SGV 3.0 - a fast detector simulation

Mikael Berggren[1]

[1]DESY, Hamburg

Contribution to LCWS, Granada, September 2011

Outline

1. The need for fast simulation
   - Ex1: $\gamma\gamma$ cross-sections
   - Ex2: SUSY scans
2. Fast simulation
3. Use-cases at the ILC
4. Status
   - Calorimeter simulation
5. Conclusions

The need for fast simulation

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer yesterday: *We need to update the physics case continuously*.
  - 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 fast simulation ?
- Answer:
  - R. Heuer yesterday: *We need to update the physics case continuously*.
  - 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 fast simulation ?
- Answer:
  - R. Heuer yesterday: *We need to update the physics case continuously*.
  - 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 fast simulation ?
- Answer:
  - R. Heuer yesterday: *We need to update the physics case continuously*.
  - 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 fast simulation ?
- Answer:
  - R. Heuer yesterday: *We need to update the physics case continuously*.
  - 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. But:This goes to 3000 years with full simulation.

Cross-section and event-generation time

PYTHIA obtains a total cross-section for $e^+e^- \to \gamma\gamma e^+e^- \to q\bar{q}e^+e^-$ 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} $\to$ 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. But:This goes to 3000 years with full simulation.

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. But:This goes to 3000 years 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

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

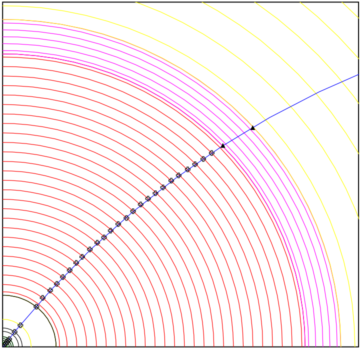
Common for all:

Detector simulation time $\approx$ time to generate event by an efficient generator like PYTHIA 6

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.

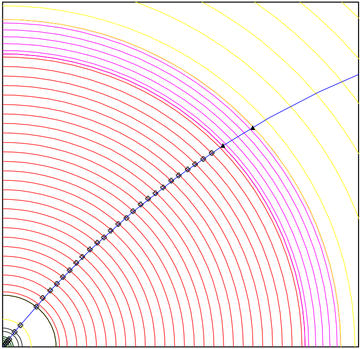


- From this, calculate cov. mat. at perigee, including effects of material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters accordingly, with Choleski decomposition (takes all correlations into account)
- Information on hit-pattern accessible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

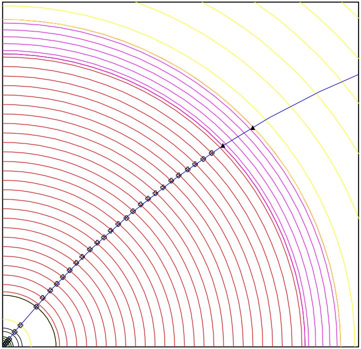


- From this, calculate cov. mat. at perigee, including effects of material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters accordingly, with Choleski decomposition (takes all correlations into account)
- Information on hit-pattern accessible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

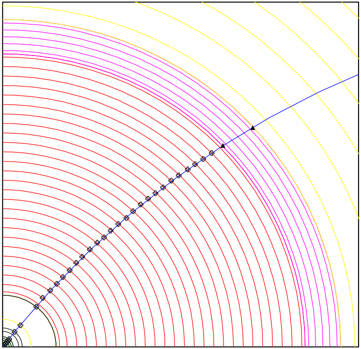


- From this, calculate cov. mat. at perigee, including effects of material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters accordingly, with Choleski decomposition (takes all correlations into account)
- Information on hit-pattern accessible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

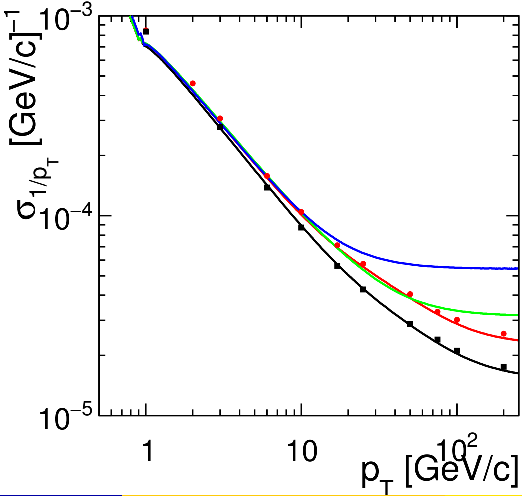
Use-cases at the ILC

- 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)

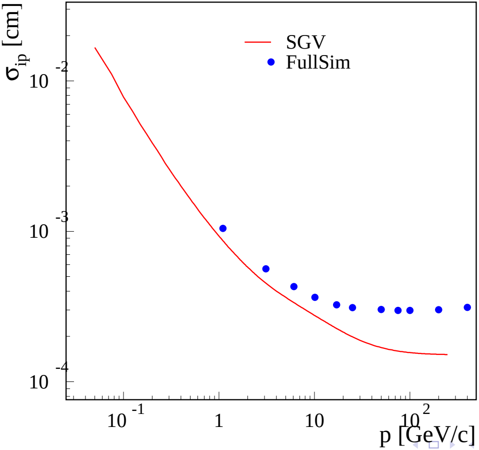
SGV and FullSim LDC/ILD: momentum resolution

Lines: SGV, dots: Mokka+Marlin



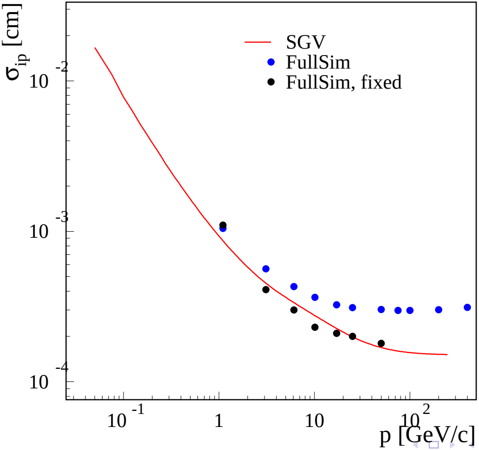
SGV and FullSim LDC/ILD: ip resolution vs P

Lines: SGV, dots: Mokka+Marlin



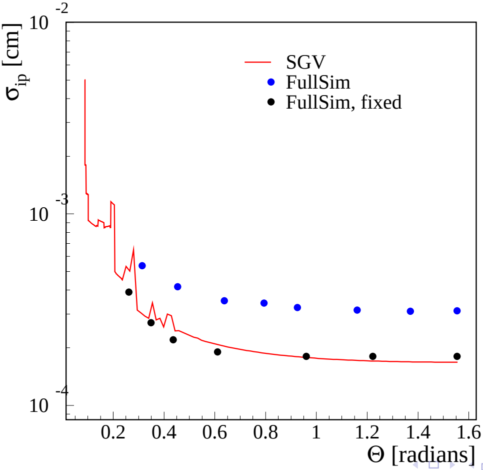
SGV and FullSim LDC/ILD: ip resolution vs P

Lines: SGV, dots: Mokka+Marlin



SGV and FullSim LDC/ILD: ip resolution vs angle

Lines: SGV, dots: Mokka+Marlin



White paper

- Written in Fortran 95.
- CERNLIB dependence. Much reduced wrt. old F77 version, mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Features:
  - Callable PYTHIA, Whizard.
  - Input from PYJETS or stdhep.
  - Output of generated event to PYJETS or stdhep.
  - samples 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.

White paper

- Written in Fortran 95.
- CERNLIB dependence. Much reduced wrt. old F77 version, mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Features:
  - Callable PYTHIA, Whizard.
  - Input from PYJETS or stdhep.
  - Output of generated event to PYJETS or stdhep.
  - samples 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.

White paper

- Written in Fortran 95.
- CERNLIB dependence. Much reduced wrt. old F77 version, mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Features:
  - Callable PYTHIA, Whizard.
  - Input from PYJETS or stdhep.
  - Output of generated event to PYJETS or stdhep.
  - samples 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.

White paper

- Written in Fortran 95.
- CERNLIB dependence. Much reduced wrt. old F77 version, mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Features:
  - Callable PYTHIA, Whizard.
  - Input from PYJETS or stdhep.
  - Output of generated event to PYJETS or stdhep.
  - samples 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.

White paper

- Written in Fortran 95.
- CERNLIB dependence. Much reduced wrt. old F77 version, mostly by using Fortran 95's built-in matrix algebra.
- Managed in SVN.Install script included.
- Features:
  - Callable PYTHIA, Whizard.
  - Input from PYJETS or stdhep.
  - Output of generated event to PYJETS or stdhep.
  - samples 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.

Installing SGV

```
svn export https://svnsrv.desy.de/public/sgv/tags/SGV-3.0rc1/
SGV-3.0rc1/
```

Then

```
bash install
```

This will take you about a minute ...
Study README, and README in the samples sub-directory, to eg.:

- Get STDHEP installed.
- Get CERNLIB installed in native 64bit.
- Get Whizard (basic or ILC-tuned) installed, with complications solved.
- Get the LCIO-DST writer set up

Installing SGV

```
svn export https://svnsrv.desy.de/public/sgv/tags/SGV-3.0rc1/
SGV-3.0rc1/
```

Then

```
bash install
```

This will take you about a minute ...
Study README, and README in the samples sub-directory, to eg.:

- Get STDHEP installed.
- Get CERNLIB installed in native 64bit.
- Get Whizard (basic or ILC-tuned) installed, with complications solved.
- Get the LCIO-DST writer set up

Installing SGV

```
svn export https://svnsrv.desy.de/public/sgv/tags/SGV-3.0rc1/
SGV-3.0rc1/
```

Then

```
bash install
```

This will take you about a minute ...
Study README, and README in the samples sub-directory, to eg.:

- Get STDHEP installed.
- Get CERNLIB installed in native 64bit.
- Get Whizard (basic or ILC-tuned) installed, with complications solved.
- Get the LCIO-DST writer set up

Installing SGV

```
svn export https://svnsrv.desy.de/public/sgv/tags/SGV-3.0rc1/
SGV-3.0rc1/
```

Then

```
bash install
```

This will take you about a minute ...
Study README, and README in the samples sub-directory, to eg.:

- Get STDHEP installed.
- Get CERNLIB installed in native 64bit.
- Get Whizard (basic or ILC-tuned) installed, with complications solved.
- Get the LCIO-DST writer set up

Future developments

- Include a filter-mode:
  - Generate event inside SGV.
  - Run SGV detector simulation and analysis.
  - Decide what to do: Fill some histos, fill ntuple, output LCIO, or better do full sim
  - In the last case: output STDHEP of event
- Update documentation and in-line comments, to reflect new structure.
- Consolidate use of Fortran 95/203/2008 features. Possibly - when gcc/gfortran 4.4 (ie. Fortran 2003) is common-place - Object Orientation, if there is no performance penalty.
- Further reduce CERNLIB dependence - at a the cost of backward compatibility on steering files ? HBOOK dependence will remain in the forseable future - but only for user convenience : SGV itself doesn't need it.

Future developments

- Include a filter-mode:
  - Generate event inside SGV.
  - Run SGV detector simulation and analysis.
  - Decide what to do: Fill some histos, fill ntuple, output LCIO, or better do full sim
  - In the last case: output STDHEP of event
- Update documentation and in-line comments, to reflect new structure.
- Consolidate use of Fortran 95/203/2008 features. Possibly - when gcc/gfortran 4.4 (ie. Fortran 2003) is common-place - Object Orientation, if there is no performance penalty.
- Further reduce CERNLIB dependence - at a the cost of backward compatibility on steering files ? HBOOK dependence will remain in the forseable future - but only for user convenience : SGV itself doesn't need it.

Future developments

- Include a filter-mode:
  - Generate event inside SGV.
  - Run SGV detector simulation and analysis.
  - Decide what to do: Fill some histos, fill ntuple, output LCIO, or better do full sim
  - In the last case: output STDHEP of event
- Update documentation and in-line comments, to reflect new structure.
- Consolidate use of Fortran 95/203/2008 features. Possibly - when gcc/gfortran 4.4 (ie. Fortran 2003) is common-place - Object Orientation, if there is no performance penalty.
- Further reduce CERNLIB dependence - at a the cost of backward compatibility on steering files ? HBOOK dependence will remain in the forseable future - but only for user convenience : SGV itself doesn't need it.

Future developments

- Include a filter-mode:
  - Generate event inside SGV.
  - Run SGV detector simulation and analysis.
  - Decide what to do: Fill some histos, fill ntuple, output LCIO, or better do full sim
  - In the last case: output STDHEP of event
- Update documentation and in-line comments, to reflect new structure.
- Consolidate use of Fortran 95/203/2008 features. Possibly - when gcc/gfortran 4.4 (ie. Fortran 2003) is common-place - Object Orientation, if there is no performance penalty.
- Further reduce CERNLIB dependence - at a the cost of backward compatibility on steering files ? HBOOK dependence will remain in the forseable future - but only for user convenience : SGV itself doesn't need it.

Future developments

- Include a filter-mode:
  - Generate event inside SGV.
  - Run SGV detector simulation and analysis.
  - Decide what to do: Fill some histos, fill ntuple, output LCIO, or better do full sim
  - In the last case: output STDHEP of event
- Update documentation and in-line comments, to reflect new structure.
- Consolidate use of Fortran 95/203/2008 features. Possibly - when gcc/gfortran 4.4 (ie. Fortran 2003) is common-place - Object Orientation, if there is no performance penalty.
- Further reduce CERNLIB dependence - at a the cost of backward compatibility on steering files ? HBOOK dependence will remain in the forseable future - but only for user convenience : SGV itself doesn't need it.

Future developments

- Include a filter-mode:
  - Generate event inside SGV.
  - Run SGV detector simulation and analysis.
  - Decide what to do: Fill some histos, fill ntuple, output LCIO, or better do full sim
  - In the last case: output STDHEP of event
- Update documentation and in-line comments, to reflect new structure.
- Consolidate use of Fortran 95/203/2008 features. Possibly - when gcc/gfortran 4.4 (ie. Fortran 2003) is common-place - Object Orientation, if there is no performance penalty.
- Further reduce CERNLIB dependence - at a the cost of backward compatibility on steering files ? HBOOK dependence will remain in the forseable future - but only for user convenience : SGV itself doesn't need it.

Future developments

- Include a filter-mode:
  - Generate event inside SGV.
  - Run SGV detector simulation and analysis.
  - Decide what to do: Fill some histos, fill ntuple, output LCIO, or better do full sim
  - In the last case: output STDHEP of event
- Update documentation and in-line comments, to reflect new structure.
- Consolidate use of Fortran 95/203/2008 features. Possibly - when gcc/gfortran 4.4 (ie. Fortran 2003) is common-place - Object Orientation, if there is no performance penalty.
- Further reduce CERNLIB dependence - at a the cost of backward compatibility on steering files ? HBOOK dependence will remain in the forseable future - but only for user convenience : SGV itself doesn't need it.

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 → 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

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, and the local density of clusters.
  - 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, and the local density of clusters.
  - 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.
  - So: Create true clusters from CalorimeterHits, associated to MCParticles (optional output from RecoMCTruthLinker. Then calculate E and axes of "tensor of inertia"
- Split probability wrt. cluster props
- Track-Cluster merge probability wrt. distance between true originators and cluster props. On-going, but Pandora close to perfect, so maybe not needed.
- Decide when to reject tracks, based on risk to make a mistake. Also here: Pandora very good, but hard to parametrisise. On-going.

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.
  - So: Create true clusters from CalorimeterHits, associated to MCParticles (optional output from RecoMCTruthLinker. Then calculate E and axes of "tensor of inertia"
- Split probability wrt. cluster props
- Track-Cluster merge probability wrt. distance between true originators and cluster props. On-going, but Pandora close to perfect, so maybe not needed.
- Decide when to reject tracks, based on risk to make a mistake. Also here: Pandora very good, but hard to parametrisise. On-going.

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.
  - So: Create true clusters from CalorimeterHits, associated to MCParticles (optional output from RecoMCTruthLinker. Then calculate E and axes of “tensor of inertia”
- Split probability wrt. cluster props
- Track-Cluster merge probability wrt. distance between true originators and cluster props. On-going, but Pandora close to perfect, so maybe not needed.
- Decide when to reject tracks, based on risk to make a mistake. Also here: Pandora very good, but hard to parametrisise. On-going.

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.
  - So: Create true clusters from CalorimeterHits, associated to MCParticles (optional output from RecoMCTruthLinker. Then calculate E and axes of "tensor of inertia"
- Split probability wrt. cluster props
- Track-Cluster merge probability wrt. distance between true originators and cluster props. On-going, but Pandora close to perfect, so maybe not needed.
- Decide when to reject tracks, based on risk to make a mistake. Also here: Pandora very good, but hard to parametrisise. On-going.

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.
  - So: Create true clusters from CalorimeterHits, associated to MCParticles (optional output from RecoMCTruthLinker. Then calculate E and axes of “tensor of inertia”
- Split probability wrt. cluster props
- Track-Cluster merge probability wrt. distance between true originators and cluster props. On-going, but Pandora close to perfect, so maybe not needed.
- Decide when to reject tracks, based on risk to make a mistake. Also here: Pandora very good, but hard to parametrisise. On-going.

Tuning to Mokka+Marlin

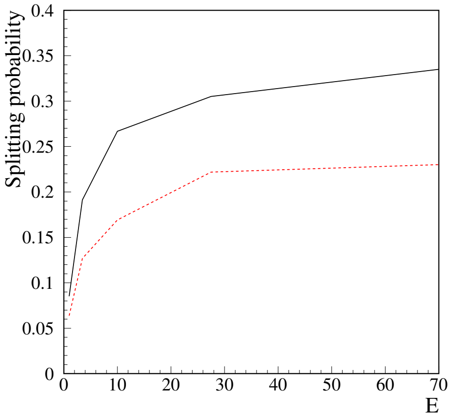
- Use LOI sample (6k udsc), compare PandoraPFO:s to MCParticles.
- Also have the same sample re-reconstructed with PandoraNew.
- Also: possibility to replace SGV:s detector simulation by Mokka:
  - From LCIO : MCParticles, Tracks, CalorimeterHits
  - Create true clusters
  - Fill and output SGV's internal data-structure
  - Read this back with SGV's “input simulated event” feature.
  - Analyse in SGV.

Tuning to Mokka+Marlin

- Use LOI sample (6k udsc), compare PandoraPFO:s to MCParticles.
- Also have the same sample re-reconstructed with PandoraNew.
- Also: possibility to replace SGV:s detector simulation by Mokka:
  - From LCIO : MCParticles, Tracks, CalorimeterHits
  - Create true clusters
  - Fill and output SGV's internal data-structure
  - Read this back with SGV's "input simulated event" feature.
  - Analyse in SGV.

Tuning to Mokka+Marlin: Splitting

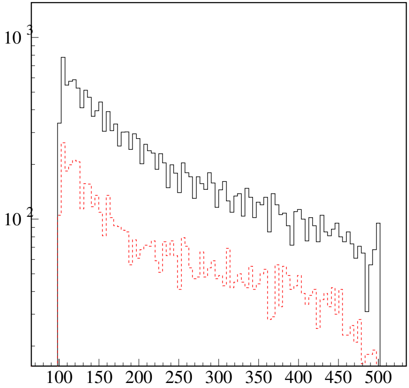
- Probability to split cluster in two vs E
- Fraction the energy in the smaller cluster
- Distribution of fraction vs E
- Distance beteen split hadron-showers
- ... and EM



(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Splitting

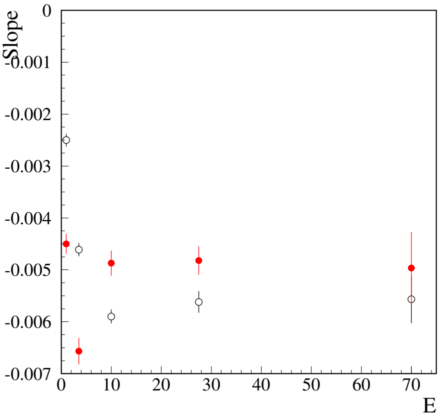
- Probability to split cluster in two vs E
- Fraction the energy in the smaller cluster
- Distribution of fraction vs E
- Distance beteen split hadron-showers
- ... and EM



(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Splitting

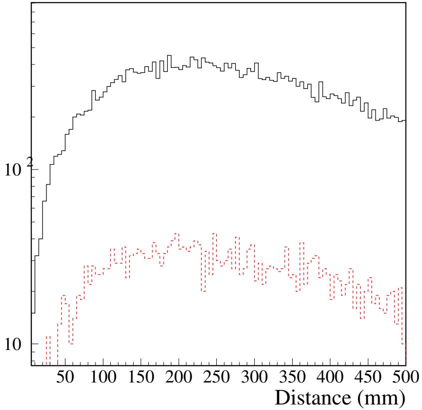
- Probability to split cluster in two vs E
- Fraction the energy in the smaller cluster
- Distribution of fraction vs E
- Distance beteen split hadron-showers
- ... and EM



(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Splitting

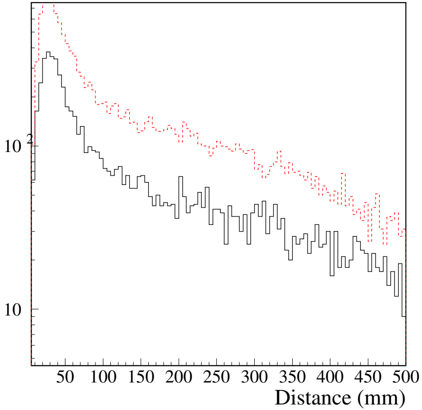
- Probability to split cluster in two vs E
- Fraction the energy in the smaller cluster
- Distribution of fraction vs E
- Distance beteen split hadron-showers
- ... and EM



(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Splitting

- 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

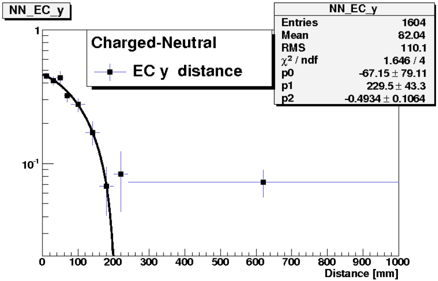


(Black solid: Charged, Red dashed: Neutral)

Tuning to Mokka+Marlin: Merging

Probability to merge a Cluster created by a neutral into a charged PFO as a function of the true track-neutral distance at the entrance of the calorimeter for:

- Hadrons ...
- and photons.

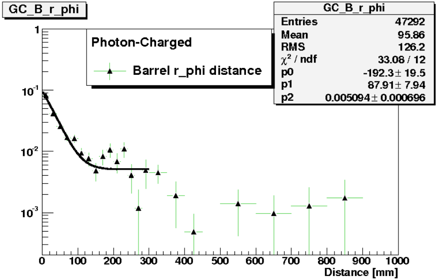


(From M. Chera, DESY)

Tuning to Mokka+Marlin: Merging

Probability to merge a Cluster created by a neutral into a charged PFO as a function of the true track-neutral distance at the entrance of the calorimeter for:

- Hadrons ...
- and photons.

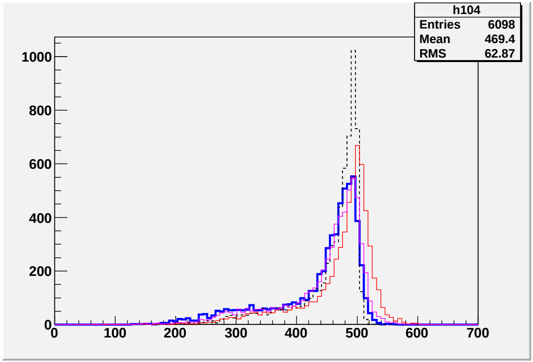


(From M. Chera, DESY)

Tuning to Mokka+Marlin: Where we are

Resulting Total visible energy

- Blue: Complete Full simulation.
- Black dash: SGV "perfect".
- Magenta: Mokka+SGV
- Red : SGV with detector-interactions and cluster-spliting.



Work is on-going, but results are already 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.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.
- The near future plans for SGV were presented: Further improvment in confusion simulation by more precise parameters. Implemetation of a Filter-mode. Longer term plans was also mentioned.

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.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.
- The near future plans for SGV were presented: Further improvment in confusion simulation by more precise parameters. Implemetation of a Filter-mode. Longer term plans was also mentioned.

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.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.
- The near future plans for SGV were presented: Further improvment in confusion simulation by more precise parameters. Implemetation of a Filter-mode. Longer term plans was also mentioned.

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.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.
- The near future plans for SGV were presented: Further improvment in confusion simulation by more precise parameters. Implemetation of a Filter-mode. Longer term plans was also mentioned.

Conclusions

Thank You !