gamma\gamma e^+e^- \rightarrow q\bar{q}e^+e^-$ at $E_{CMS} = 500$ GeV of 28371 pb
(+ another 7170 pb if the diffractive and elastic components are included, but these classes do not contribute to high $P_{T\ miss}$-events)

- $\int \mathcal{L}dt$ = 500 fb^{-1} $\rightarrow$ 14 $\star 10^9$ events are expected.
- 10 ms to generate one event.
- 10 ms to fastsim (SGV) one event.

10^8 s of CPU time is needed, ie more than 3 years. This goes to 3000 years with full simulation.

Clearly, there is need to reduce this number by one or two orders of magnitude, by using generator level cuts.

Cross-section and event-generation time

PYTHIA obtains a total cross-section for $e^+e^- \rightarrow \gamma\gamma e^+e^- \rightarrow q\bar{q}e^+e^-$ at $E_{CMS} = 500$ GeV of 28371 pb
(+ another 7170 pb if the diffractive and elastic components are included, but these classes do not contribute to high $P_{T\ miss}$-events)

- $\int \mathcal{L}dt = 500\ \mathrm{fb}^{-1} \rightarrow 14 \star 10^9$ events are expected.
- 10 ms to generate one event.
- 10 ms to fastsim (SGV) one event.

10^8 s of CPU time is needed, ie more than 3 years. This goes to 3000 years with full simulation.

Clearly, there is need to reduce this number by one or two orders of magnitude, by using generator level cuts.

SUSY parameter scans

Simple example:

- MSUGRA: 4 parameters + sign of μ
- Scan each in eg. 20 steps
- Eg. 5000 events per point (modest requirement: in sps1a' almost 1 million SUSY events are expected for 500 fb^{-1} !)
- = $20^4 \times 2 \times 5000 = 1.6 \times 10^9$ events to generate...

Slower to generate and simulate than $\gamma\gamma$ events

Also here: CPU millenniums with full simulation

SUSY parameter scans

Simple example:

- MSUGRA: 4 parameters + sign of μ
- Scan each in eg. 20 steps
- Eg. 5000 events per point (modest requirement: in sps1a' almost 1 million SUSY events are expected for 500 fb^{-1} !)
- = $20^4 \times 2 \times 5000 = 1.6 \times 10^9$ events to generate...

Slower to generate and simulate than $\gamma\gamma$ events

Also here: CPU millenniums with full simulation

Fast simulation

Different types, with different levels of sophistication:

- 4-vector smearing.
- Parametric. Eg SIMDET
- Covariance matrix machines. Eg. LiCToy, SGV

Common for all:

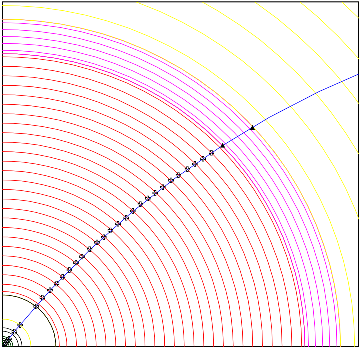
Detector simulation time $\approx$ time to generate event by an efficient generator (PYTHIA 6, SUSYGEN)

I will talk about SGV.

SGV: How it works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector, to find what layers are hit by the particle.

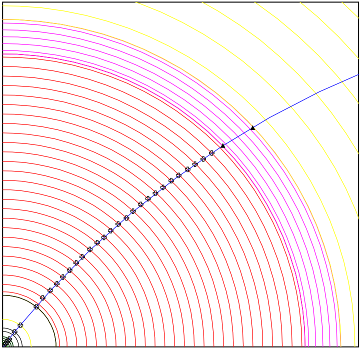


- From this, calculate cov. mat. at perigee, including effects of material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters accordingly, with Choleski decomposition (takes all correlations into account)
- Information on hit-pattern accessible to analysis. Co-ordinates of hits accessible.

SGV: How it works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector, to find what layers are hit by the particle.

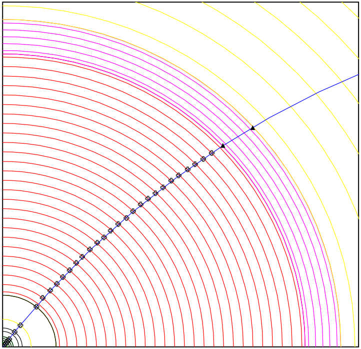


- From this, calculate cov. mat. at perigee, including effects of material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters accordingly, with Choleski decomposition (takes all correlations into account)
- Information on hit-pattern accessible to analysis. Co-ordinates of hits accessible.

SGV: How it works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector, to find what layers are hit by the particle.

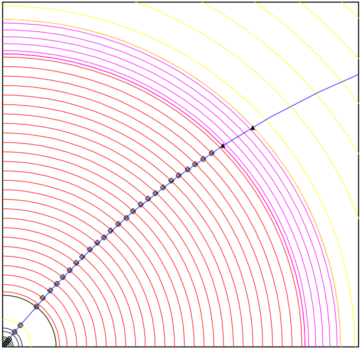


- From this, calculate cov. mat. at perigee, including effects of material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters accordingly, with Choleski decomposition (takes all correlations into account)
- Information on hit-pattern accessible to analysis. Co-ordinates of hits accessible.

SGV: How it works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector, to find what layers are hit by the particle.



- From this, calculate cov. mat. at perigee, including effects of material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters accordingly, with Choleski decomposition (takes all correlations into account)
- Information on hit-pattern accessible to analysis. Co-ordinates of hits accessible.

SGV: How it works

Calorimeters:

- Follow particle to intersection with calorimeters. Decide how the detectors will act: MIP, EM-shower, hadronic shower, below threshold, etc.
- Simulate response from parameters.
- Merge close showers
- Easy to plug in other (more sophisticated) shower-simulation

Other stuff:

- EM-interactions in detector material simulated
- Plug-ins for particle identification, track-finding efficiencies,...
- Scintilators and Taggers

SGV: How it works

Calorimeters:

- Follow particle to intersection with calorimeters. Decide how the detectors will act: MIP, EM-shower, hadronic shower, below threshold, etc.
- Simulate response from parameters.
- Merge close showers
- Easy to plug in other (more sophisticated) shower-simulation

Other stuff:

- EM-interactions in detector material simulated
- Plug-ins for particle identification, track-finding efficiencies,...
- Scintilators and Taggers

SGV: How it works

Calorimeters:

- Follow particle to intersection with calorimeters. Decide how the detectors will act: MIP, EM-shower, hadronic shower, below threshold, etc.
- Simulate response from parameters.
- Merge close showers
- Easy to plug in other (more sophisticated) shower-simulation

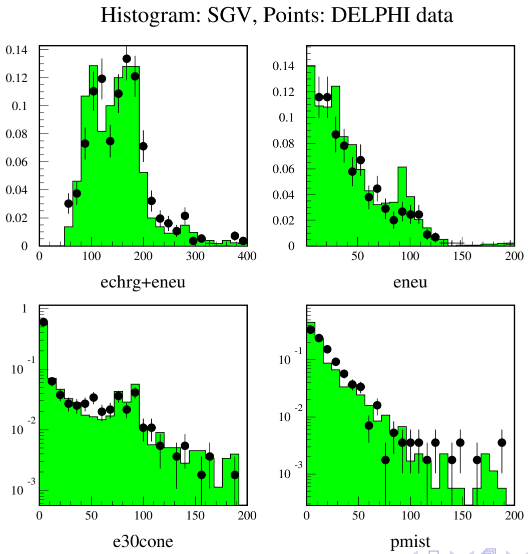
Other stuff:

- EM-interactions in detector material simulated
- Plug-ins for particle identification, track-finding efficiencies,...
- Scintilators and Taggers

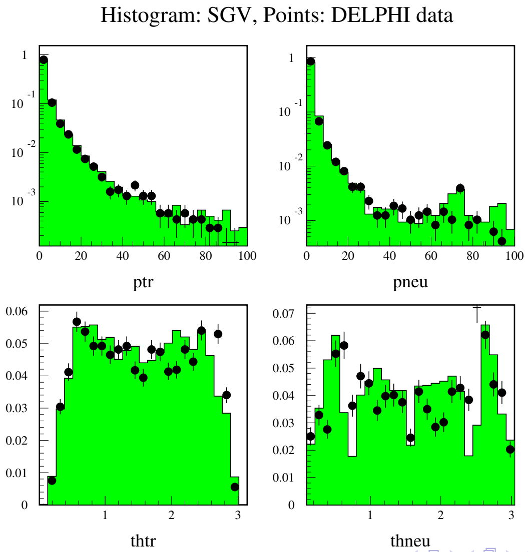
SGV physics performance

Some examples from DELPHI and ILD

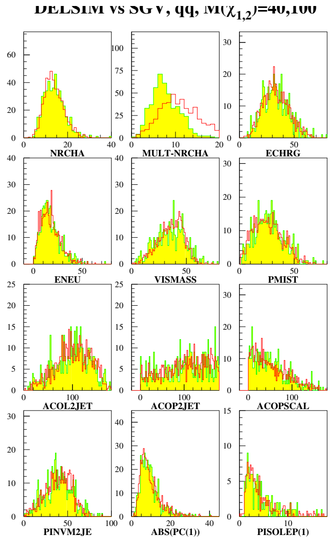
SGV and Real Data from DELPHI: Global variables

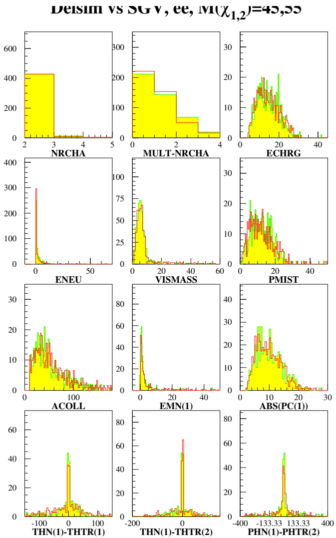


SGV and Real Data from DELPHI: Particle variables



SGV and DELSIM: Neutralino search





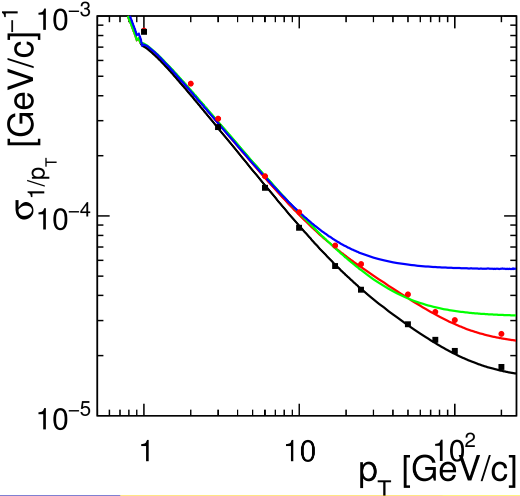
SGV for the LC: TESLA/LDC/ILD

- Used for fastsim physics studies, eg. arXiv:hep-ph/0510088, arXiv:hep-ph/0508247, arXiv:hep-ph/0406010, arXiv:hep-ph/9911345 and arXiv:hep-ph/9911344.
- Used for flavour-tagging training.
- Used for overall detector optimisation, see Eg. Vienna ECFA WS (2007), See Ilcagenda > Conference and Workshops > 2005 > ECFA Vienna Tracking
- GLD/LDC merging and LOI, see eg. Ilcagenda > Detector Design & Physics Studies > Detector Design Concepts > ILD > ILD Workshop > ILD Meeting, Cambridge > Agenda >Sub-detector Optimisation I

The latter two: Use the Covariance machine to get analytical expressions for performance (ie. *not* simulation)

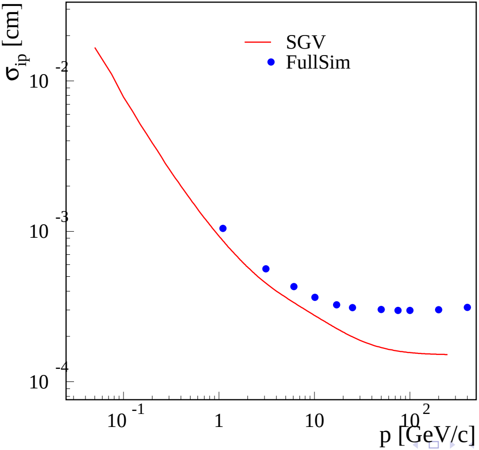
SGV and FullSim LDC/ILD: momentum resolution

Lines: SGV, dots: Mokka+Marlin



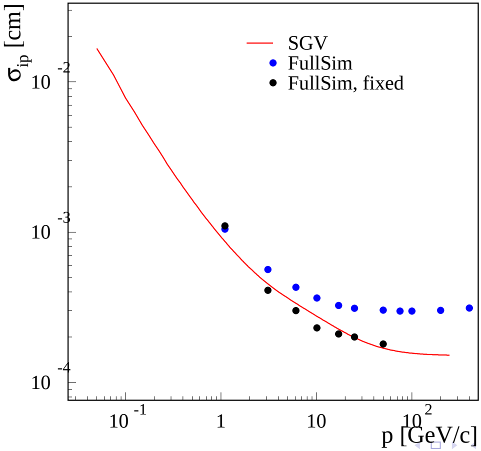
SGV and FullSim LDC/ILD: ip resolution vs P

Lines: SGV, dots: Mokka+Marlin



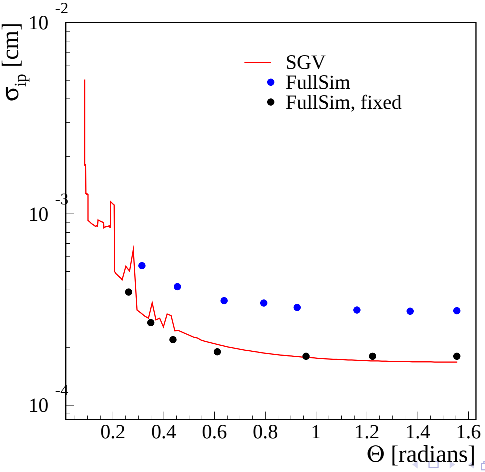
SGV and FullSim LDC/ILD: ip resolution vs P

Lines: SGV, dots: Mokka+Marlin



SGV and FullSim LDC/ILD: ip resolution vs angle

Lines: SGV, dots: Mokka+Marlin



In the past

In the past (up to v. 2.32):

- Language: FORTRAN77
- Code management: PATCHY
- Depends on CERNLIB
- Distributed as: Single compressed file (Gzip), self-installing. Download from http://berggren.web.cern.ch/berggren/sgv.html.
- 35 000 lines, installed 2.9 MB (including 1.1 MB documentation)

Recent developments

- Transformed to Fortran 95.
- Removed most CERNLIB dependence. Mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Removed some options: PYTHIA pre version 6, SUSYGEN.
- Added several features:
  - Callable Whizard.
  - Input from stdhep.
  - Output of generated event to PYJETS or stdhep.
  - Sample subdirectory with steering and code for eg. scan single particles, create hbook ntuple with "all" information (can be converted to ROOT w/ h2root). And: output LCIO DST.
  - Development on calorimeters (see later)
- Tested to work on both 32 and 64 bit out-of-the-box.
- Timing verified to be faster (by 15%) than the f77 version.
- Will be made available for svn check-out/export shortly.

Recent developments

- Transformed to Fortran 95.
- Removed most CERNLIB dependence. Mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Removed some options: PYTHIA pre version 6, SUSYGEN.
- Added several features:
  - Callable Whizard.
  - Input from stdhep.
  - Output of generated event to PYJETS or stdhep.
  - Sample subdirectory with steering and code for eg. scan single particles, create hbook ntuple with "all" information (can be converted to ROOT w/ h2root). And: output LCIO DST.
  - Development on calorimeters (see later)
- Tested to work on both 32 and 64 bit out-of-the-box.
- Timing verified to be faster (by 15%) than the f77 version.
- Will be made available for svn check-out/export shortly.

Recent developments

- Transformed to Fortran 95.
- Removed most CERNLIB dependence. Mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Removed some options: PYTHIA pre version 6, SUSYGEN.
- Added several features:
  - Callable Whizard.
  - Input from stdhep.
  - Output of generated event to PYJETS or stdhep.
  - Sample subdirectory with steering and code for eg. scan single particles, create hbook ntuple with "all" information (can be converted to ROOT w/ h2root). And: output LCIO DST.
  - Development on calorimeters (see later)
- Tested to work on both 32 and 64 bit out-of-the-box.
- Timing verified to be faster (by 15%) than the f77 version.
- Will be made available for svn check-out/export shortly.

Recent developments

- Transformed to Fortran 95.
- Removed most CERNLIB dependence. Mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Removed some options: PYTHIA pre version 6, SUSYGEN.
- Added several features:
  - Callable Whizard.
  - Input from stdhep.
  - Output of generated event to PYJETS or stdhep.
  - Sample subdirectory with steering and code for eg. scan single particles, create hbook ntuple with “all” information (can be converted to ROOT w/ h2root). And: output LCIO DST.
  - Development on calorimeters (see later)
- Tested to work on both 32 and 64 bit out-of-the-box.
- Timing verified to be faster (by 15%) than the f77 version.
- Will be made available for svn check-out/export shortly.

Recent developments

- Transformed to Fortran 95.
- Removed most CERNLIB dependence. Mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Removed some options: PYTHIA pre version 6, SUSYGEN.
- Added several features:
  - Callable Whizard.
  - Input from stdhep.
  - Output of generated event to PYJETS or stdhep.
  - Sample subdirectory with steering and code for eg. scan single particles, create hbook ntuple with "all" information (can be converted to ROOT w/ h2root). And: output LCIO DST.
  - Development on calorimeters (see later)
- Tested to work on both 32 and 64 bit out-of-the-box.
- Timing verified to be faster (by 15%) than the f77 version.
- Will be made available for svn check-out/export shortly.

Recent developments

- Transformed to Fortran 95.
- Removed most CERNLIB dependence. Mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Removed some options: PYTHIA pre version 6, SUSYGEN.
- Added several features:
  - Callable Whizard.
  - Input from stdhep.
  - Output of generated event to PYJETS or stdhep.
  - Sample subdirectory with steering and code for eg. scan single particles, create hbook ntuple with “all” information (can be converted to ROOT w/ h2root). And: output LCIO DST.
  - Development on calorimeters (see later)
- Tested to work on both 32 and 64 bit out-of-the-box.
- Timing verified to be faster (by 15%) than the f77 version.
- Will be made available for svn check-out/export shortly.

Future developments

- Update documentation and in-line comments, to reflect new structure.
- Consolidate use of Fortran 95/203/2008 features:
  - Use of user-defined types.
  - Use of PURE and ELEMENTAL routines,
  - Optimal choice between pointer, allocatable and automatic and/or assumed-size, assumed-shape, and explicit arrays.
- I/O over FIFO:s to avoid storage and I/O rate limitations.
- The Grid.
- Investigate running on GPU:s.
- Possibly - when gcc/gfortran 4.4 (ie. Fortran 2003) is common-place - Object Orientation, if there is no performance penalty.

ILD-specific Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Consequences:
  - If a (part of) a neutral cluster associated to track → Energy is lost.
  - If a (part of) a charged cluster not associated to any track → Energy is double-counted.
  - Other errors (split neutral cluster, charged cluster assoiated with wrong track ....) are of less importance.
- These features are expected to depend on
  - The 4-mom of the incomming particle.
  - The calorimeter entry point of the particle.
  - The shape of the cluster
  - The nature of the incomming particle
  - ....

ILD-specific Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Consequences:
  - If a (part of) a neutral cluster associated to track → Energy is lost.
  - If a (part of) a charged cluster not associated to any track → Energy is double-counted.
  - Other errors (split neutral cluster, charged cluster assoiated with wrong track ....) are of less importance.
- These features are expected to depend on
  - The 4-mom of the incomming particle.
  - The calorimeter entry point of the particle.
  - The shape of the cluster
  - The nature of the incomming particle
  - ....

ILD-specific Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Consequences:
  - If a (part of) a neutral cluster associated to track $\rightarrow$ Energy is lost.
  - If a (part of) a charged cluster not associated to any track $\rightarrow$ Energy is double-counted.
  - Other errors (split neutral cluster, charged cluster assoiated with wrong track ....) are of less importance.
- These features are expected to depend on
  - The 4-mom of the incomming particle.
  - The calorimeter entry point of the particle.
  - The shape of the cluster
  - The nature of the incomming particle
  - ....

ILD-specific Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Consequences:
  - If a (part of) a neutral cluster associated to track → Energy is lost.
  - If a (part of) a charged cluster not associated to any track → Energy is double-counted.
  - Other errors (split neutral cluster, charged cluster assoiated with wrong track ....) are of less importance.
- These features are expected to depend on
  - The 4-mom of the incomming particle.
  - The calorimeter entry point of the particle.
  - The shape of the cluster
  - The nature of the incomming particle
  - ....

ILD-specific Calorimeter simulation

Implementation of these mechanisms in SGV:

- SGV already
  - knows about where the particle hits the calorimeters.
  - has procdures to generate energy, position and shower-axes from geometry file input parameters.
  - has procedures to merge clusters based on generated shower positions and axes steerable by steering file.
  - has procedures to associate clusters to tracks, also steerable.
- So what is needed is mostly to determine sensible parameters:
  - Cluster energy, position and axis distributions, given 4-mom of entering particle.
  - Probability to merge two clusters given their properties
  - Probability to associate incomming tracks to (possibly merged) clusters, given incomming 4-mom and cluster properties
  - Probability to split clusters.

ILD-specific Calorimeter simulation

Implementation of these mechanisms in SGV:

- SGV already
  - knows about where the particle hits the calorimeters.
  - has procdures to generate energy, position and shower-axes from geometry file input parameters.
  - has procedures to merge clusters based on generated shower positions and axes steerable by steering file.
  - has procedures to associate clusters to tracks, also steerable.
- So what is needed is mostly to determine sensible parameters:
  - Cluster energy, position and axis distributions, given 4-mom of entering particle.
  - Probability to merge two clusters given their properties
  - Probability to associate incomming tracks to (possibly merged) clusters, given incomming 4-mom and cluster properties
  - Probability to split clusters.

ILD-specific Calorimeter simulation

Input: From full simulation and/or test-beam:

- E error for isolated (hadronic and em). Done.
- Cluster merge probability wrt. distance between true originators entering. On-going.
  - Not ideal: better to compare clusters with clusters, but difficult to know true cluster on DST.
- Track-Cluster merge probability wrt. distance between true originators and cluster props. On-going, but Pandora close to perfect, so maybe not needed.
- Split probability wrt. cluster props

ILD-specific Calorimeter simulation

Input: From full simulation and/or test-beam:

- E error for isolated (hadronic and em). Done.
- Cluster merge probability wrt. distance between true originators entering. On-going.
  - Not ideal: better to compare clusters with clusters, but difficult to know true cluster on DST.
- Track-Cluster merge probability wrt. distance between true originators and cluster props. On-going, but Pandora close to perfect, so maybe not needed.
- Split probability wrt. cluster props

ILD-specific Calorimeter simulation

Input: From full simulation and/or test-beam:

- E error for isolated (hadronic and em). Done.
- Cluster merge probability wrt. distance between true originators entering. On-going.
  - Not ideal: better to compare clusters with clusters, but difficult to know true cluster on DST.
- Track-Cluster merge probability wrt. distance between true originators and cluster props. On-going, but Pandora close to perfect, so maybe not needed.
- Split probability wrt. cluster props

ILD-specific Calorimeter simulation

Input: From full simulation and/or test-beam:

- E error for isolated (hadronic and em). Done.
- Cluster merge probability wrt. distance between true originators entering. On-going.
  - Not ideal: better to compare clusters with clusters, but difficult to know true cluster on DST.
- Track-Cluster merge probability wrt. distance between true originators and cluster props. On-going, but Pandora close to perfect, so maybe not needed.
- Split probability wrt. cluster props

ILD-specific Calorimeter simulation

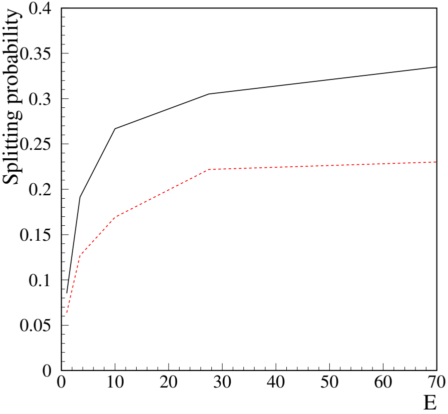
Some development to SGV from this:

- NB: zdcalo is a user-routine (with a sensible default supplied), so code-wise one can do “anything” at the level of the single incomming particle.
  - Eg.: Splitting of clusters
- In SGV core:
  - Different properties neutral-charged originator.
  - More intricate track-cluster matching (or always correct?)
  - Handling splitting: one particle in to zdcalo, several out.

Tuning to Mokka+Marlin: Pandora effects

Use LOI sample (6k udsc), compare PandoraPFO:s to MCParticles

- Probability to split cluster in two vs E
- Fraction the energy in the smaller cluster
- Distribution of fraction vs E
- Distance beteen split hadron-showers
- ... and EM

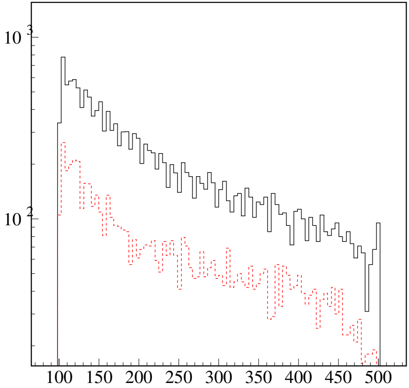


(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Pandora effects

Use LOI sample (6k udsc), compare PandoraPFO:s to MCParticles

- Probability to split cluster in two vs E
- Fraction the energy in the smaller cluster
- Distribution of fraction vs E
- Distance beteen split hadron-showers
- ... and EM

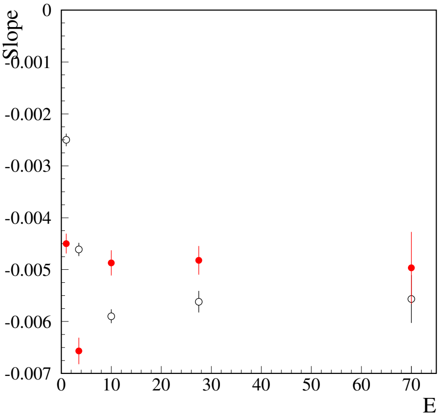


(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Pandora effects

Use LOI sample (6k udsc), compare PandoraPFO:s to MCParticles

- Probability to split cluster in two vs E
- Fraction the energy in the smaller cluster
- Distribution of fraction vs E
- Distance beteen split hadron-showers
- ... and EM

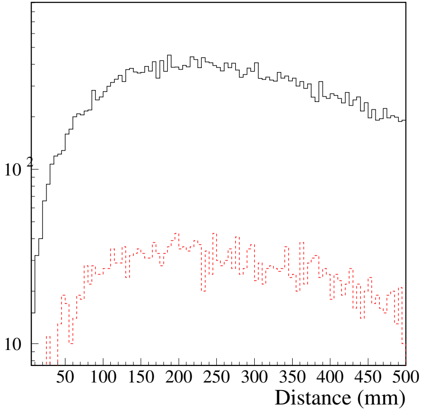


(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Pandora effects

Use LOI sample (6k udsc), compare PandoraPFO:s to MCParticles

- Probability to split cluster in two vs E
- Fraction the energy in the smaller cluster
- Distribution of fraction vs E
- Distance beteen split hadron-showers
- ... and EM

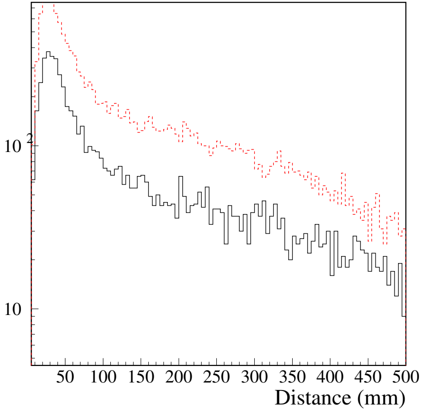


(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Pandora effects

Use LOI sample (6k udsc), compare PandoraPFO:s to MCParticles

- Probability to split cluster in two vs E
- Fraction the energy in the smaller cluster
- Distribution of fraction vs E
- Distance beteen split hadron-showers
- ... and EM

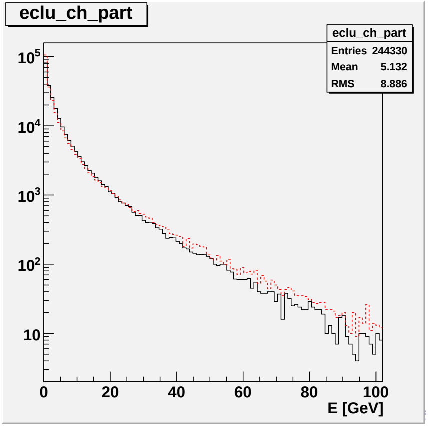


(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Tentative settings

Compare Mokka+Marlin (Red) to LCIO-DST produced by SGV, with either perfect matching (but smeared meassurements) (Black), or with tentative cluster merging and EM-interations on (Blue). NB: no splitting, yet !

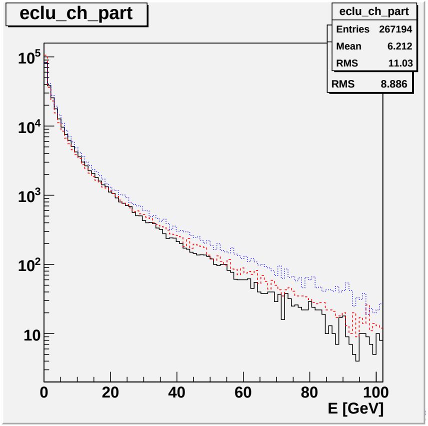
- Charged cluster energy.
- Neutral cluster energy.
- Unseen neutral energy, due to associating track to neutral
- Double-counted charged enegry due to un-associated cluster.



Tuning to Mokka+Marlin: Tentative settings

Compare Mokka+Marlin (Red) to LCIO-DST produced by SGV, with either perfect matching (but smeared meassurements) (Black), or with tentative cluster merging and EM-interations on (Blue). NB: no splitting, yet !

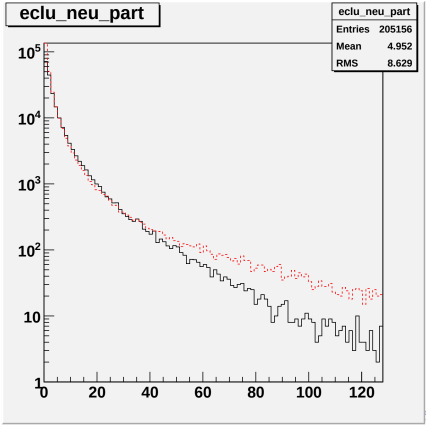
- Charged cluster energy.
- Neutral cluster energy.
- Unseen neutral energy, due to associating track to neutral
- Double-counted charged enegry due to un-associated cluster.



Tuning to Mokka+Marlin: Tentative settings

Compare Mokka+Marlin (Red) to LCIO-DST produced by SGV, with either perfect matching (but smeared meassurements) (Black), or with tentative cluster merging and EM-interations on (Blue). NB: no splitting, yet !

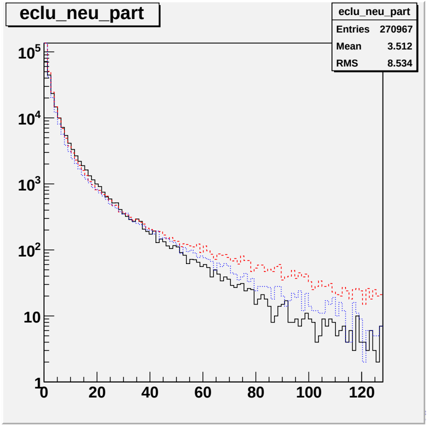
- Charged cluster energy.
- Neutral cluster energy.
- Unseen neutral energy, due to associating track to neutral
- Double-counted charged enegry due to un-associated cluster.



Tuning to Mokka+Marlin: Tentative settings

Compare Mokka+Marlin (Red) to LCIO-DST produced by SGV, with either perfect matching (but smeared meassurements) (Black), or with tentative cluster merging and EM-interations on (Blue). NB: no splitting, yet !

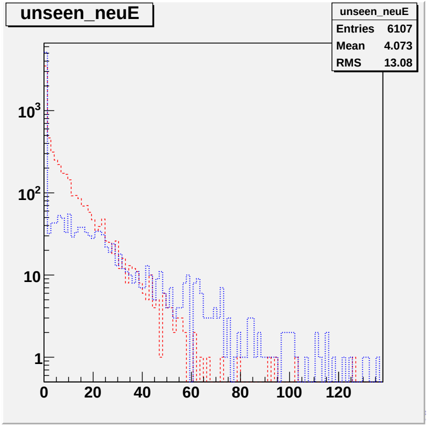
- Charged cluster energy.
- Neutral cluster energy.
- Unseen neutral energy, due to associating track to neutral
- Double-counted charged enegry due to un-associated cluster.



Tuning to Mokka+Marlin: Tentative settings

Compare Mokka+Marlin (Red) to LCIO-DST produced by SGV, with either perfect matching (but smeared meassurements) (Black), or with tentative cluster merging and EM-interations on (Blue). NB: no splitting, yet !

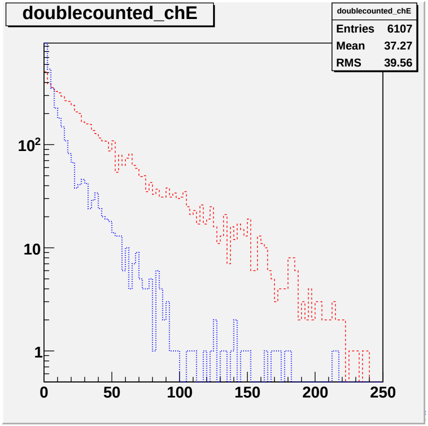
- Charged cluster energy.
- Neutral cluster energy.
- Unseen neutral energy, due to associating track to neutral
- Double-counted charged enegry due to un-associated cluster.



Tuning to Mokka+Marlin: Tentative settings

Compare Mokka+Marlin (Red) to LCIO-DST produced by SGV, with either perfect matching (but smeared meassurements) (Black), or with tentative cluster merging and EM-interations on (Blue). NB: no splitting, yet !

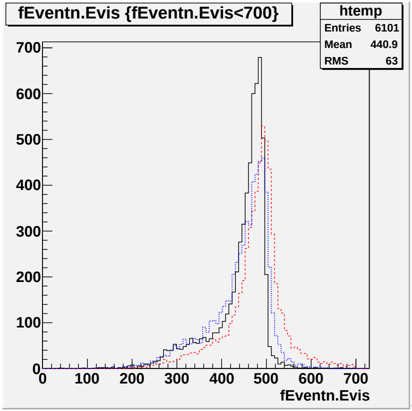
- Charged cluster energy.
- Neutral cluster energy.
- Unseen neutral energy, due to associating track to neutral
- Double-counted charged enegry due to un-associated cluster.



Tuning to Mokka+Marlin: Tentative settings

Resulting Total visible energy

Degradation seen in Full simulation reproduced. Not enough double counting, but cluster splitting not yet included.



Conclusions

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The new developents were presented: Code over-haul: F77->F95, calorimeter parametrisation, extended generator-set, and full LCIO-DST as an output-format option.
- The near future plans for SGV were presented: Further improvment in confusion simulation by allowing for splitting, and by more precise parameters. Roll-out of the SGV 3.0 SVN. Longer term plans was also mentioned.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.

Conclusions

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The new developents were presented: Code over-haul: F77->F95, calorimeter parametrisation, extended generator-set, and full LCIO-DST as an output-format option.
- The near future plans for SGV were presented: Further improvment in confusion simulation by allowing for splitting, and by more precise parameters. Roll-out of the SGV 3.0 SVN. Longer term plans was also mentioned.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.

Conclusions

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The new developents were presented: Code over-haul: F77->F95, calorimeter parametrisation, extended generator-set, and full LCIO-DST as an output-format option.
- The near future plans for SGV were presented: Further improvment in confusion simulation by allowing for splitting, and by more precise parameters. Roll-out of the SGV 3.0 SVN. Longer term plans was also mentioned.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.

Conclusions

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The new developents were presented: Code over-haul: F77->F95, calorimeter parametrisation, extended generator-set, and full LCIO-DST as an output-format option.
- The near future plans for SGV were presented: Further improvment in confusion simulation by allowing for splitting, and by more precise parameters. Roll-out of the SGV 3.0 SVN. Longer term plans was also mentioned.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.

Conclusions

Thank You !