Latest developments for SGV

Mikael Berggren[1]

[1]DESY, Hamburg

European Linear Collider Workshop ECFA LC2013 , DESY , May 2013

Outline

Features of “la Simulation à Grande Vitesse”

- Typical generation+simulation+reconstruction time $\mathcal{O}(10)$ ms.
- Fully configurable discs ’n’ cylinders geometry.
- Event generation: Callable PYTHIA, Whizard or input from PYJETS or stdhep.
- Output of generated event to PYJETS or stdhep.
- samples subdirectory with steering and code for:
  - Particle-flow parametrisation.
  - SUSY with PYTHIA
  - Whizard integration.
  - output LCIO DST
  - NEW : Beam-spectrum for PYTHIA.
  - NEW : BeamCal simulation including effects of pair-background.
  - Tools and examples.
- Written in Fortran 95.
- Some CERNLIB dependence.
- Managed in SVN.Install script included.
- Timing verified to be faster (by 15%) than the f77 version.

Features of "la Simulation à Grande Vitesse"

- Typical generation+simulation+reconstruction time $\mathcal{O}(10)$ ms.
- Fully configurable discs 'n' cylinders geometry.
- Event generation: Callable PYTHIA, Whizard or input from PYJETS or stdhep.
- Output of generated event to PYJETS or stdhep.
- samples subdirectory with steering and code for:
  - Particle-flow parametrisation.
  - SUSY with PYTHIA
  - Whizard integration.
  - output LCIO DST
  - NEW : Beam-spectrum for PYTHIA.
  - NEW : BeamCal simulation including effects of pair-background.
  - Tools and examples.
- Written in Fortran 95.
- Some CERNLIB dependence.
- Managed in SVN.Install script included.
- Timing verified to be faster (by 15%) than the f77 version.

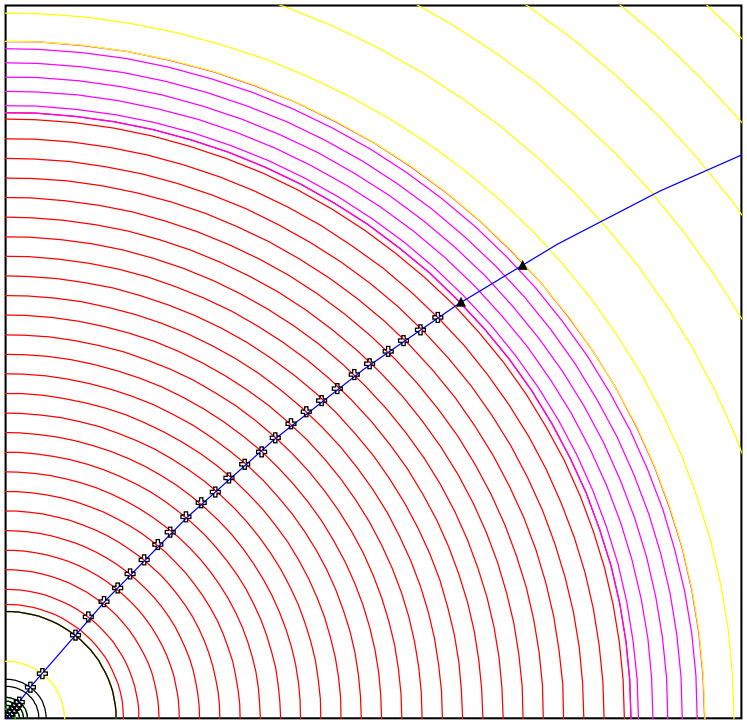
Features of "la Simulation à Grande Vitesse"

- Typical generation+simulation+reconstruction time $\mathcal{O}(10)$ ms.
- Fully configurable discs 'n' cylinders geometry.
- Event generation: Callable PYTHIA, Whizard or input from PYJETS or stdhep.
- Output of generated event to PYJETS or stdhep.
- samples subdirectory with steering and code for:
  - Particle-flow parametrisation.
  - SUSY with PYTHIA
  - Whizard integration.
  - output LCIO DST
  - NEW : Beam-spectrum for PYTHIA.
  - NEW : BeamCal simulation including effects of pair-background.
  - Tools and examples.
- Written in Fortran 95.
- Some CERNLIB dependence.
- Managed in SVN.Install script included.
- Timing verified to be faster (by 15%) than the f77 version.

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Follow track-helix through the detector-layers



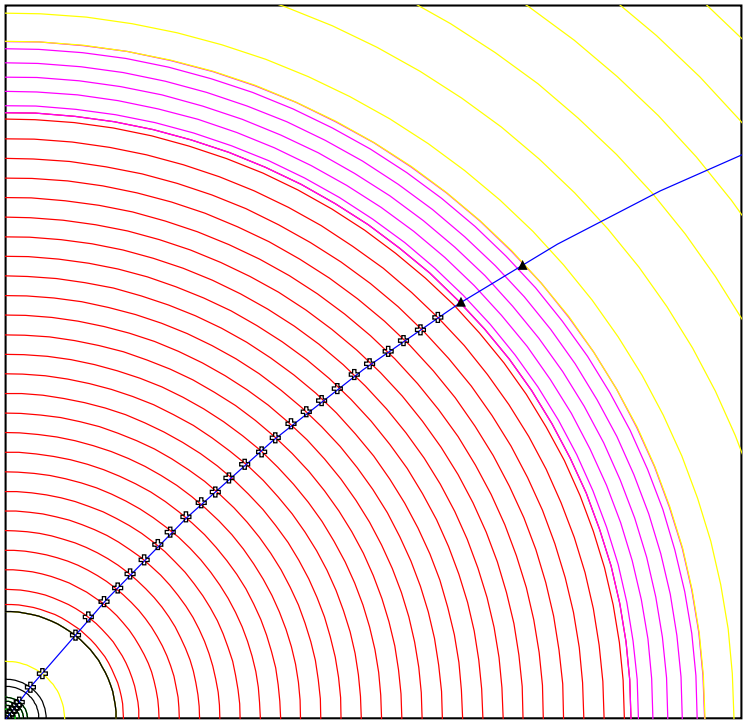
(Fringe benefit of stepping: EM-interactions in detector layers simulated)

- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation.
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: helix moved to (0,0,0) for this.
- Other stuff:
  - Plug-ins for particle identification, track-finding efficiencies,...
  - Information on hits accessible to analysis.

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Follow track-helix through the detector-layers



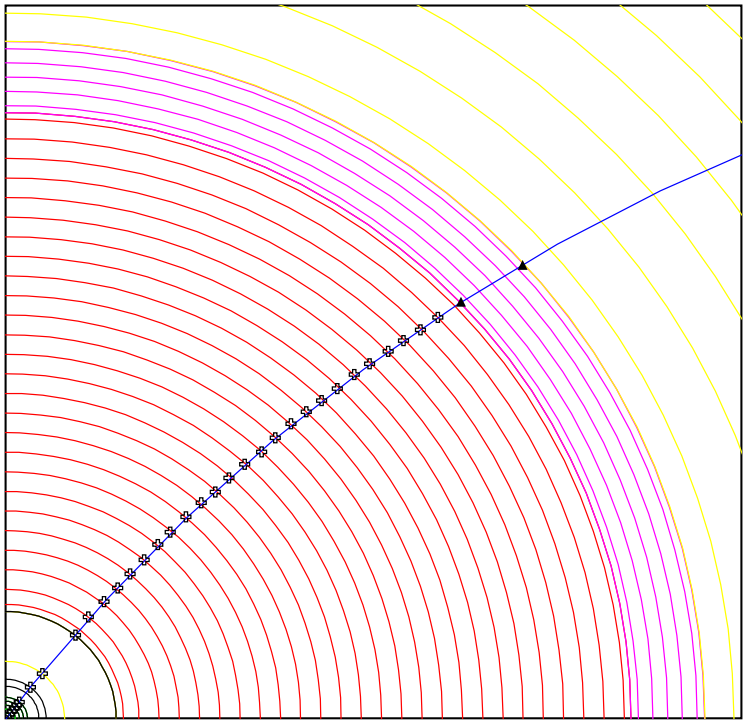
(Fringe benefit of stepping: EM-interactions in detector layers simulated)

- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation.
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: helix moved to (0,0,0) for this.
- Other stuff:
  - Plug-ins for particle identification, track-finding efficiencies,...
  - Information on hits accessible to analysis.

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Follow track-helix through the detector-layers



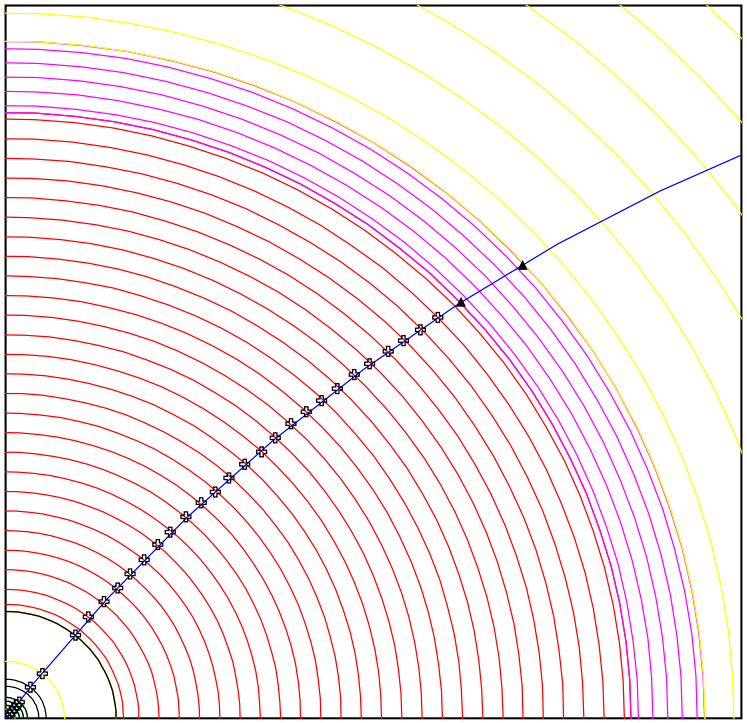
(Fringe benefit of stepping: EM-interactions in detector layers simulated)

- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation.
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: helix moved to (0,0,0) for this.
- Other stuff:
  - Plug-ins for particle identification, track-finding efficiencies,...
  - Information on hits accessible to analysis.

SGV: How tracking works

SGV is a machine to calculate covariance matrices

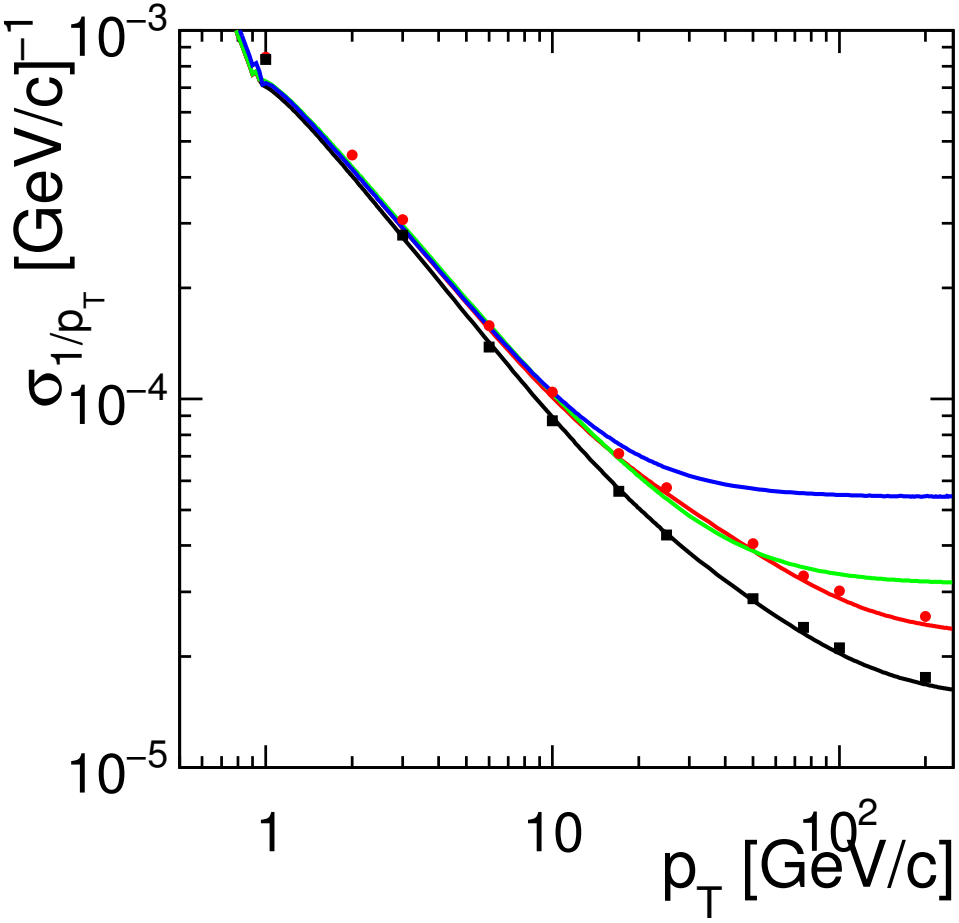
Follow track-helix through the detector-layers

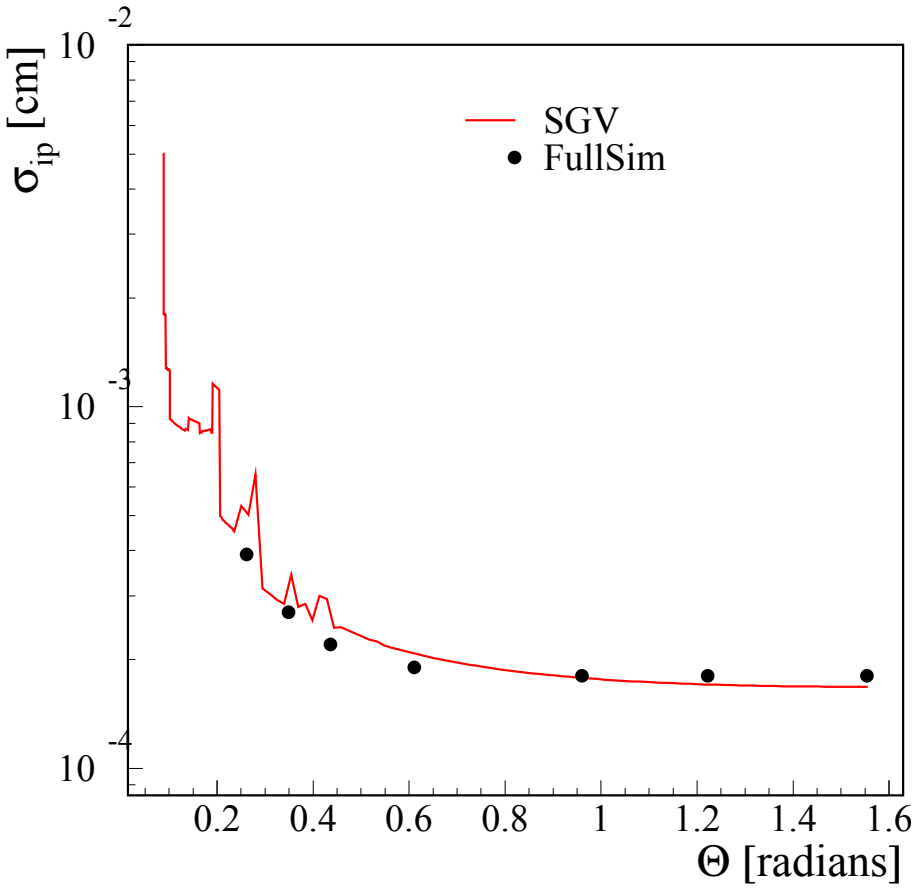


(Fringe benefit of stepping: EM-interactions in detector layers simulated)

- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation.
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: helix moved to (0,0,0) for this.
- Other stuff:
  - Plug-ins for particle identification, track-finding efficiencies,...
  - Information on hits accessible to analysis.

SGV and FullSim: P_T and D_0 resolution





Lines: SGV, dots: Mokka+Marlin

Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape. Controlled by the geometry-file.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Will depend on Energy, on distance to neighbour, on EM or hadronic, on Barrel or forward, ...
- 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 associated with wrong track ....) are of less importance.

Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape. Controlled by the geometry-file.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Will depend on Energy, on distance to neighbour, on EM or hadronic, on Barrel or forward, ...
- 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 associated with wrong track ....) are of less importance.

Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape. Controlled by the geometry-file.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Will depend on Energy, on distance to neighbour, on EM or hadronic, on Barrel or forward, ...
- 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 associated with wrong track ....) are of less importance.

Study and Parametrisation of PFA

Look at how PFA on FullSim has associated tracks and clusters:

- From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
- Use RecoMCTruthLinker, with all switches on: Full relations.
- Create true clusters:
  - Each MCParticle is connected to a set of clusters made of CalorimeterHits created by this true particle only.
  - Each of these clusters contribute to only one Pandora cluster.

With this info:

- Identify, factorise and parametrise:
  1. Probability to split
  2. If split, probability to split off/merge the entire cluster.
  3. If split, but not 100 %: Form of the p.d.f. of the fraction split off.
- All cases (EM/had - split/merge - Barrel/endcap) can be described by the same functional shapes.
- Functions are combinations of exponentials and lines.
- 28 parameters $\times$ 4 cases (em/had $\times$ double-counting/loss)

Study and Parametrisation of PFA

Look at how PFA on FullSim has associated tracks and clusters:

- From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
- Use RecoMCTruthLinker, with all switches on: Full relations.
- Create true clusters:
  - Each MCParticle is connected to a set of clusters made of CalorimeterHits created by this true particle only.
  - Each of these clusters contribute to only one Pandora cluster.

With this info:

- Identify, factorise and parametrise:
  1. Probability to split
  2. If split, probability to split off/merge the entire cluster.
  3. If split, but not 100 %: Form of the p.d.f. of the fraction split off.
- All cases (EM/had - split/merge - Barrel/endcap) can be described by the same functional shapes.
- Functions are combinations of exponentials and lines.
- 28 parameters $\times$ 4 cases (em/had $\times$ double-counting/loss)

Study and Parametrisation of PFA

Look at how PFA on FullSim has associated tracks and clusters:

- From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
- Use RecoMCTruthLinker, with all switches on: Full relations.
- Create true clusters:
  - Each MCParticle is connected to a set of clusters made of CalorimeterHits created by this true particle only.
  - Each of these clusters contribute to only one Pandora cluster.

With this info:

- Identify, factorise and parametrise:
  1. Probability to split
  2. If split, probability to split off/merge the entire cluster.
  3. If split, but not 100 %: Form of the p.d.f. of the fraction split off.
- All cases (EM/had - split/merge - Barrel/endcap) can be described by the same functional shapes.
- Functions are combinations of exponentials and lines.
- 28 parameters $\times$ 4 cases (em/had $\times$ double-counting/loss)

Study and Parametrisation of PFA

Look at how PFA on FullSim has associated tracks and clusters:

- From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
- Use RecoMCTruthLinker, with all switches on: Full relations.
- Create true clusters:
  - Each MCParticle is connected to a set of clusters made of CalorimeterHits created by this true particle only.
  - Each of these clusters contribute to only one Pandora cluster.

With this info:

- Identify, factorise and parametrise:
  1. Probability to split
  2. If split, probability to split off/merge the entire cluster.
  3. If split, but not 100 %: Form of the p.d.f. of the fraction split off.
- All cases (EM/had - split/merge - Barrel/endcap) can be described by the same functional shapes.
- Functions are combinations of exponentials and lines.
- 28 parameters $\times$ 4 cases (em/had $\times$ double-counting/loss)

Study and Parametrisation of PFA

Look at how PFA on FullSim has associated tracks and clusters:

- From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
- Use RecoMCTruthLinker, with all switches on: Full relations.
- Create true clusters:
  - Each MCParticle is connected to a set of clusters made of CalorimeterHits created by this true particle only.
  - Each of these clusters contribute to only one Pandora cluster.

With this info:

- Identify, factorise and parametrise:
  1. Probability to split
  2. If split, probability to split off/merge the entire cluster.
  3. If split, but not 100 %: Form of the p.d.f. of the fraction split off.
- All cases (EM/had - split/merge - Barrel/endcap) can be described by the same functional shapes.
- Functions are combinations of exponentials and lines.
- 28 parameters $\times$ 4 cases (em/had $\times$ double-counting/loss)

Checking the parametrisation

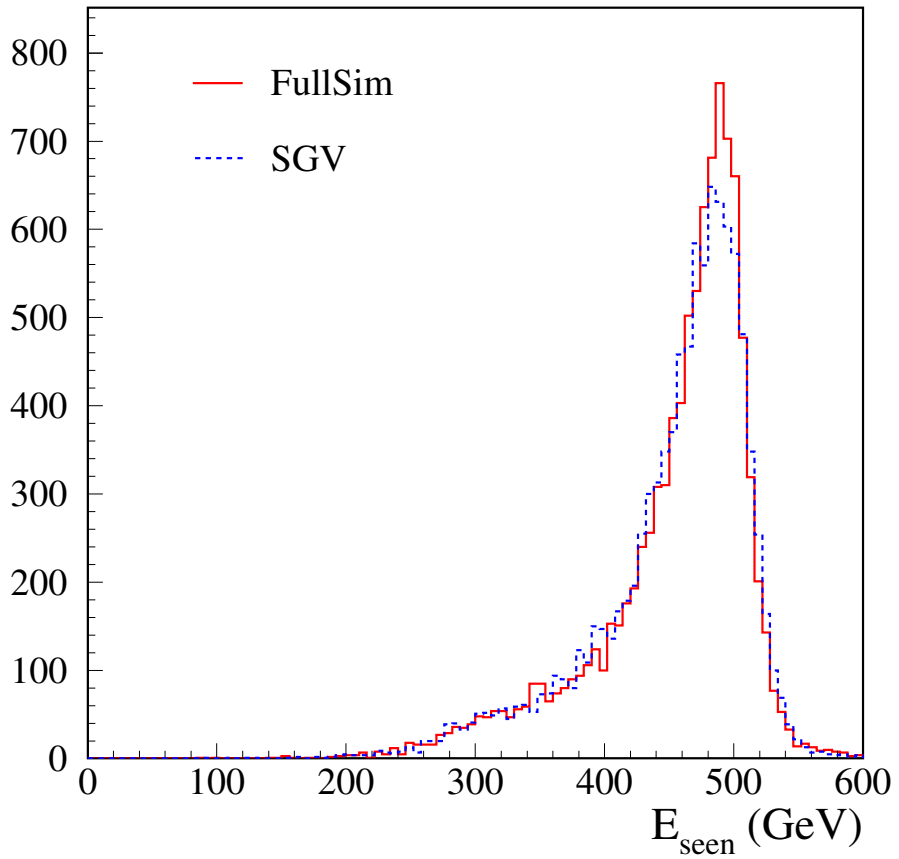
Feed exactly the same physics events through FullSim or SGV.

- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag

Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

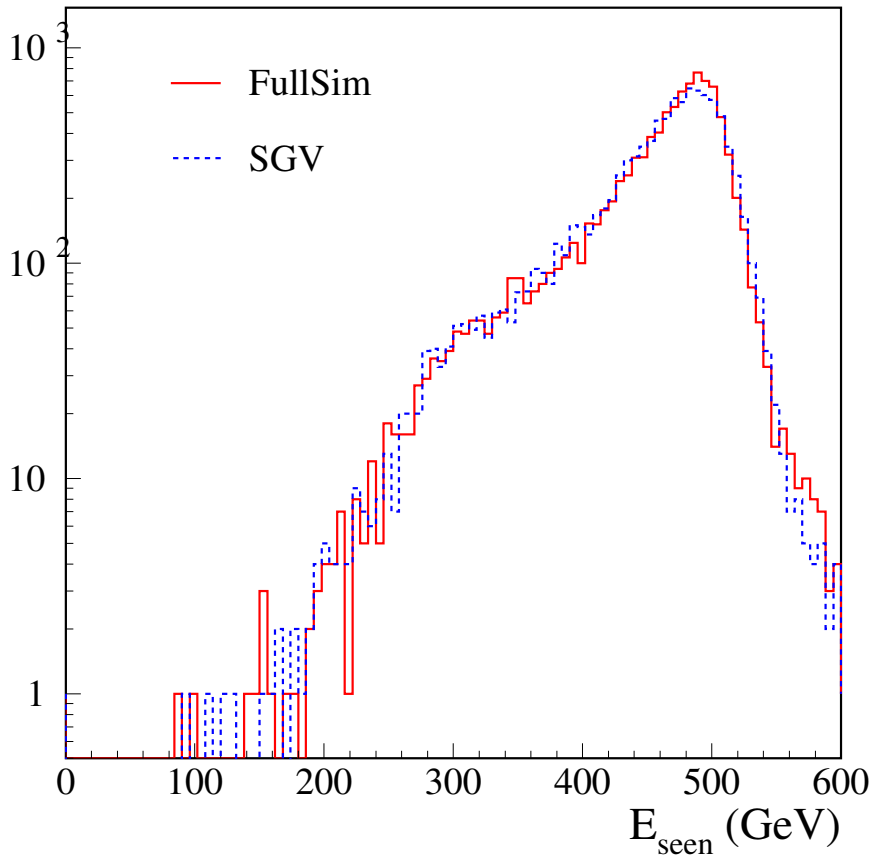
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag



Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag



Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag

Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

- Overall:
  - Total seen energy
- $e^+e^- \to ZZ \to$ four jets:

"True jets"

- Find initial hadrons from each colour-singlet (string).
- Find the quarks at each end of the string.
- Group hadrons to jets by which quark they are closest to.
- Follow the decay-chains and assign all particles in the event to the jet of their respective ancestor hadron.

  - Visible E
  - Higgs Mass
  - b-tag

Checking the parametrisation

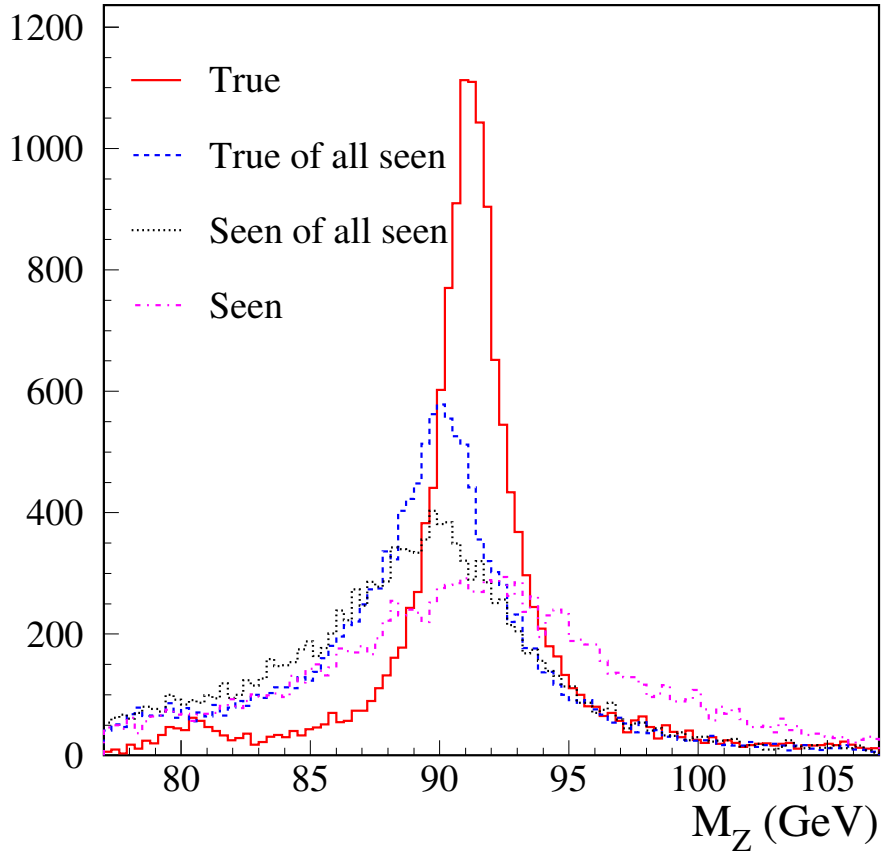
Feed exactly the same physics events through FullSim or SGV.

- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag

Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

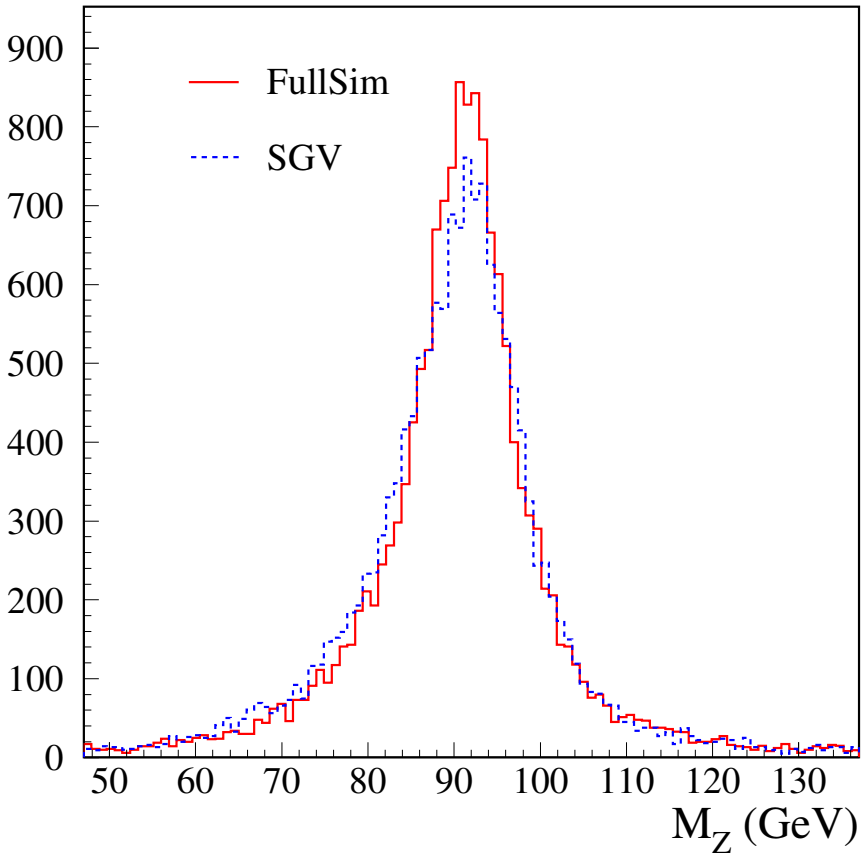
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag



Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

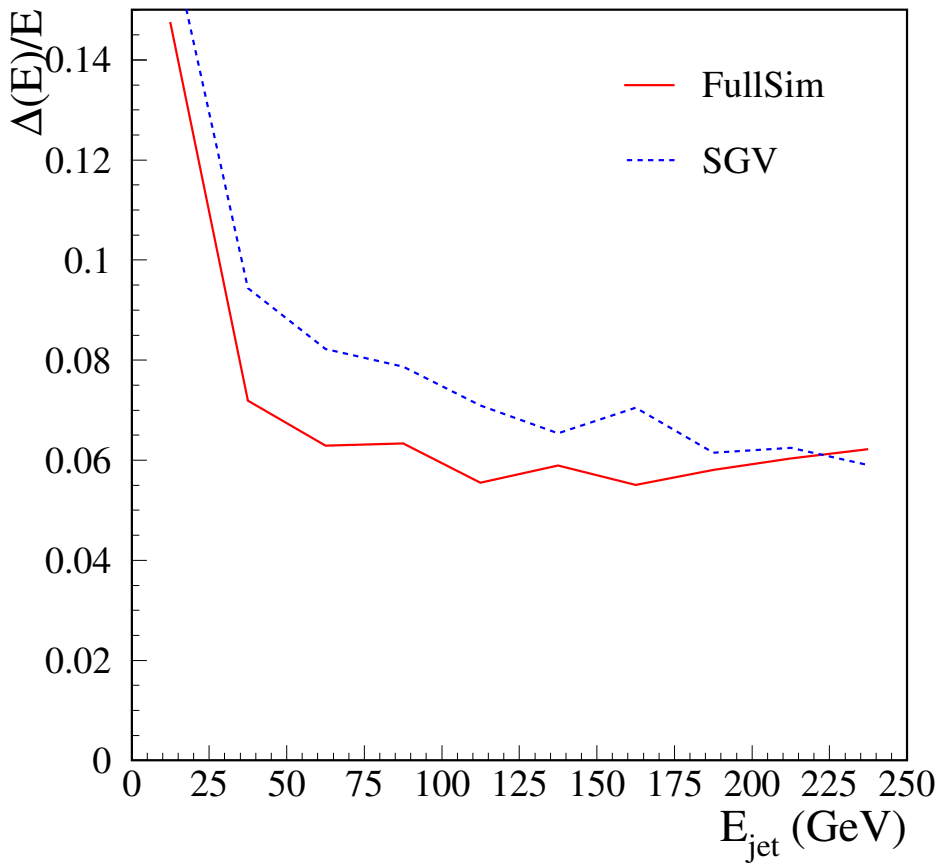
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag



Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

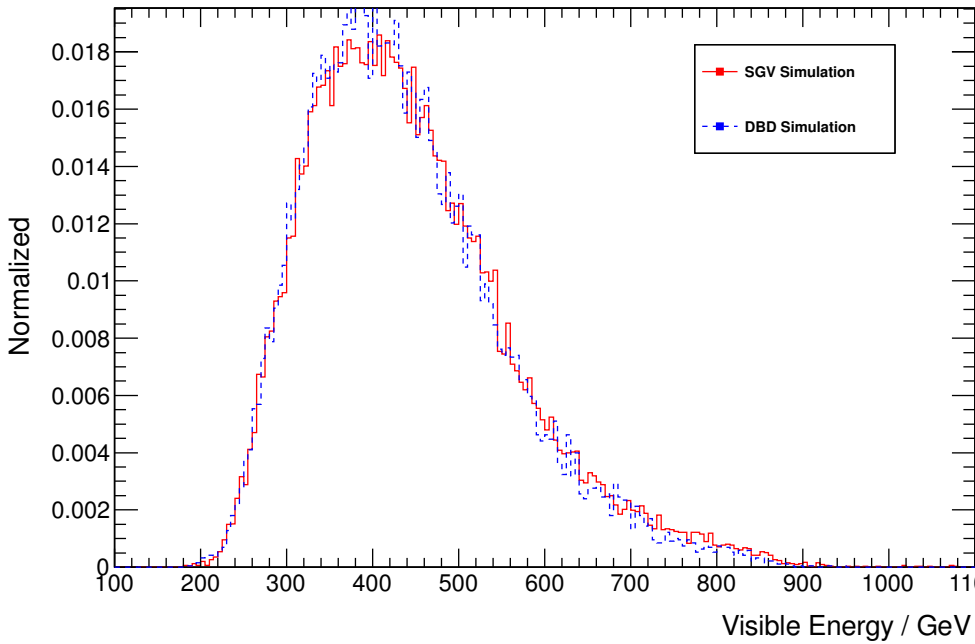
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag



Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

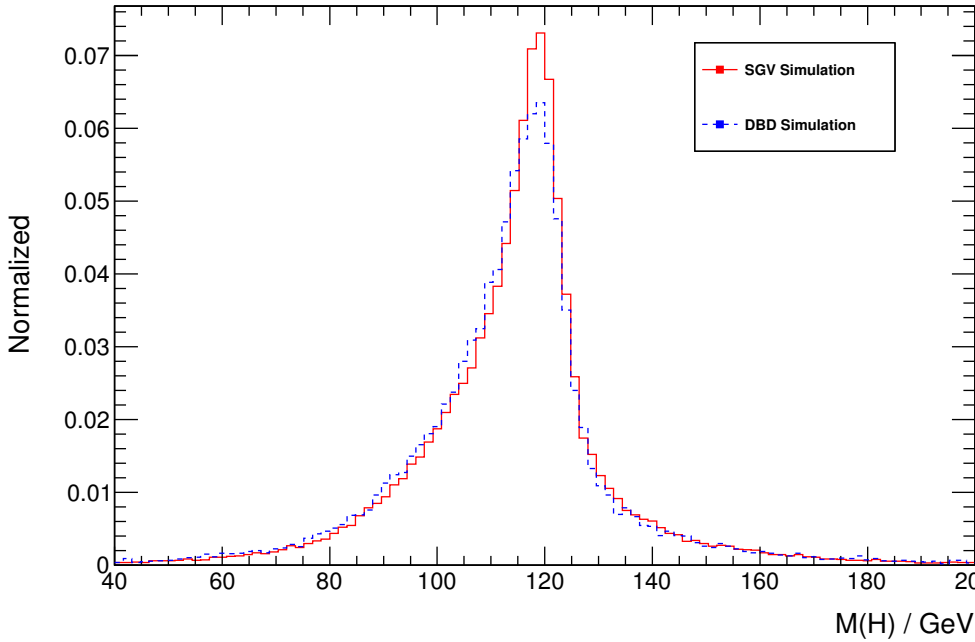
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag



Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

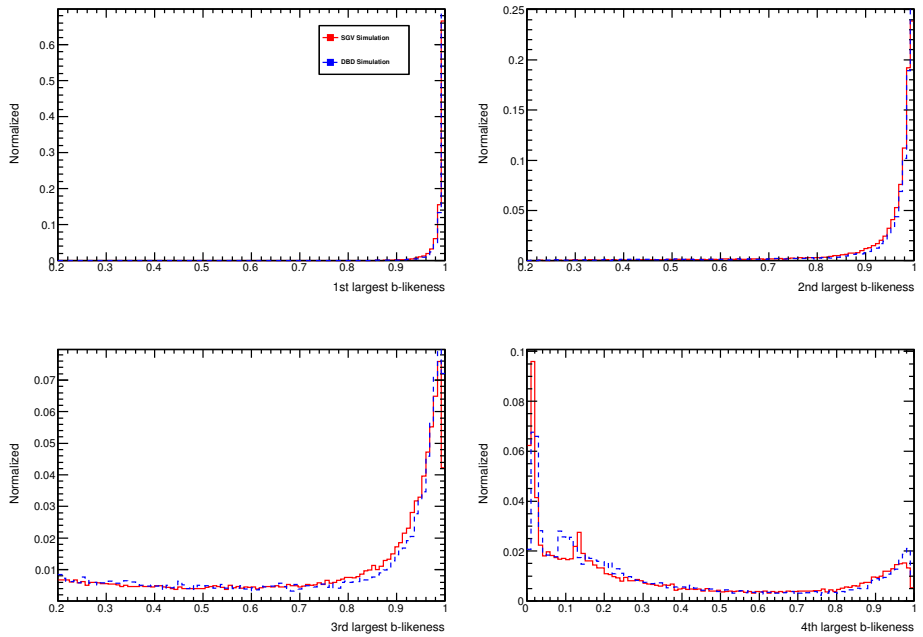
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag



Checking the parametrisation

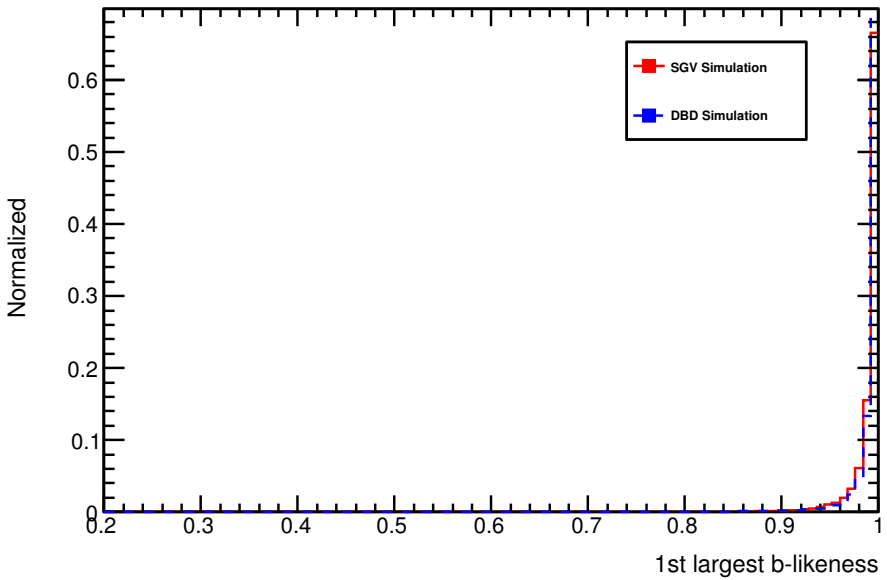
Feed exactly the same physics events through FullSim or SGV.

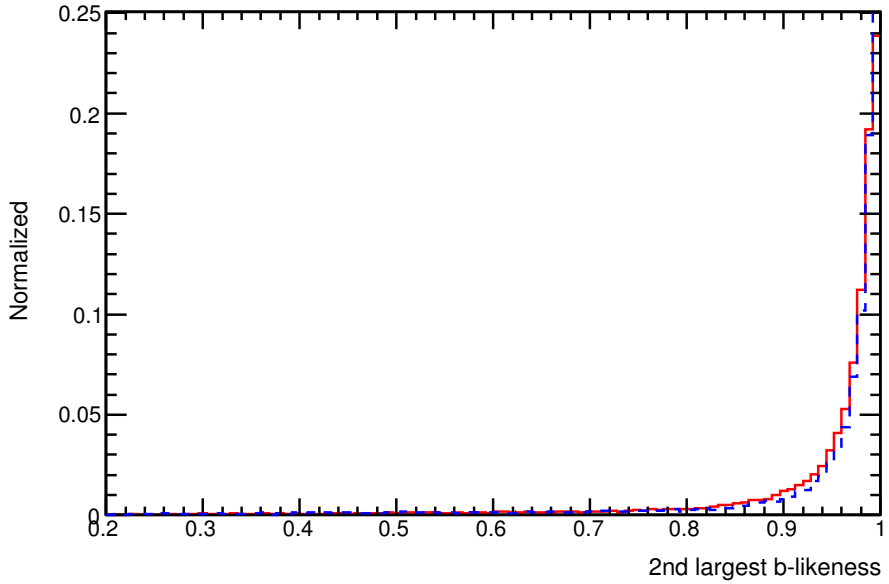
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag

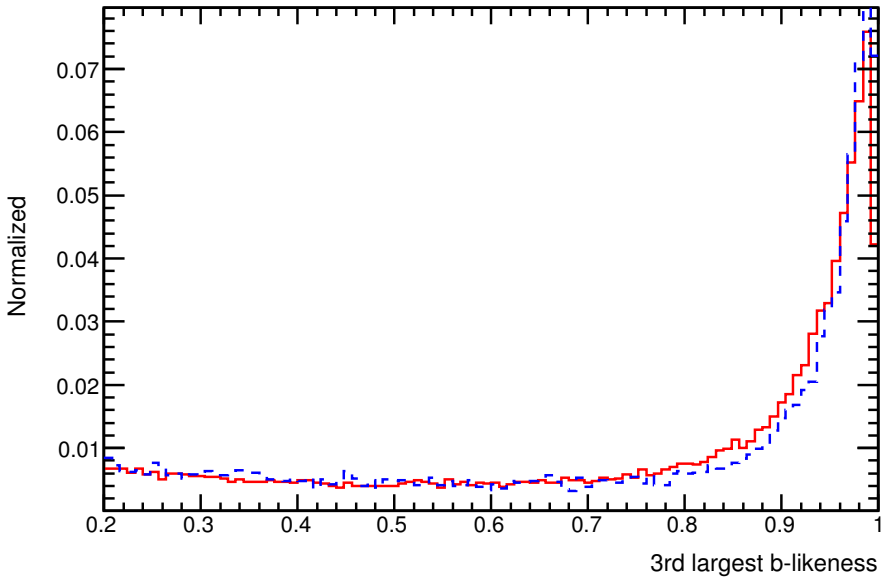


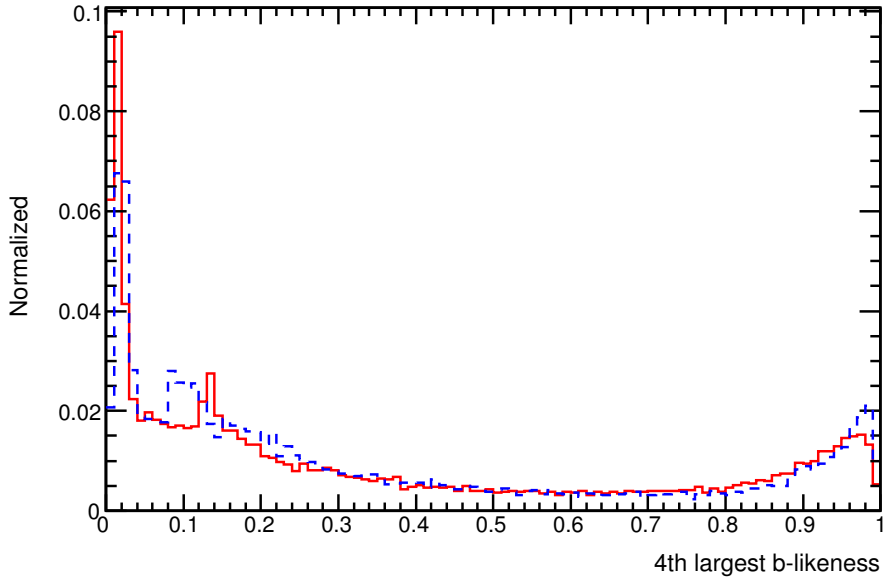
C

F


SGV Simulation
DBD Simulation
Normalized
1st largest b-likeness


Normalized
2nd largest b-likeness

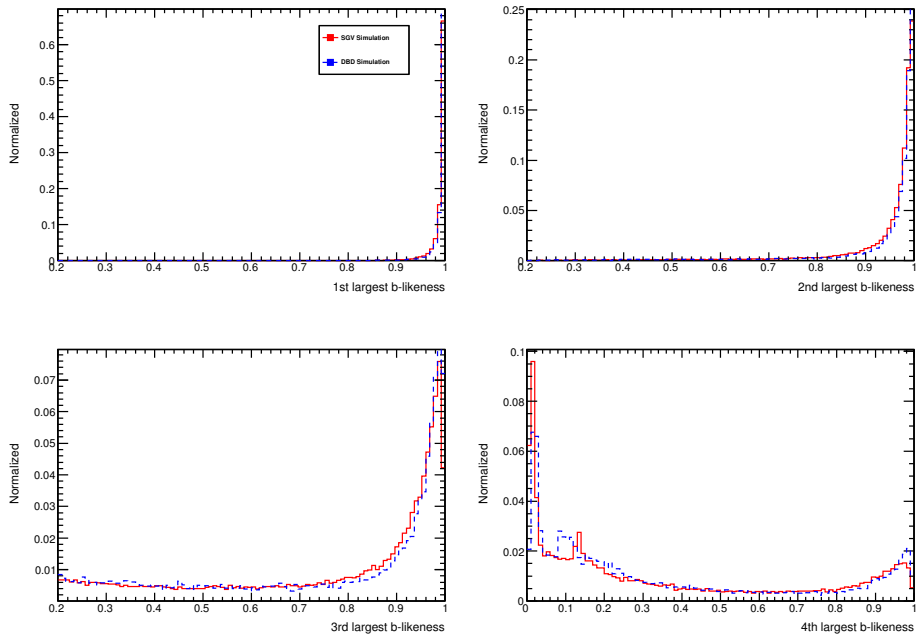

Normalized
3rd largest b-likeness


Normalized
4th largest b-likeness

Checking the parametrisation

Feed exactly the same physics events through FullSim or SGV.

- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag



LCIO DST mass-production

SGV has been used to produce ILD LCIO DST:s for the full DBD benchmarks- several times.

- 43 Mevents.
- $\sim$ 1 hour of wall-clock time (first submit to last completed) on the German NAF.
- NEW: Also added $\gamma\gamma \to f\bar{f}$ at 500 GeV. Another 167 Mevents (0.5 TB of DSTs ...). Took 30 minutes on the NAF.
- On the grid under:
  - lfn:/grid/ilc/users/berggren/mc-dbd/sgv-dst_y/zzz/xxx
  - (xxx= 2f, 4f, ... , zzz= 1000-B1b_ws, 500-TDR_ws, ... (y is 6 right now. Always use the latest !)

LCIO DST mass-production

SGV has been used to produce ILD LCIO DST:s for the full DBD benchmarks- several times.

- 43 Mevents.
- $\sim$ 1 hour of wall-clock time (first submit to last completed) on the German NAF.
- NEW: Also added $\gamma\gamma \to f\bar{f}$ at 500 GeV. Another 167 Mevents (0.5 TB of DSTs ...). Took 30 minutes on the NAF.
- On the grid under:
  - lfn:/grid/ilc/users/berggren/mc-dbd/sgv-dst_y/zzz/xxx
  - (xxx= 2f, 4f, ... , zzz= 1000-B1b_ws, 500-TDR_ws, ... (y is 6 right now. Always use the latest !)

LCIO DST mass-production

SGV has been used to produce ILD LCIO DST:s for the full DBD benchmarks- several times.

- 43 Mevents.
- $\sim$ 1 hour of wall-clock time (first submit to last completed) on the German NAF.
- NEW: Also added $\gamma\gamma \to f\bar{f}$ at 500 GeV. Another 167 Mevents (0.5 TB of DSTs ...). Took 30 minutes on the NAF.
- On the grid under:
  - lfn:/grid/ilc/users/berggren/mc-dbd/sgv-dst_y/zzz/xxx
  - (xxx= 2f, 4f, ... , zzz= 1000-B1b_ws, 500-TDR_ws, ... (y is 6 right now. Always use the latest !)

Installing SGV

Do

svn co https://svnsrv.desy.de/public/sgv/trunk/ sgv/

Then

cd sgv ; . ./install

This will take you about 30 seconds ...

- Study README do get the first test job done (another 30 seconds)
- Look README in the samples sub-directory, to enhance the capabilities, eg.:
  - Get STDHEP installed.
  - Get CERNLIB installed in native 64bit.
  - Get Whizard (basic or ILC-tuned) installed.
  - Activate PFlow parametrisation.
  - Get the LCIO-DST writer set up
  - ...

Summary

- The main features of the SGV FastSim program was presented.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) was explained.
- Comparisons to FullSim (Mokka/Marlin) was shown to be quite good.
- Recently added features:
  - Beam-spectrum for PYTHIA.
  - Correct BeamCal treatment (including the complicated geometry).
  - (Even) better PFlow parametrisation (not yet in SVN).
- SGV mass production works
  - Is done in $\mathcal{O}(1)$ hour. Recently added 167 Mevents of $\gamma\gamma \to f\bar{f}$ at 500 GeV
- More info: My slides from the Zeuthen FastSim workshop "Particle Flow ILC"

Summary

- The main features of the SGV FastSim program was presented.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) was explained.
- Comparisons to FullSim (Mokka/Marlin) was shown to be quite good.
- Recently added features:
  - Beam-spectrum for PYTHIA.
  - Correct BeamCal treatment (including the complicated geometry).
  - (Even) better PFlow parametrisation (not yet in SVN).
- SGV mass production works
  - Is done in $\mathcal{O}(1)$ hour. Recently added 167 Mevents of $\gamma\gamma \to f\bar{f}$ at 500 GeV
- More info: My slides from the Zeuthen FastSim workshop "Particle Flow ILC"

Summary

- The main features of the SGV FastSim program was presented.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) was explained.
- Comparisons to FullSim (Mokka/Marlin) was shown to be quite good.
- Recently added features:
  - Beam-spectrum for PYTHIA.
  - Correct BeamCal treatment (including the complicated geometry).
  - (Even) better PFlow parametrisation (not yet in SVN).
- SGV mass production works
  - Is done in $O(1)$ hour. Recently added 167 Mevents of $\gamma\gamma \to f\bar{f}$ at 500 GeV
- More info: My slides from the Zeuthen FastSim workshop "Particle Flow ILC"

Summary

- The main features of the SGV FastSim program was presented.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) was explained.
- Comparisons to FullSim (Mokka/Marlin) was shown to be quite good.
- Recently added features:
  - Beam-spectrum for PYTHIA.
  - Correct BeamCal treatment (including the complicated geometry).
  - (Even) better PFlow parametrisation (not yet in SVN).
- SGV mass production works
  - Is done in $\mathcal{O}(1)$ hour. Recently added 167 Mevents of $\gamma\gamma \to f\bar{f}$ at 500 GeV
- More info: My slides from the Zeuthen FastSim workshop "Particle Flow ILC"

Summary

- The main features of the SGV FastSim program was presented.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) was explained.
- Comparisons to FullSim (Mokka/Marlin) was shown to be quite good.
- Recently added features:
  - Beam-spectrum for PYTHIA.
  - Correct BeamCal treatment (including the complicated geometry).
  - (Even) better PFlow parametrisation (not yet in SVN).
- SGV mass production works
  - Is done in $\mathcal{O}(1)$ hour. Recently added 167 Mevents of $\gamma\gamma \to f\bar{f}$ at 500 GeV
- More info: My slides from the Zeuthen FastSim workshop "Particle Flow ILC"

Summary

- The main features of the SGV FastSim program was presented.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) was explained.
- Comparisons to FullSim (Mokka/Marlin) was shown to be quite good.
- Recently added features:
  - Beam-spectrum for PYTHIA.
  - Correct BeamCal treatment (including the complicated geometry).
  - (Even) better PFlow parametrisation (not yet in SVN).
- SGV mass production works
  - Is done in $\mathcal{O}(1)$ hour. Recently added 167 Mevents of $\gamma\gamma \to f\bar{f}$ at 500 GeV
- More info: My slides from the Zeuthen FastSim workshop "Particle Flow ILC"

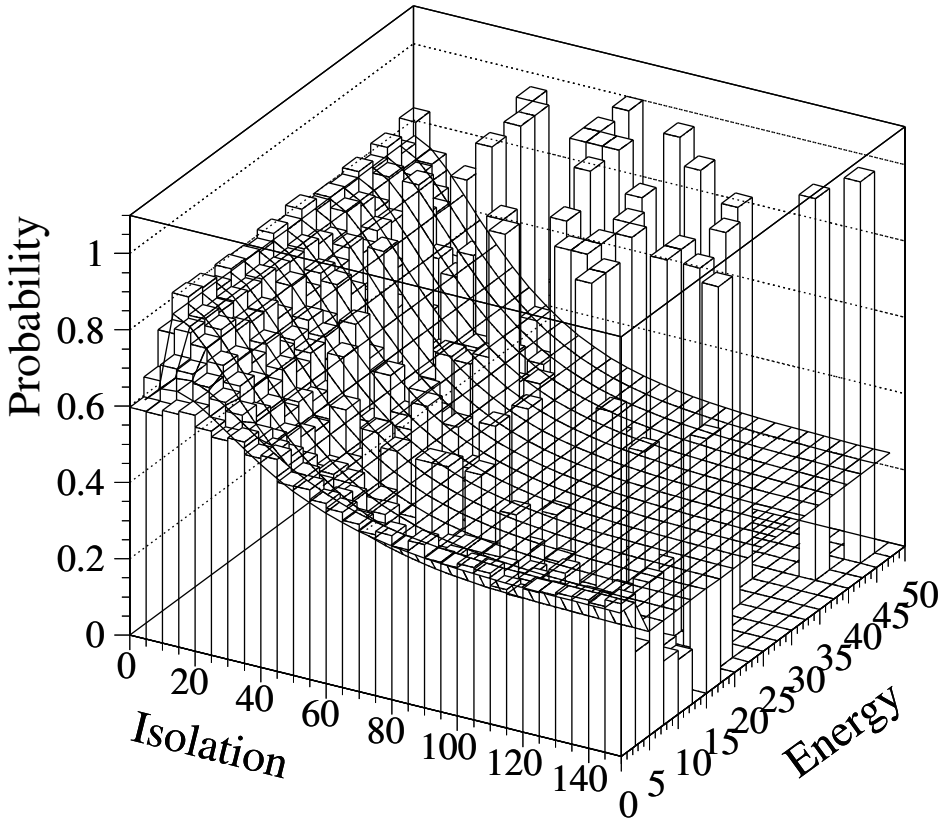
Thank You !

Backup

BACKUP SLIDES

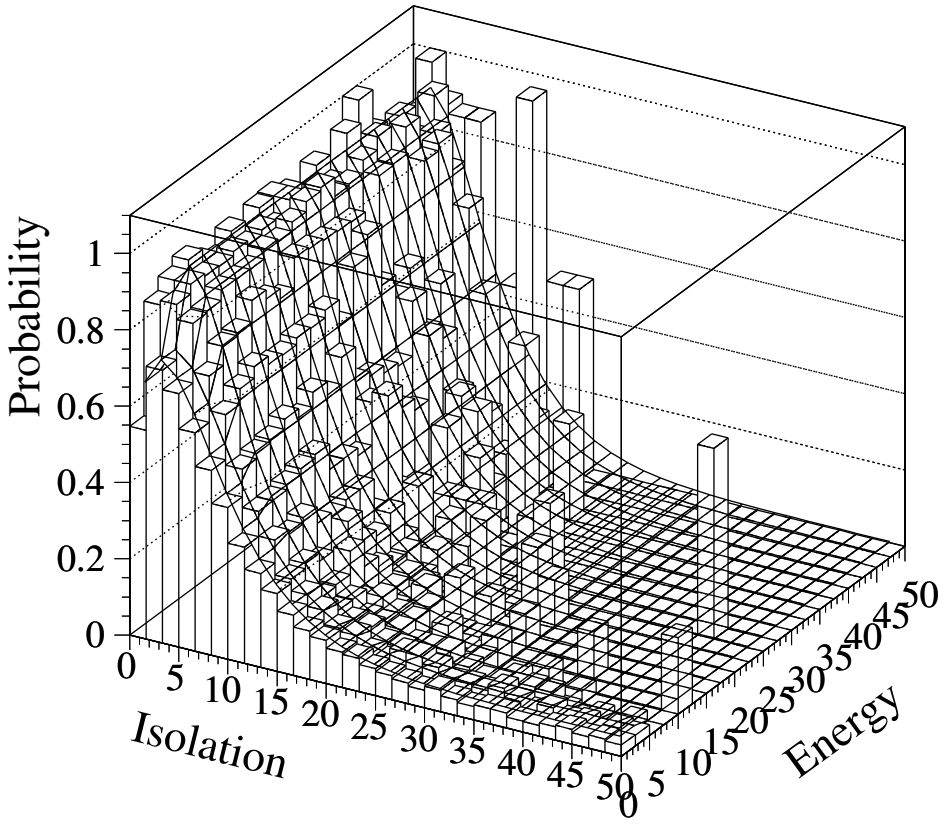
Observed distributions

- Probability to split (charged had or γ)
- Fraction the energy vs distance
- ... and vs E
- Fit of the Distribution of the fraction
- Average fraction vs. E and distance.



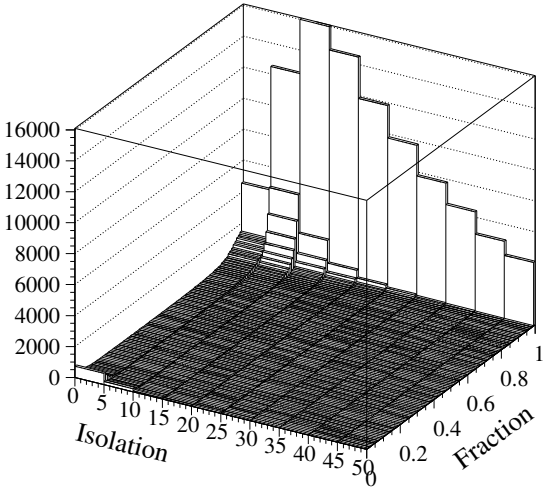
Observed distributions

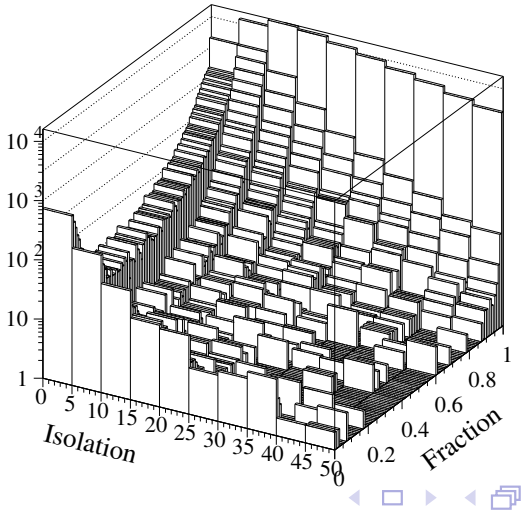
- Probability to split (charged had or γ)
- Fraction the energy vs distance
- ... and vs E
- Fit of the Distribution of the fraction
- Average fraction vs. E and distance.



Observed distributions

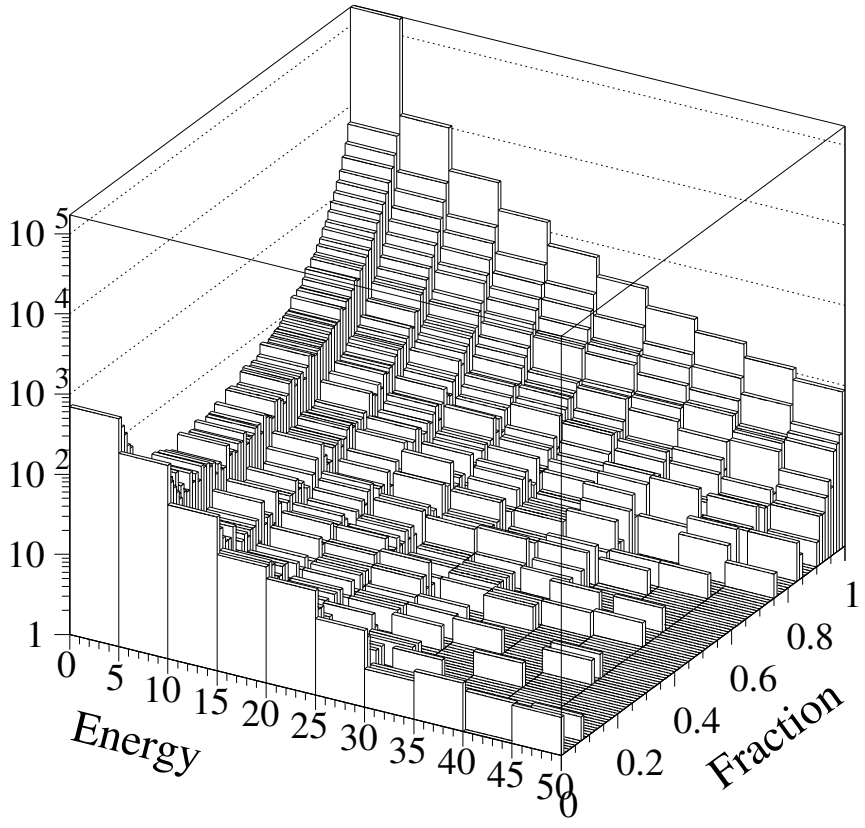
- Probability to split (charged had or γ)
- Fraction the energy vs distance
- ... and vs E
- Fit of the Distribution of the fraction
- Average fraction vs. E and distance.





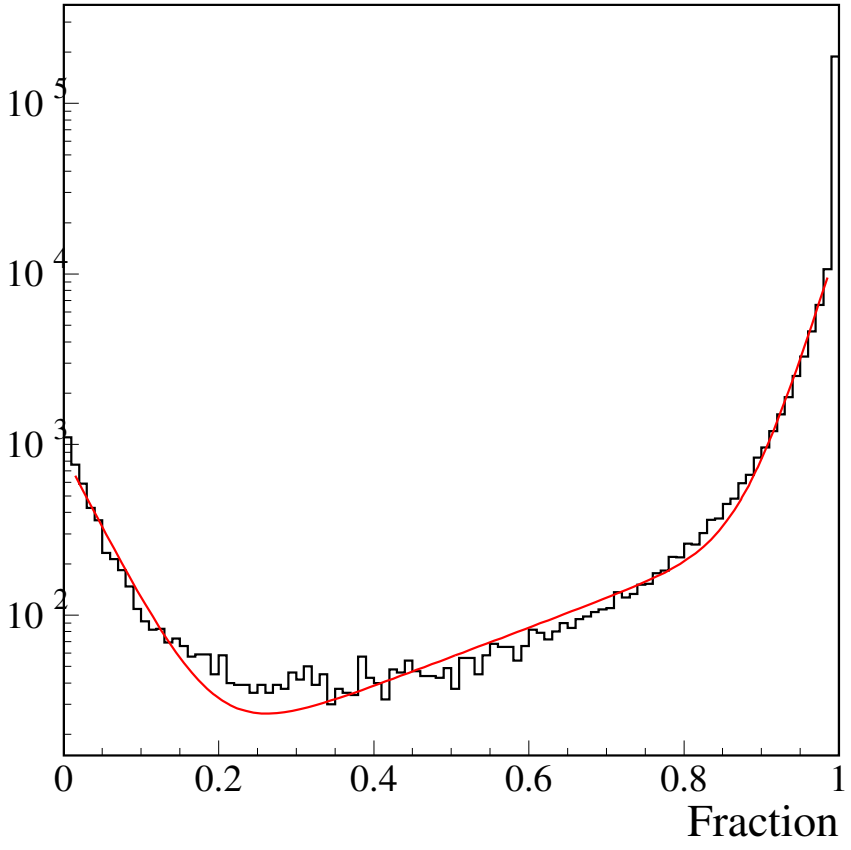
Observed distributions

- Probability to split (charged had or γ)
- Fraction the energy vs distance
- ... and vs E
- Fit of the Distribution of the fraction
- Average fraction vs. E and distance.



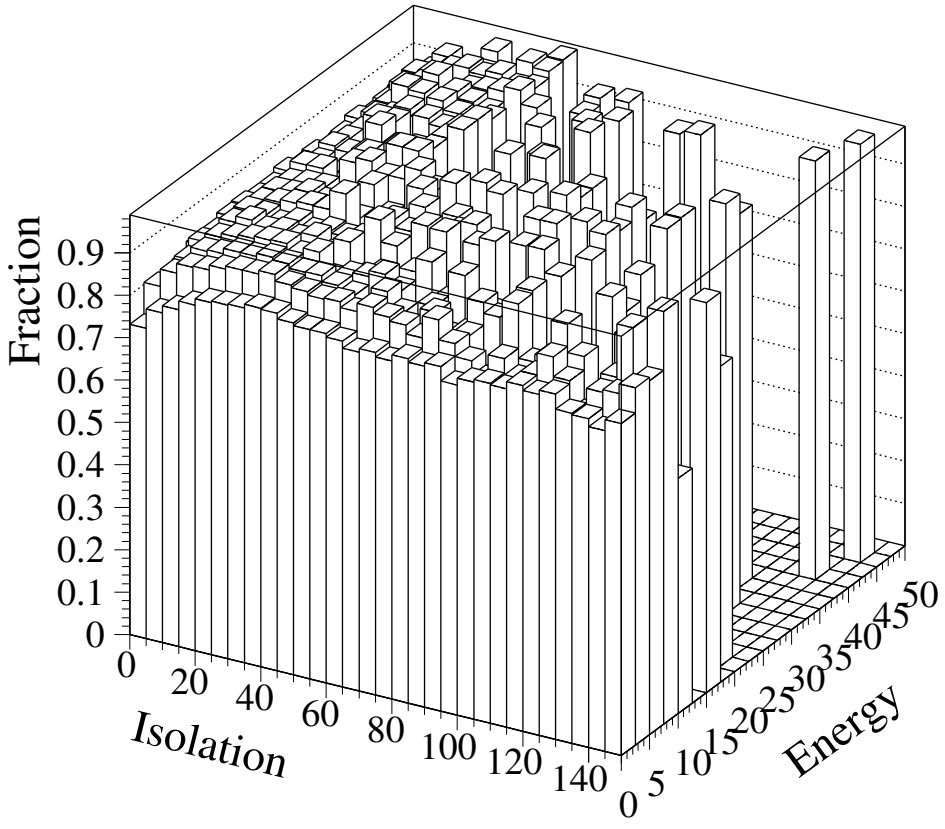
Observed distributions

- Probability to split (charged had or γ)
- Fraction the energy vs distance
- ... and vs E
- Fit of the Distribution of the fraction
- Average fraction vs. E and distance.



Observed distributions

- Probability to split (charged had or γ)
- Fraction the energy vs distance
- ... and vs E
- Fit of the Distribution of the fraction
- Average fraction vs. E and distance.



$\gamma\gamma$ background

Total cross-section for $e^+e^- \to \gamma\gamma e^+e^- \to q\bar{q}e^+e^-$: 35 nb (PYTHIA)

- $\int \mathcal{L}dt = 500$ fb^{-1} $\to$ 18 $\star 10^9$ 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.

$\gamma\gamma$ background

Total cross-section for $e^+e^- \to \gamma\gamma e^+e^- \to q\bar{q}e^+e^-$: 35 nb (PYTHIA)

- $\int \mathcal{L}dt = 500$ fb^{-1} $\to$ 18 $\star 10^9$ 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.

$\gamma\gamma$ background

Total cross-section for $e^+e^- \to \gamma\gamma e^+e^- \to q\bar{q}e^+e^-$: 35 nb (PYTHIA)

- $\int \mathcal{L}dt = 500$ fb^{-1} $\to$ 18 $\star 10^9$ 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

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)

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

Calorimeter simulation: SGV strategy

- Concentrate on what really matters:
  - True charged particles splitting off (a part of) their shower: double-counting.
  - True neutral particles merging (a part of) their shower with charged particles: enetgy loss.
- Don't care about neutral-neutral or charged-charged merging.
- Nor about multiple splitting/merging.
- Then: identify the most relevant variables available in fast simulation:
  - Cluster energy.
  - Distance to nearest particle of "the other type".
  - EM or hadron.
  - Barrel or end-cap.

Calorimeter simulation: SGV strategy

- Concentrate on what really matters:
  - True charged particles splitting off (a part of) their shower: double-counting.
  - True neutral particles merging (a part of) their shower with charged particles: enetgy loss.
- Don't care about neutral-neutral or charged-charged merging.
- Nor about multiple splitting/merging.
- Then: identify the most relevant variables available in fast simulation:
  - Cluster energy.
  - Distance to nearest particle of "the other type".
  - EM or hadron.
  - Barrel or end-cap.

Calorimeter simulation: SGV strategy

- Concentrate on what really matters:
  - True charged particles splitting off (a part of) their shower: double-counting.
  - True neutral particles merging (a part of) their shower with charged particles: enetgy loss.
- Don't care about neutral-neutral or charged-charged merging.
- Nor about multiple splitting/merging.
- Then: identify the most relevant variables available in fast simulation:
  - Cluster energy.
  - Distance to nearest particle of "the other type"
  - EM or hadron.
  - Barrel or end-cap.

Collections

- Added sensible values to all collections that will (probably) be there on the DST from the fullSim production.
  - BuildUpVertex
  - BuildUpVertex_RP
  - MarlinTrkTracks
  - PandoraClusters
  - PandoraPFOs
  - PrimaryVertex
  - RecoMCTruthLink
  - MCParticlesSkimmed
  - V0Vertices
  - V0RecoParticles
  - BCALParticles
  - BCALClusters
  - BCALMCTruthLink
  - PrimaryVertex_RP
- Also added more relation links:
  - MCTruthRecoLink
  - ClusterMCTruthLink
  - MCTruthClusterLink
  - MCTruthTrackLink
  - TrackMCTruthLink
  - MCTruthBcalLink

Comments

Secondary vertices (as before):

- Use true information to find all secondary vertices.
- For all vertices with ≥ 2 seen charged tracks: do vertex fit.
- Concequence:
  - Vertex *finding* is too good.
  - Vertex *quality* should be comparable to FullSim.

In addition: Decide from parent pdg-code if it goes into BuildUpVertex or V0Vertices !
MCParticle :

- There might be some issues with history codes in the earlier part of the event (initial beam-particles, 94-objects, ...)

Comments

Clusters:

- Are done with the Pandora confusion parametrisation on.
- Expect $\sim$ correct dispersion of jet energy, but a few % to high central value.
- See my talk three weeks ago.
- Warning: Clusters are always only in one detector , so don't use E_{had}/E_{EM} for e/π: It will be $\equiv$ 100 % efficient !

Navigators

- All the navigators that the TruthLinker processor makes when all flags are switched on are created:
  - Both Seen to True and True to Seen (weights are different !)
  - Seen is both PFOs, tracks and clusters.
  - The standard RecoMCTruthLink collection is as it would be from FullSim ie. weights between 0 and 1.

Outlook

- 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.
  - 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.

Outlook

- 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.
  - 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.

Outlook

- 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.
  - 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.

Outlook

- 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.
  - 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.

Outlook

- 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.
  - 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.

Outlook

- 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.
  - 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.

Outlook

- 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.
  - 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.