Fast detector simulation at the ILC: Simulation á Grande Vitesse - SGV

Mikael Berggren[1]

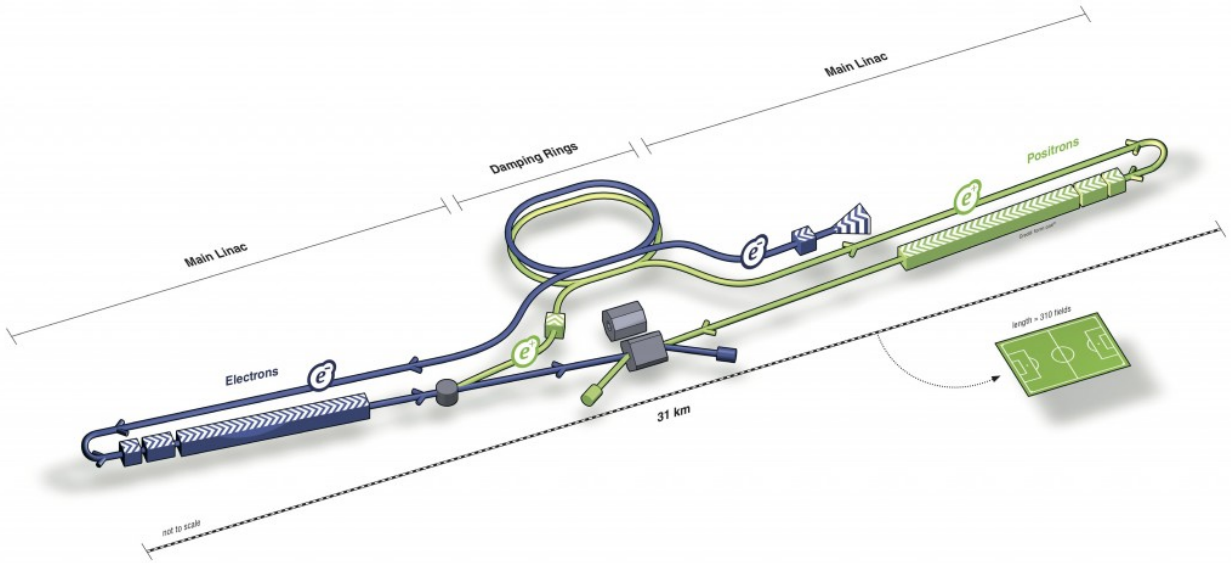
[1]DESY, Hamburg

FastSim workshop, DESY-Zeuthen, Jan 2014

Outline

1. ILC
2. Fast simulation
   - The need for fast simulation at ILC
   - Ex1: $\gamma\gamma$ cross-sections
   - Ex2: SUSY scans
   - Fast simulation types
3. SGV
   - Working priciples
   - Technicalities
   - Usage
   - Use-cases, Comparisions
     - SGV for detector design
     - SGV and FullSim
4. Outlook and Summary

The ILC



- A linear e^+e^- collider.
- E_{CMS} tunable between 200 and 500 GeV, upgradable to 1 TeV.
- Total length 31 km
- $\int \mathcal{L} \sim 500$ fb^{-1} in 4 years
- Polarisation e$^-$: 80% (e$^+$: $\geq$ 30%)
- 2 experiments, but (possibly) only one interaction region.
- Concurrent running with the LHC

The ILC is not LHC

- Lepton-collider: Initial state is known.
- Production is EW $\Rightarrow$
  - Small theoretical uncertainties.
  - No "underlaying event".
  - Low cross-sections wrt. LHC, also for background.
  - Trigger-less operation.
  - High precision (sub-%) measurements needed, to extend our knowledge beyond LEP, Tevatron, LHC.
  - Interesting physics at low angles: t-channel di-boson production ...
- Extremely small beam-spot: 5 nm $\times$ 100 nm $\times$ 150 μm.
- High luminosity: 2×10^{34} cm^{-2} s^{-1}. Single pass operation $\Rightarrow$ this is the lumi for every bunch-crossing.

The ILC is not LHC

- Lepton-collider: Initial state is known.
- Production is EW $\Rightarrow$
  - Small theoretical uncertainties.
  - No "underlaying event".
  - Low cross-sections wrt. LHC, also for background.
  - Trigger-less operation.
  - High precision (sub-%) measurements needed, to extend our knowledge beyond LEP, Tevatron, LHC.
  - Interesting physics at low angles: t-channel di-boson production ...
- Extremely small beam-spot: 5 nm $\times$ 100 nm $\times$ 150 μm.
- High luminosity: 2×10^{34} cm^{-2} s^{-1}. Single pass operation $\Rightarrow$ this is the lumi for every bunch-crossing.

The ILC is not LHC

- Lepton-collider: Initial state is known.
- Production is EW $\Rightarrow$
  - Small theoretical uncertainties.
  - No "underlaying event".
  - Low cross-sections wrt. LHC, also for background.
  - Trigger-less operation.
  - High precision (sub-%) measurements needed, to extend our knowledge beyond LEP, Tevatron, LHC.
  - Interesting physics at low angles: t-channel di-boson production ...
- Extremely small beam-spot: 5 nm $\times$ 100 nm $\times$ 150 μm.
- High luminosity: 2×10^{34} cm^{-2} s^{-1}. Single pass operation $\Rightarrow$ this is the lumi for every bunch-crossing.

The ILC is not LHC

- Lepton-collider: Initial state is known.
- Production is EW $\Rightarrow$
  - Small theoretical uncertainties.
  - No "underlaying event".
  - Low cross-sections wrt. LHC, also for background.
  - Trigger-less operation.
  - High precision (sub-%) measurements needed, to extend our knowledge beyond LEP, Tevatron, LHC.
  - Interesting physics at low angles: t-channel di-boson production ...
- Extremely small beam-spot: 5 nm $\times$ 100 nm $\times$ 150 μm.
- High luminosity: 2×10^{34} cm^{-2} s^{-1}. Single pass operation $\Rightarrow$ this is the lumi for every bunch-crossing.

The ILC is not LHC

- Lepton-collider: Initial state is known.
- Production is EW $\Rightarrow$
  - Small theoretical uncertainties.
  - No "underlaying event".
  - Low cross-sections wrt. LHC, also for background.
  - Trigger-less operation.
  - High precision (sub-%) measurements needed, to extend our knowledge beyond LEP, Tevatron, LHC.
  - Interesting physics at low angles: t-channel di-boson production ...
- Extremely small beam-spot: 5 nm $\times$ 100 nm $\times$ 150 μm.
- High luminosity: 2×10^{34} cm^{-2} s^{-1}. Single pass operation $\Rightarrow$ this is the lumi for every bunch-crossing.

The ILC is not LEP, either

- Small beam-spot $\Rightarrow$ Beam-beam interactions $\Rightarrow$
  - Large amounts of synchrotron photons ...
  - ... that get Compton back-scattered ...
  - They might create e^+e^- pairs when interacting with the field: The pairs-background.
  - Or interact with each other: mini-jets
- Single pass operation, ondulator positron-source, beam-beam effects: Beam-spectrum is not a δ-function.
- Luminosity/bunch-crossing three orders of magnitude higher: pile-up of $\gamma\gamma$ events (a few/BX, yielding a few particles, so we're not talking LHC conditions here !)

The ILC is not LEP, either

- Small beam-spot $\Rightarrow$ Beam-beam interactions $\Rightarrow$
  - Large amounts of synchrotron photons ...
  - ... that get Compton back-scattered ...
  - They might create e^+e^- pairs when interacting with the field: The pairs-background.
  - Or interact with each other: mini-jets
- Single pass operation, ondulator positron-source, beam-beam effects: Beam-spectrum is not a δ-function.
- Luminosity/bunch-crossing three orders of magnitude higher: pile-up of $\gamma\gamma$ events (a few/BX, yielding a few particles, so we're not talking LHC conditions here !)

The ILC is not LEP, either

- Small beam-spot $\Rightarrow$ Beam-beam interactions $\Rightarrow$
  - Large amounts of synchrotron photons ...
  - ... that get Compton back-scattered ...
  - They might create e^+e^- pairs when interacting with the field: The pairs-background.
  - Or interact with each other: mini-jets
- Single pass operation, ondulator positron-source, beam-beam effects: Beam-spectrum is not a δ-function.
- Luminosity/bunch-crossing three orders of magnitude higher: pile-up of $\gamma\gamma$ events (a few/BX, yielding a few particles, so we're not talking LHC conditions here !)

The ILC is not LEP, either

- Small beam-spot $\Rightarrow$ Beam-beam interactions $\Rightarrow$
  - Large amounts of synchrotron photons ...
  - ... that get Compton back-scattered ...
  - They might create e^+e^- pairs when interacting with the field: The pairs-background.
  - Or interact with each other: mini-jets
- Single pass operation, ondulator positron-source, beam-beam effects: Beam-spectrum is not a δ-function.
- Luminosity/bunch-crossing three orders of magnitude higher: pile-up of $\gamma\gamma$ events (a few/BX, yielding a few particles, so we're not talking LHC conditions here !)

The ILC is not LEP, either

- Small beam-spot $\Rightarrow$ Beam-beam interactions $\Rightarrow$
  - Large amounts of synchrotron photons ...
  - ... that get Compton back-scattered ...
  - They might create e^+e^- pairs when interacting with the field: The pairs-background.
  - Or interact with each other: mini-jets
- Single pass operation, ondulator positron-source, beam-beam effects: Beam-spectrum is not a δ-function.
- Luminosity/bunch-crossing three orders of magnitude higher: pile-up of $\gamma\gamma$ events (a few/BX, yielding a few particles, so we're not talking LHC conditions here !)

The ILC is not LEP, either

- Small beam-spot $\Rightarrow$ Beam-beam interactions $\Rightarrow$
  - Large amounts of synchrotron photons ...
  - ... that get Compton back-scattered ...
  - They might create e^+e^- pairs when interacting with the field: The pairs-background.
  - Or interact with each other: mini-jets
- Single pass operation, ondulator positron-source, beam-beam effects: Beam-spectrum is not a δ-function.
- Luminosity/bunch-crossing three orders of magnitude higher: pile-up of $\gamma\gamma$ events (a few/BX, yielding a few particles, so we're not talking LHC conditions here !)

The ILC is not LEP, either

- Small beam-spot $\Rightarrow$ Beam-beam interactions $\Rightarrow$
  - Large amounts of synchrotron photons ...
  - ... that get Compton back-scattered ...
  - They might create e^+e^- pairs when interacting with the field: The pairs-background.
  - Or interact with each other: mini-jets
- Single pass operation, ondulator positron-source, beam-beam effects: Beam-spectrum is not a δ-function.
- Luminosity/bunch-crossing three orders of magnitude higher: pile-up of $\gamma\gamma$ events (a few/BX, yielding a few particles, so we're not talking LHC conditions here !)

The ILC : Detectors

- Low background $\Rightarrow$ detectors can be:
  - Thin : few % X_0 in front of calorimeters
  - Very close to IP: first layer of VXD at 1.5 cm.
  - Close to 4π: holes for beam-pipe only few cm = 0.2 msr un-covered = Area of Suisse Romande (or Schleswig-Holstein) relative to earth.
- High precision measurements:
  - Extremely high demands on tracking.
  - Tracking to low angles
  - Identify and measure every particle in the event = Particle-flow:
    - Measure charged particles with tracker, not ECAL, which is second best.
    - Need to separate neutral hadrons from charged in calorimeters.
    - Separate showers in calorimeters $\Rightarrow$ high granularity.

The ILC : Detectors

- Low background $\Rightarrow$ detectors can be:
  - Thin : few % X_0 in front of calorimeters
  - Very close to IP: first layer of VXD at 1.5 cm.
  - Close to 4π: holes for beam-pipe only few cm = 0.2 msr un-covered = Area of Suisse Romande (or Schleswig-Holstein) relative to earth.
- High precision measurements:
  - Extremely high demands on tracking.
  - Tracking to low angles
  - Identify and measure every particle in the event = Particle-flow:
    - Measure charged particles with tracker and leave unassociated
    - Need to separate neutral hadrons from charged in calorimeters
    - Separate showers in calorimeters $\Rightarrow$ high granularity

The ILC : Detectors

- Low background $\Rightarrow$ detectors can be:
  - Thin : few % X_0 in front of calorimeters
  - Very close to IP: first layer of VXD at 1.5 cm.
  - Close to 4π: holes for beam-pipe only few cm = 0.2 msr un-covered = Area of Suisse Romande (or Schleswig-Holstein) relative to earth.
- High precision measurements:
  - Extremely high demands on tracking.
  - Tracking to low angles
  - Identify and measure every particle in the event = Particle-flow:
    - Measure charged particles with tracker, neutrals with calorimeters.
    - Need to separate neutral clusters from charged in calorimeters.
    - Separate showers in calorimeters $\Rightarrow$ high granularity.

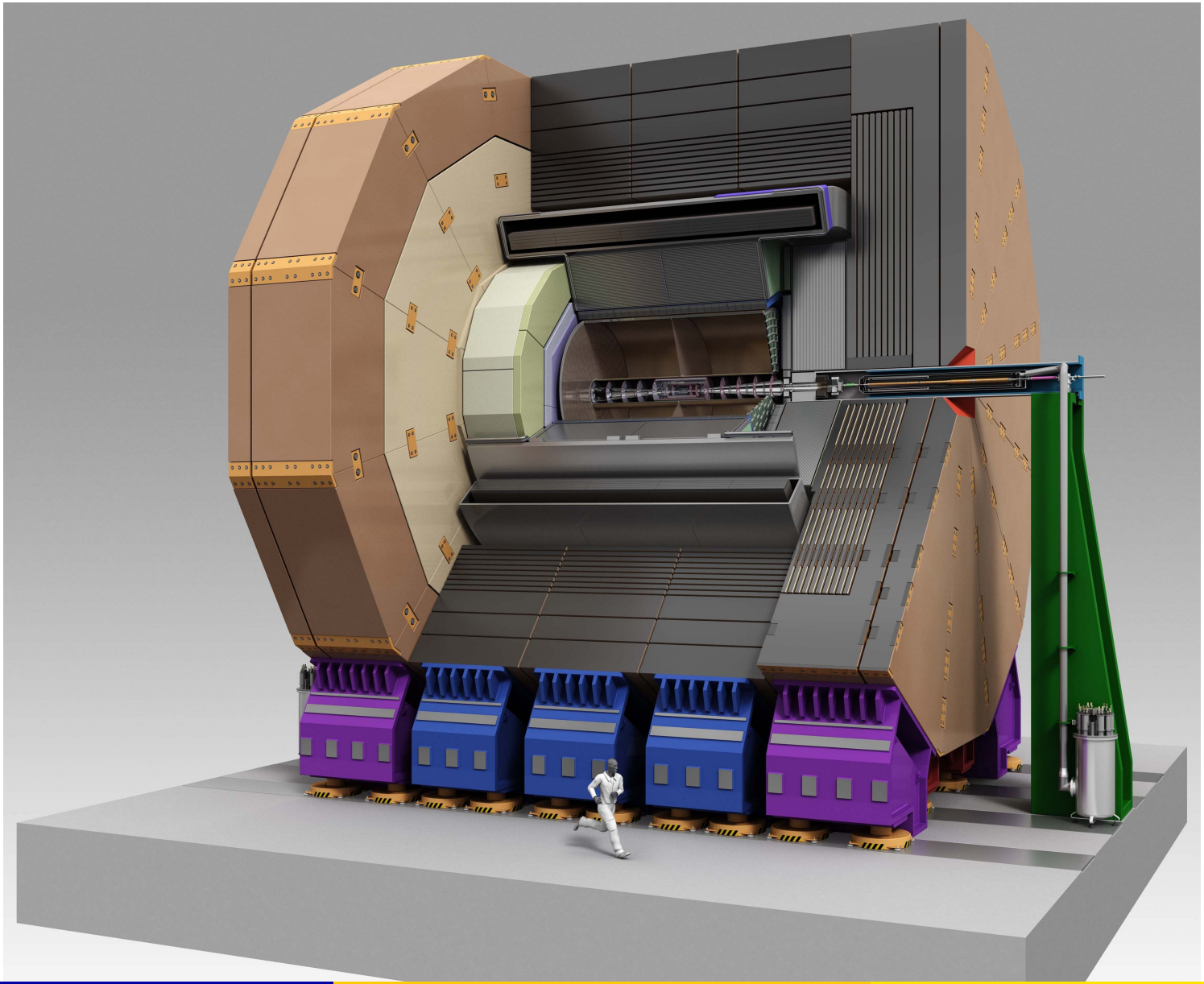
The ILC : Detectors

- Low background $\Rightarrow$ detectors can be:
  - Thin : few % X_0 in front of calorimeters
  - Very close to IP: first layer of VXD at 1.5 cm.
  - Close to 4π: holes for beam-pipe only few cm = 0.2 msr un-covered = Area of Suisse Romande (or Schleswig-Holstein) relative to earth.
- High precision measurements:
  - Extremely high demands on tracking.
  - Tracking to low angles
  - Identify and measure every particle in the event = Particle-flow:
    - Measure charged particles with tracker, neutrals with calorimeters.
    - Need to separate neutral clusters from charged in calorimeters.
    - Separate showers in calorimeters $\Rightarrow$ high granularity.

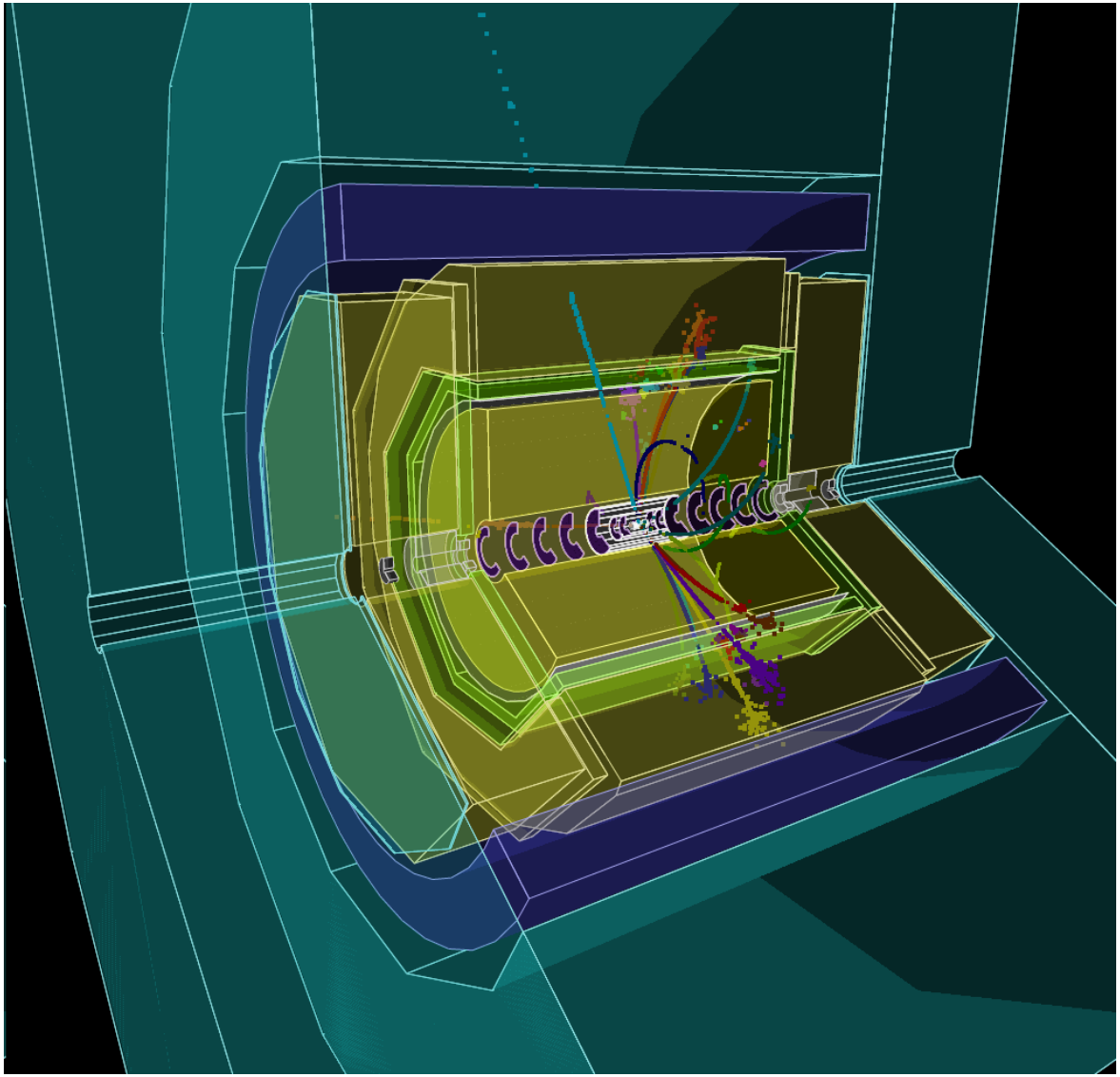
The ILC : Detectors

- Low background $\Rightarrow$ detectors can be:
  - Thin : few % X_0 in front of calorimeters
  - Very close to IP: first layer of VXD at 1.5 cm.
  - Close to 4π: holes for beam-pipe only few cm = 0.2 msr un-covered = Area of Suisse Romande (or Schleswig-Holstein) relative to earth.
- High precision measurements:
  - Extremely high demands on tracking.
  - Tracking to low angles
  - Identify and measure every particle in the event = Particle-flow:
    - Measure charged particles with tracker, neutrals with calorimeters.
    - Need to separate neutral clusters from charged in calorimeters.
    - Separate showers in calorimeters $\Rightarrow$ high granularity.

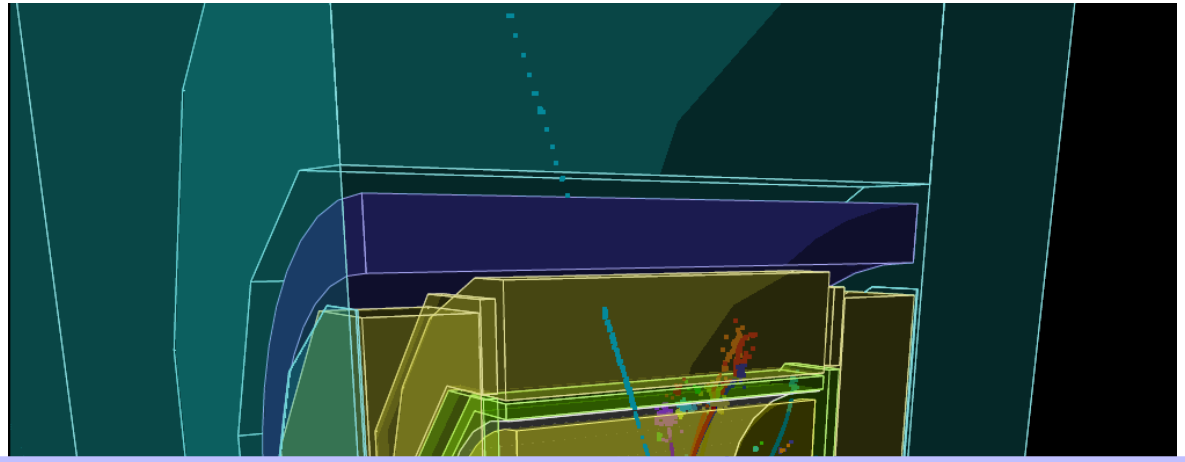
The ILD Detector



Example: $Zh \to \mu\mu q\bar{q}$ at 250 GeV



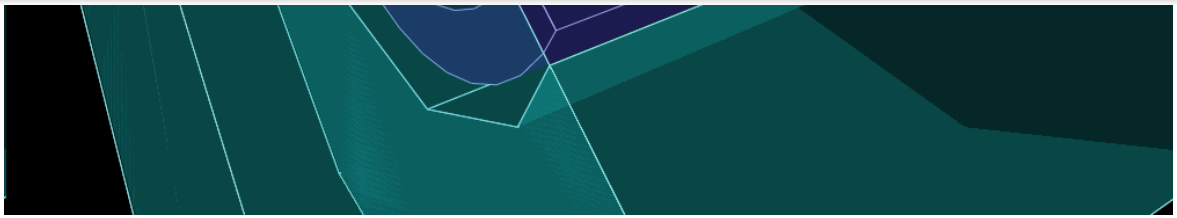
Example: $Zh \to \mu\mu q\bar{q}$ at 250 GeV



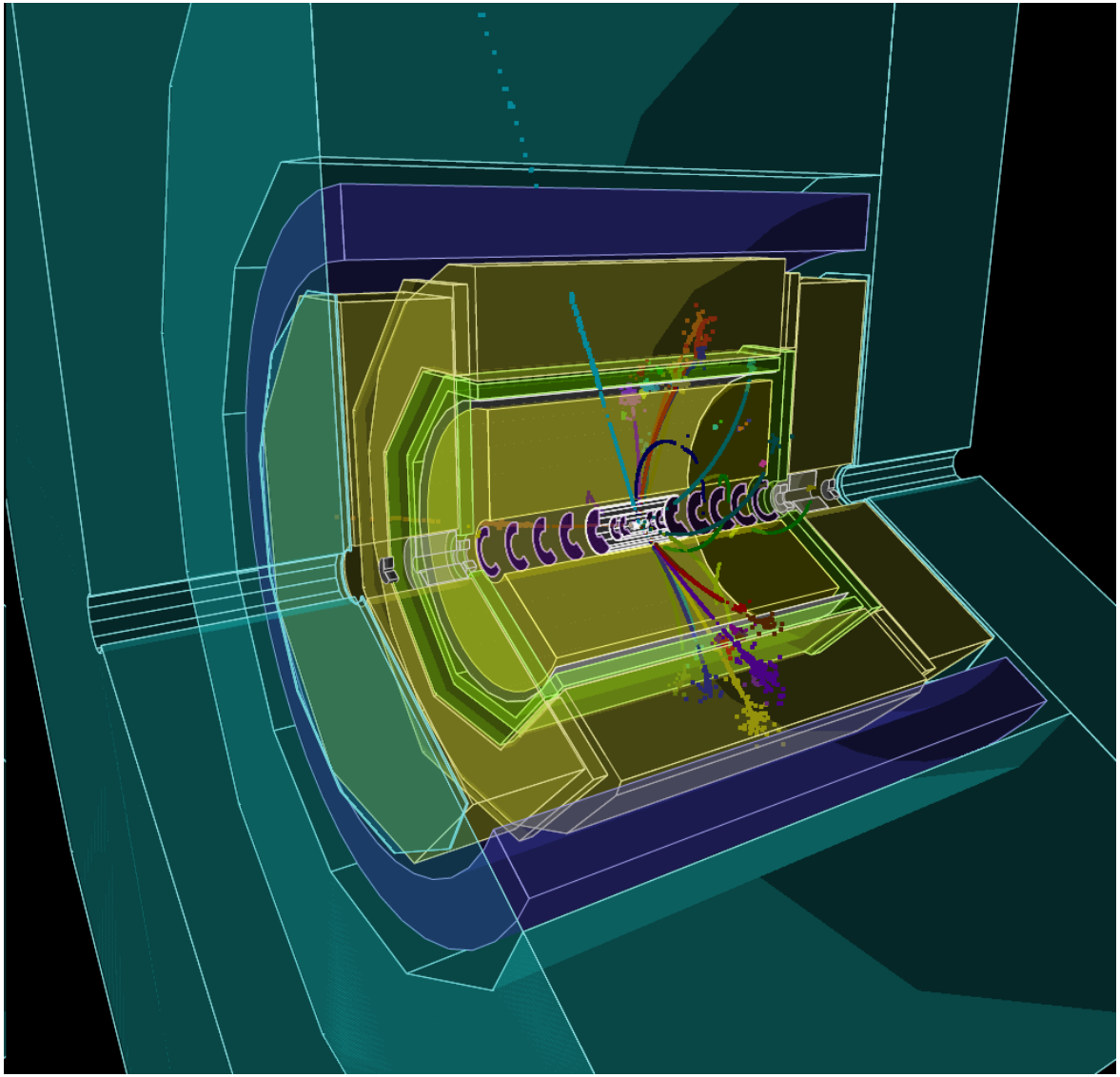
Topic: Model independent Higgs mass

Recoil mass measurement:

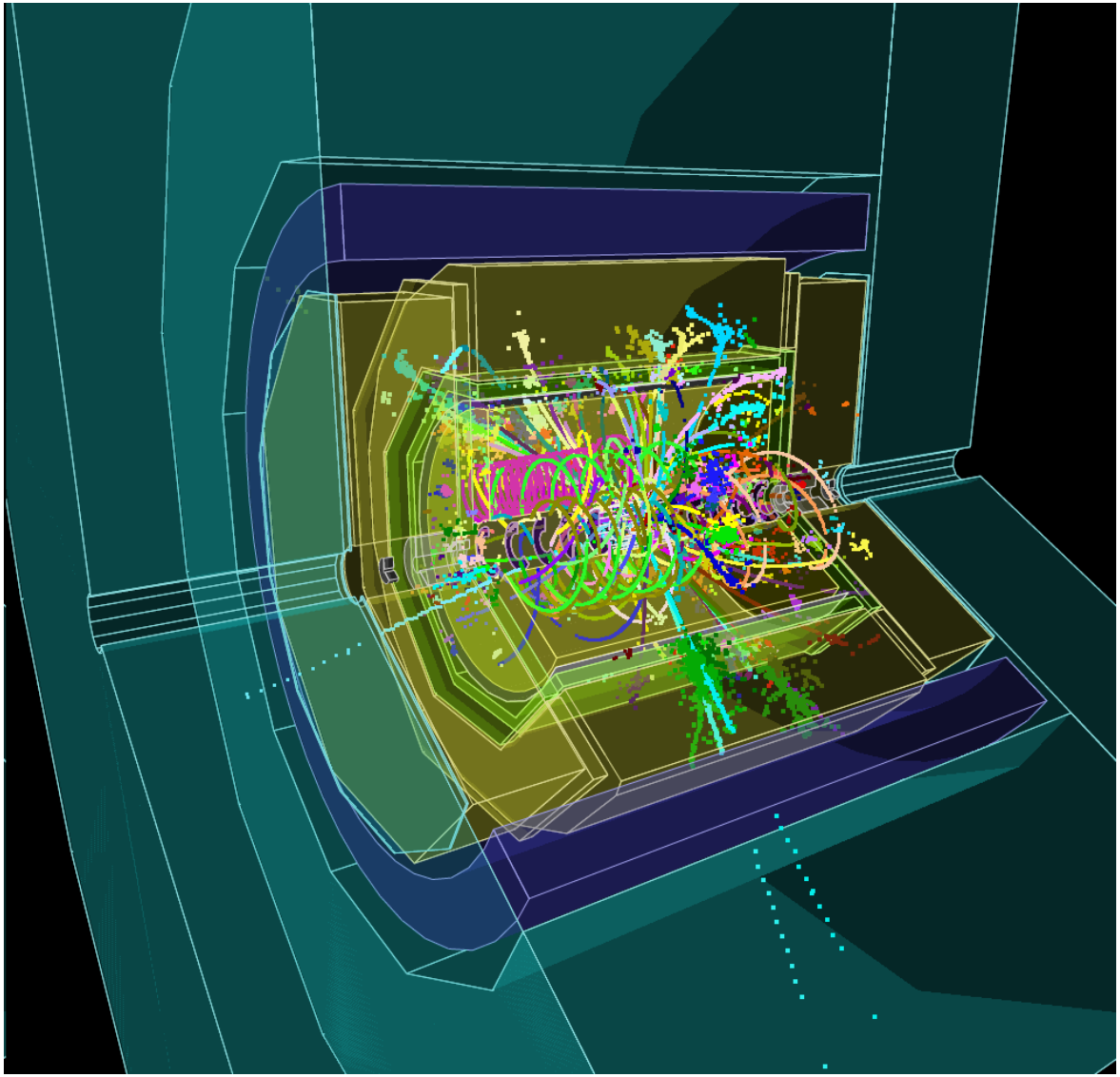
- Only reconstruct the $Z \to \mu\mu$
- Using E & p conservation the Higgs mass can be measured from the recoil independent of the decay mode



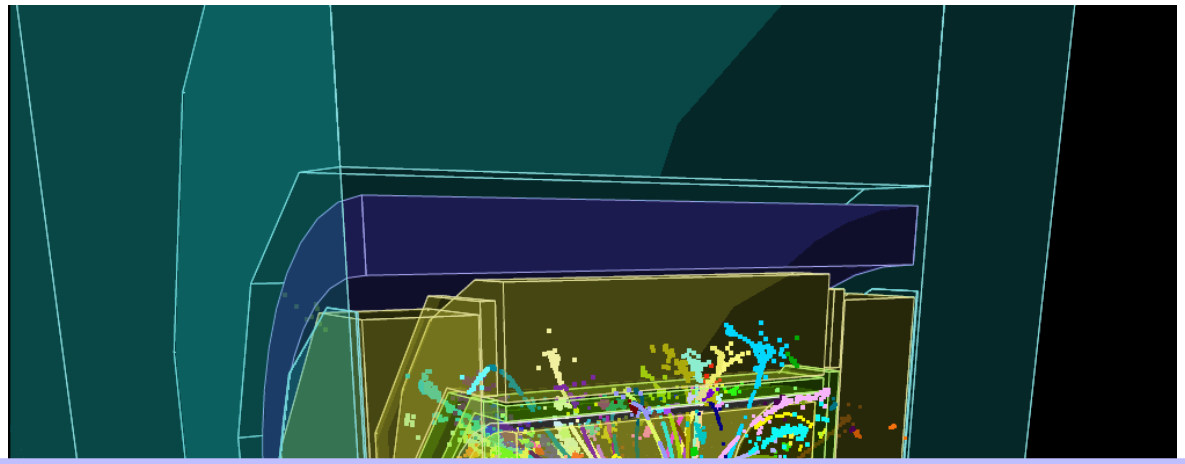
Example: $Zh \to \mu\mu q\bar{q}$ at 250 GeV



Example: $tth \rightarrow$ 8 jets at 1 TeV

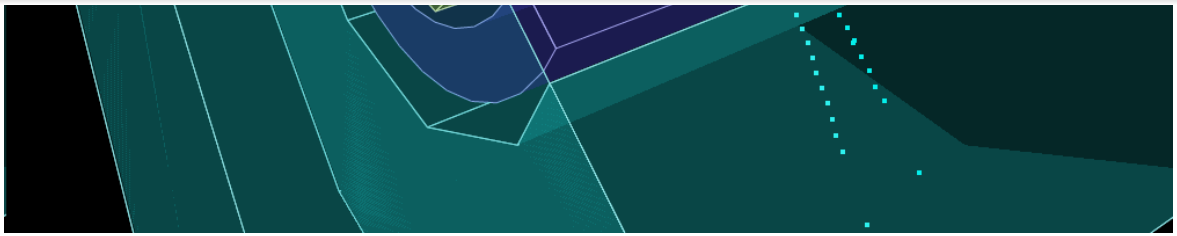


Example: $tth \rightarrow$ 8 jets at 1 TeV

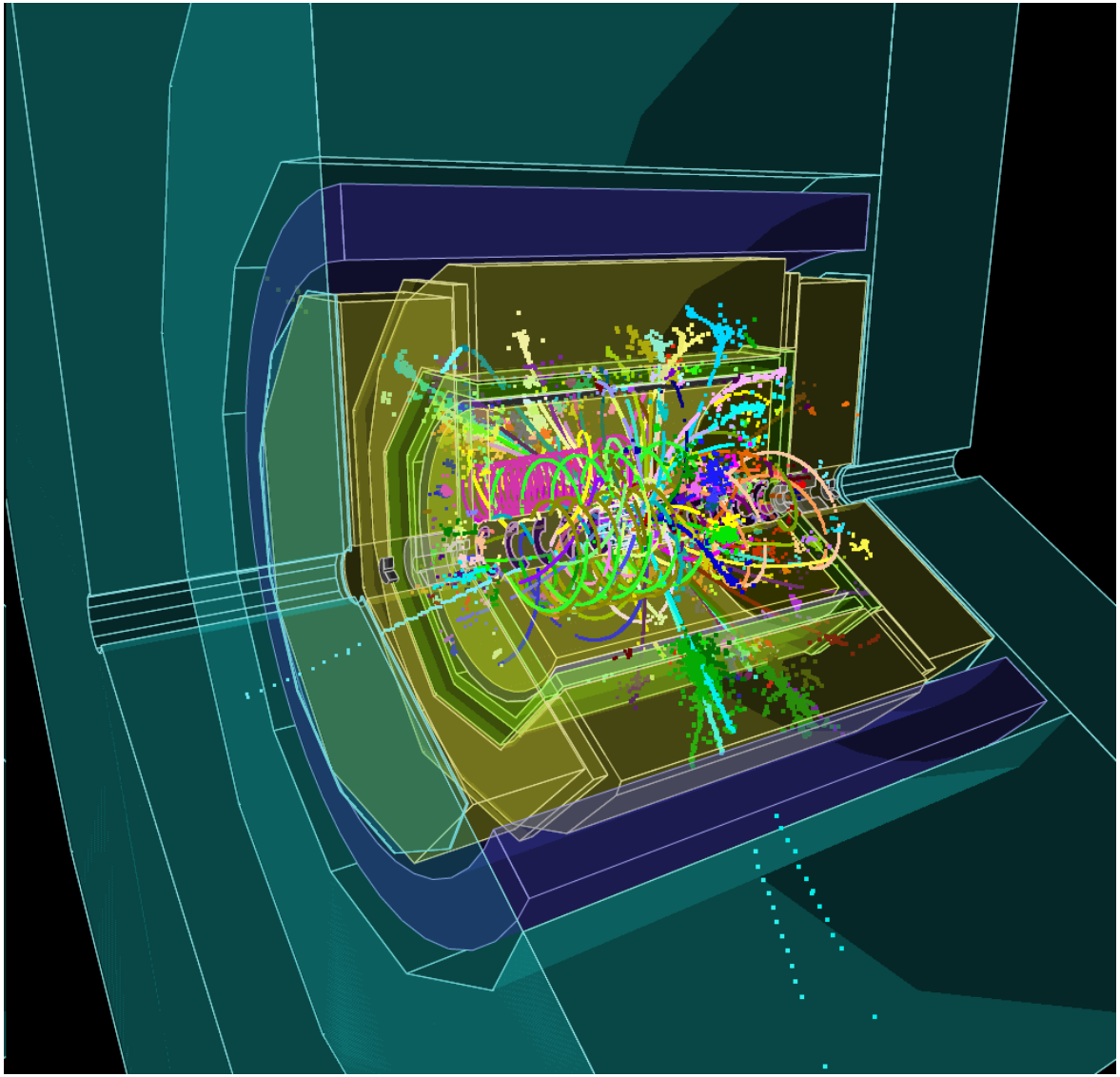


Topic: Higgs-top coupling

- Find all 8 jets.
- Find the jets from each top-decay and from the Higgs decay $\Rightarrow$
  - Flavour-tagging and jet-energy resolution.
- Find cross-section.



Example: $tth \rightarrow$ 8 jets at 1 TeV



The need for fast simulation at ILC

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer at LCWS 2011: *We need to update the physics case continuously*.
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the Letter Of Intent exercise in 2009 showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed at the ILC ?

The need for fast simulation at ILC

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer at LCWS 2011: *We need to update the physics case continuously*.
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the Letter Of Intent exercise in 2009 showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed at the ILC ?

The need for fast simulation at ILC

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer at LCWS 2011: *We need to update the physics case continuously*.
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the Letter Of Intent exercise in 2009 showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed at the ILC ?

The need for fast simulation at ILC

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer at LCWS 2011: *We need to update the physics case continuously*.
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the Letter Of Intent exercise in 2009 showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed at the ILC ?

The need for fast simulation at ILC

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer at LCWS 2011: *We need to update the physics case continuously*.
  - Light-weight: run anywhere, no need to read tons of manuals and doxygen pages.
  - Anyhow, the Letter Of Intent exercise in 2009 showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed at the ILC ?

Ex1: $\gamma\gamma$ 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.

- And that's only to keep up with the real data ...
- For MC-statistics not to be the dominating systematic error one would like to to have at least 5-10 times the data.

Ex1: $\gamma\gamma$ 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.

- And that's only to keep up with the real data ...
- For MC-statistics not to be the dominating systematic error one would like to to have at least 5-10 times the data.

Ex1: $\gamma\gamma$ 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.

- And that's only to keep up with the real data ...
- For MC-statistics not to be the dominating systematic error one would like to to have at least 5-10 times the data.

Ex1: $\gamma\gamma$ 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.

- And that's only to keep up with the real data ...
- For MC-statistics not to be the dominating systematic error one would like to to have at least 5-10 times the data.

SUSY parameter scans

Simple example:

- Modest pMSSM scan with 10 parameters.
- Scan each in eg. 10 steps
- Eg. 1000 events per point (also a modest requirement: in eg. sps1a' almost 1 million SUSY events are expected for 500 fb^{-1} !)
- = $10^{10} \times 1000 = 100 \times 10^9$ events to generate...

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

Also here: CPU millenniums with full simulation

SUSY parameter scans

Simple example:

- Modest pMSSM scan with 10 parameters.
- Scan each in eg. 10 steps
- Eg. 1000 events per point (also a modest requirement: in eg. sps1a' almost 1 million SUSY events are expected for 500 fb^{-1} !)
- = $10^{10} \times 1000 = 100 \times 10^9$ events to generate...

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

Also here: CPU millenniums with full simulation

Fast simulation types

Different types, with increasing level of sophistication:

- 4-vector smearing. ILC Ex. SimpleFastMCProcessor.
- Parametric. ILC ex.: SIMDET
- Covariance matrix machines. ILC ex.: LiCToy, SGV

Common for all:

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

I will talk about

"la Simulation à Grande Vitesse", SGV.

Fast simulation types

Different types, with increasing level of sophistication:

- 4-vector smearing. ILC Ex. SimpleFastMCProcessor.
- Parametric. ILC ex.: SIMDET
- Covariance matrix machines. ILC ex.: LiCToy, SGV

Common for all:

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

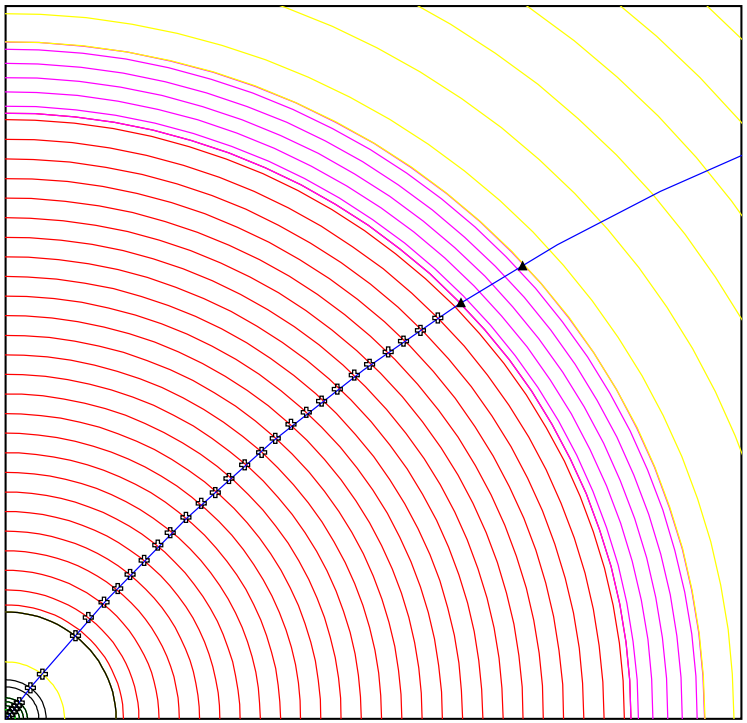
I will talk about

"la Simulation à Grande Vitesse", SGV.

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.

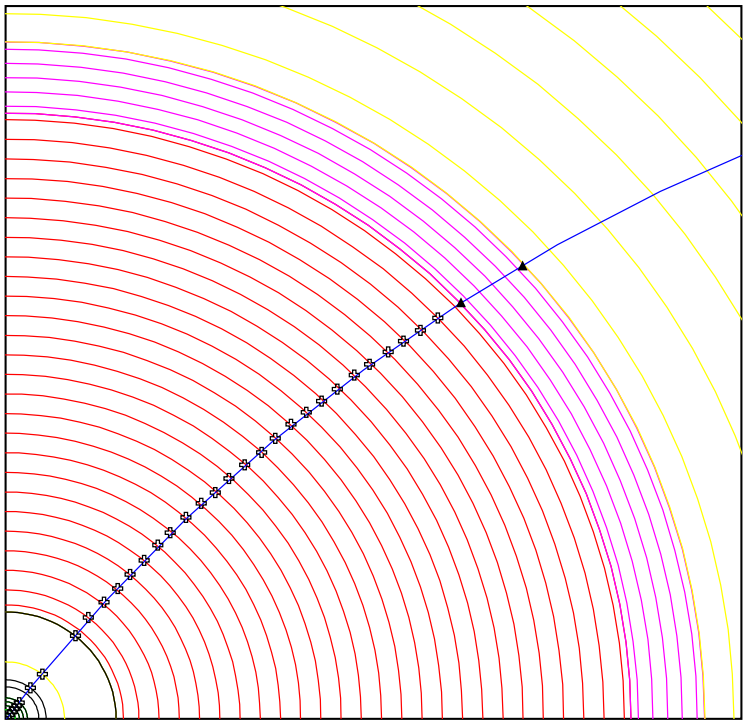


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: helix moved to (0,0,0) for this.

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.

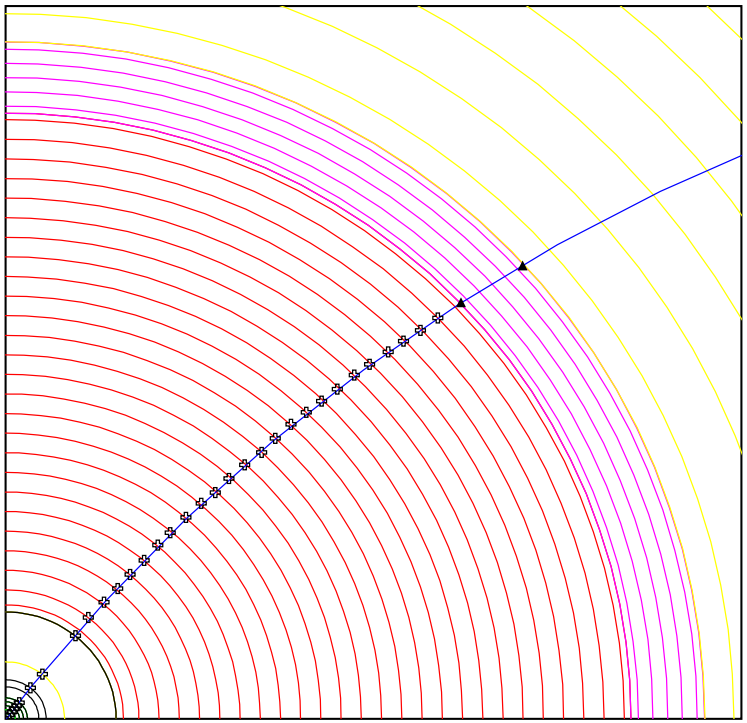


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: helix moved to (0,0,0) for this.

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.

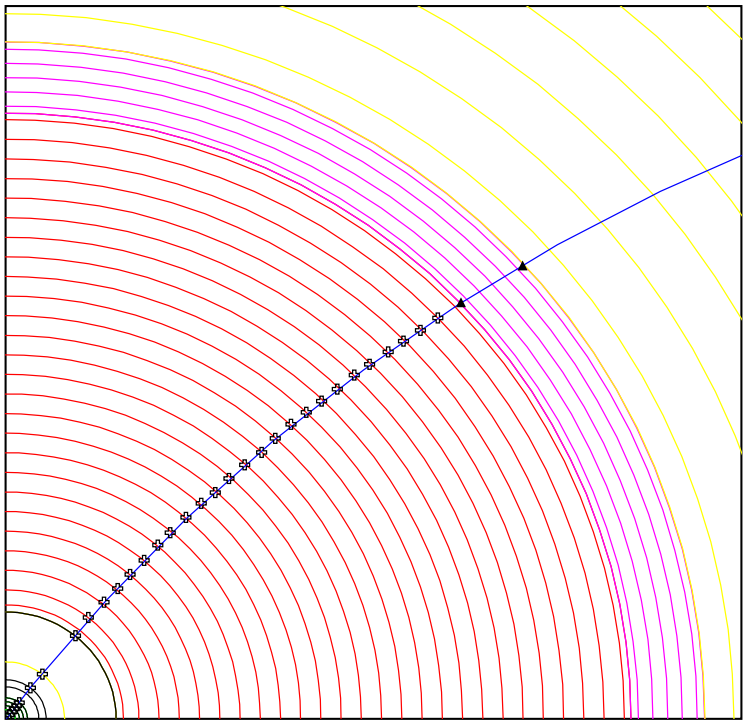


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: helix moved to (0,0,0) for this.

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.



- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- 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.

SGV: How the rest works

Calorimeters:

- Follow particle to intersection with calorimeters. Simulate:
  - Response type: MIP, EM-shower, hadronic shower, below threshold, etc.
  - Simulate single particle response from parameters.
  - Easy to plug in other (more sophisticated) shower-simulation. See Madalina Chera's talk tomorrow.

Other stuff:

- EM-interactions in detector material simulated
- Plug-ins for particle identification, track-finding efficiencies,...
- Information on hits accessible to analysis.

SGV: How the rest works

Calorimeters:

- Follow particle to intersection with calorimeters. Simulate:
  - Response type: MIP, EM-shower, hadronic shower, below threshold, etc.
  - Simulate single particle response from parameters.
  - Easy to plug in other (more sophisticated) shower-simulation. See Madalina Chera's talk tomorrow.

Other stuff:

- EM-interactions in detector material simulated
- Plug-ins for particle identification, track-finding efficiencies,...
- Information on hits accessible to analysis.

Technicalities

- Written in Fortran 95, a re-write of the Fortran77-based SGV2 series, battle-tested at LEP.
- Some CERNLIB dependence. Much reduced wrt. old F77 version, mostly by using Fortran 95's built-in matrix algebra. Just one core dependence (Choleski decomposition), so in a future version, the dependence will be optional.
- Managed in SVN.Install script included.
- Typical generation+simulation+reconstruction time $\mathcal{O}(10)$ ms.
- Timing verified to be faster (by 15%) than the f77 version.

Technicalities

- Written in Fortran 95, a re-write of the Fortran77-based SGV2 series, battle-tested at LEP.
- Some CERNLIB dependence. Much reduced wrt. old F77 version, mostly by using Fortran 95's built-in matrix algebra. Just one core dependence (Choleski decomposition), so in a future version, the dependence will be optional.
- Managed in SVN.Install script included.
- Typical generation+simulation+reconstruction time $\mathcal{O}(10)$ ms.
- Timing verified to be faster (by 15%) than the f77 version.

Technicalities

- Written in Fortran 95, a re-write of the Fortran77-based SGV2 series, battle-tested at LEP.
- Some CERNLIB dependence. Much reduced wrt. old F77 version, mostly by using Fortran 95's built-in matrix algebra. Just one core dependence (Choleski decomposition), so in a future version, the dependence will be optional.
- Managed in SVN.Install script included.
- Typical generation+simulation+reconstruction time $\mathcal{O}(10)$ ms.
- Timing verified to be faster (by 15%) than the f77 version.

Technicalities

- Written in Fortran 95, a re-write of the Fortran77-based SGV2 series, battle-tested at LEP.
- Some CERNLIB dependence. Much reduced wrt. old F77 version, mostly by using Fortran 95's built-in matrix algebra. Just one core dependence (Choleski decomposition), so in a future version, the dependence will be optional.
- Managed in SVN.Install script included.
- Typical generation+simulation+reconstruction time $\mathcal{O}(10)$ ms.
- Timing verified to be faster (by 15%) than the f77 version.

Features

- Physics events from callable PYTHIA, Whizard, with beam-spectrum.
- .. or input from PYJETS or stdhep.
- Output of generated event to PYJETS or stdhep.
- samples subdirectory with steering and code for eg. scan single
  - Scan single particles ("particle gun").
  - Create hbook ntuple with "all" information (can be converted to ROOT w/ h2root). (*Is there a C-binding to root, at least for tree definition, filling and I/O ?*)
  - Code for more elaborate calorimeters
    - Particle-Flow with confusion (Madalina's talk).
    - More complicated acceptance, eg. small angle calorimeters with beam-background and crossing-angle.
  - Output LCIO DST, the common ILC data-model, with content identical to that for FullSim ⇒ SGV for rapid prototyping of analyses.

Features

- Physics events from callable PYTHIA, Whizard, with beam-spectrum.
- .. or input from PYJETS or stdhep.
- Output of generated event to PYJETS or stdhep.
- samples subdirectory with steering and code for eg. scan single
  - Scan single particles (“particle gun”).
  - Create hbook ntuple with “all” information (can be converted to ROOT w/ h2root). (*Is there a C-binding to root, at least for tree definition, filling and I/O ?*)
  - Code for more elaborate calorimeters
    - Particle-Flow with confusion (Madalina's talk).
    - More complicated acceptance, eg. small angle calorimeters with beam-background and crossing angle.
  - Output LCIO DST, the common ILC data-model, with content identical to that for FullSim ⇒ SGV for rapid prototyping of analyses.

Features

- Physics events from callable PYTHIA, Whizard, with beam-spectrum.
- .. or input from PYJETS or stdhep.
- Output of generated event to PYJETS or stdhep.
- samples subdirectory with steering and code for eg. scan single
  - Scan single particles (“particle gun”).
  - Create hbook ntuple with “all” information (can be converted to ROOT w/ h2root). (*Is there a C-binding to root, at least for tree definition, filling and I/O ?*)
  - Code for more elaborate calorimeters
    - Particle-Flow with confusion (Madalina’s talk).
    - More complicated acceptance, eg. small angle calorimters with beam-background and crossing-angle.
  - Output LCIO DST, the common ILC data-model, with content identical to that for FullSim $\Rightarrow$ SGV for rapid prototyping of analyses.

Features

- Physics events from callable PYTHIA, Whizard, with beam-spectrum.
- .. or input from PYJETS or stdhep.
- Output of generated event to PYJETS or stdhep.
- samples subdirectory with steering and code for eg. scan single
  - Scan single particles (“particle gun”).
  - Create hbook ntuple with “all” information (can be converted to ROOT w/ h2root). (*Is there a C-binding to root, at least for tree definition, filling and I/O ?*)
  - Code for more elaborate calorimeters
    - Particle-Flow with confusion (Madalina’s talk).
    - More complicated acceptance, eg. small angle calorimters with beam-background and crossing-angle.
  - Output LCIO DST, the common ILC data-model, with content identical to that for FullSim $\Rightarrow$ SGV for rapid prototyping of analyses.

Stucture

SGV is made of six loosely connected parts:

- The Steering, which takes care of initialisation and ending, and runs the event loop.
- In the event loop,
  - The Event Generator, which either calls the external generator, or reads in pre-generated events.
  - The Detector Simulation
  - The Event Dispatcher

  is called.
- The Detector Simulation calls
  - The Covariance Matrix Machine
  - The Calorimeter response simulation.

Stucture

- The Event Dispatcher trasferes the data directly to the Event Analysis, and/or writes to an file / FIFO / shared memmory (RZ/FZ).
- Communication between parts - steering *and* data - is by arguments only (except for Event Generator → Detector Simulation → Event Dispatcher.)
- Therefore, the Event analysis can be called by any process that can attribute values to the elements the Event-dispatcher sends to the Analyser.
  - Use-case: ROOT delegates analysis to SGV.

Stucture

- The Event Dispatcher trasferes the data directly to the Event Analysis, and/or writes to an file / FIFO / shared memmory (RZ/FZ).
- Communication between parts - steering *and* data - is by arguments only (except for Event Generator → Detector Simulation → Event Dispatcher.)
- Therefore, the Event analysis can be called by any process that can attribute values to the elements the Event-dispatcher sends to the Analyser.
  - Use-case: ROOT delegates analysis to SGV.

Stucture

- The Event Dispatcher trasferes the data directly to the Event Analysis, and/or writes to an file / FIFO / shared memmory (RZ/FZ).
- Communication between parts - steering *and* data - is by arguments only (except for Event Generator → Detector Simulation → Event Dispatcher.)
- Therefore, the Event analysis can be called by any process that can attribute values to the elements the Event-dispatcher sends to the Analyser.
  - Use-case: ROOT delegates analysis to SGV.

Usage

- Steering:
  - A single steering file for all the steps - general, event generation, detector simulation, analysis.
- User code:
  - One routine (and it's CONTAINS:s or USE:es).
  - Default version compiled in.
  - Many more complicated examples in the "samples" sub-directory.
- User data, delivered in Module-global arrays:
  - Extended 4-vectors .
  - Track helix parameters with correlations.
  - Calorimetric clusters.
  - When relevant: true values.
  - Auxiliary information on particle history, detector-elements used etc.
  - Event-global variables.

Usage

- Steering:
  - A single steering file for all the steps - general, event generation, detector simulation, analysis.
- User code:
  - One routine (and it's CONTAINS:s or USE:es).
  - Default version compiled in.
  - Many more complicated examples in the "samples" sub-directory.
- User data, delivered in Module-global arrays:
  - Extended 4-vectors .
  - Track helix parameters with correlations.
  - Calorimetric clusters.
  - When relevant: true values.
  - Auxiliary information on particle history, detector-elements used etc.
  - Event-global variables.

Usage

- Steering:
  - A single steering file for all the steps - general, event generation, detector simulation, analysis.
- User code:
  - One routine (and it's CONTAINS:s or USE:es).
  - Default version compiled in.
  - Many more complicated examples in the "samples" sub-directory.
- User data, delivered in Module-global arrays:
  - Extended 4-vectors .
  - Track helix parameters with correlations.
  - Calorimetric clusters.
  - When relevant: true values.
  - Auxiliary information on particle history, detector-elements used etc.
  - Event-global variables.

Usage

- User Analysis tasks :
  - Jet-finding.
  - Event-shapes.
  - Primary and secondary vertex fitting.
  - Impact parameters.

  Can be calculated by routines, included in SGV. Access routines give an easy interface to the detector geometry.
- Assemble source, Compile & link of desired version:
  - Done by cresgvexe (cresgvso) script.
  - Comand-line parameters for detailed selections : -D:s , extra libraries, non-defaultscompiler/linker options, etc.

Usage

- User Analysis tasks :
  - Jet-finding.
  - Event-shapes.
  - Primary and secondary vertex fitting.
  - Impact parameters.

  Can be calculated by routines, included in SGV. Access routines give an easy interface to the detector geometry.
- Assemble source, Compile & link of desired version:
  - Done by cresgvexe (cresgvso) script.
  - Comand-line parameters for detailed selections : -D:s , extra libraries, non-defaultscompiler/linker options, etc.

Usage

- Generators:
  - Interfaces to PYTHIA and Whizard built-in.
  - Easy to interface to others (but depends on how well structured the generator is...).
  - Alternatively: Read events from external file (but beware of I/O !)
- Detector geometry:
  - Planes and cylinders.
  - Attributes attached (measurement, material, names,...).
  - Read from a human-readable ASCII-file (ex).
  - Simple visualisation of the detector included.
  - Up to three detectors can be loaded simultaneously, and will be looped over event by event.

Usage

- Generators:
  - Interfaces to PYTHIA and Whizard built-in.
  - Easy to interface to others (but depends on how well structured the generator is...).
  - Alternatively: Read events from external file (but beware of I/O !)
- Detector geometry:
  - Planes and cylinders.
  - Attributes attached (measurement, material, names,...).
  - Read from a human-readable ASCII-file (ex).
  - Simple visualisation of the detector included.
  - Up to three detectors can be loaded simultaneously, and will be looped over event by event.

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.
  - Get the LCIO-DST writer set up

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.
  - Get the LCIO-DST writer set up

Installing SGV

Do

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

Then

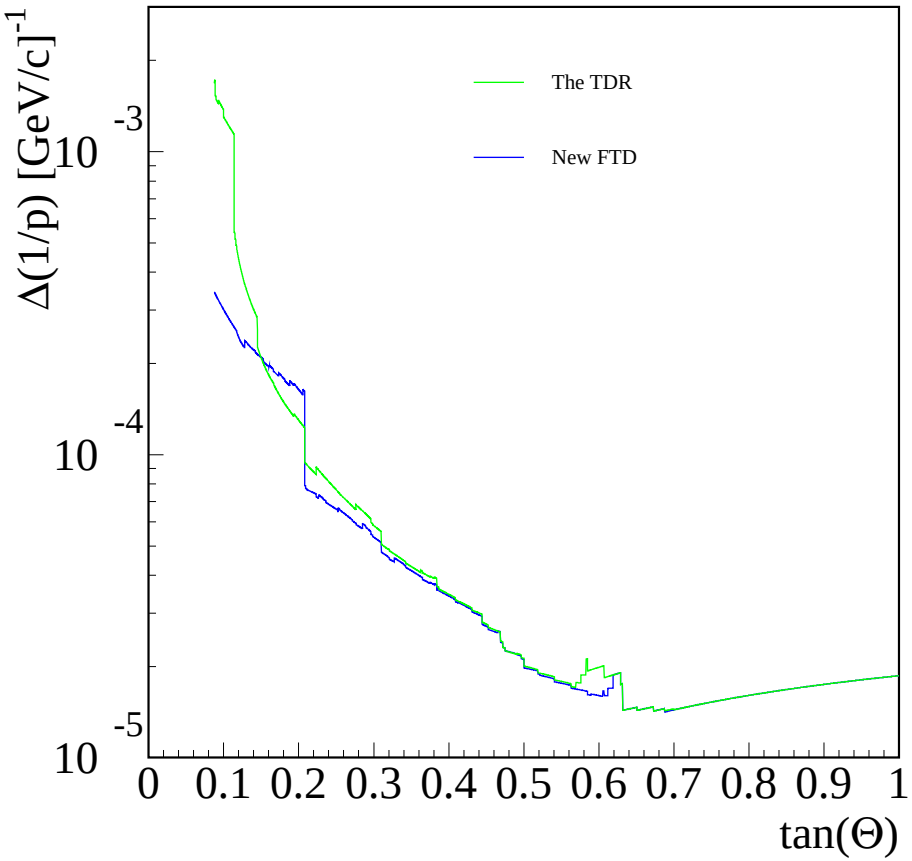
cd sgv ; . ./install

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.
  - Get the LCIO-DST writer set up

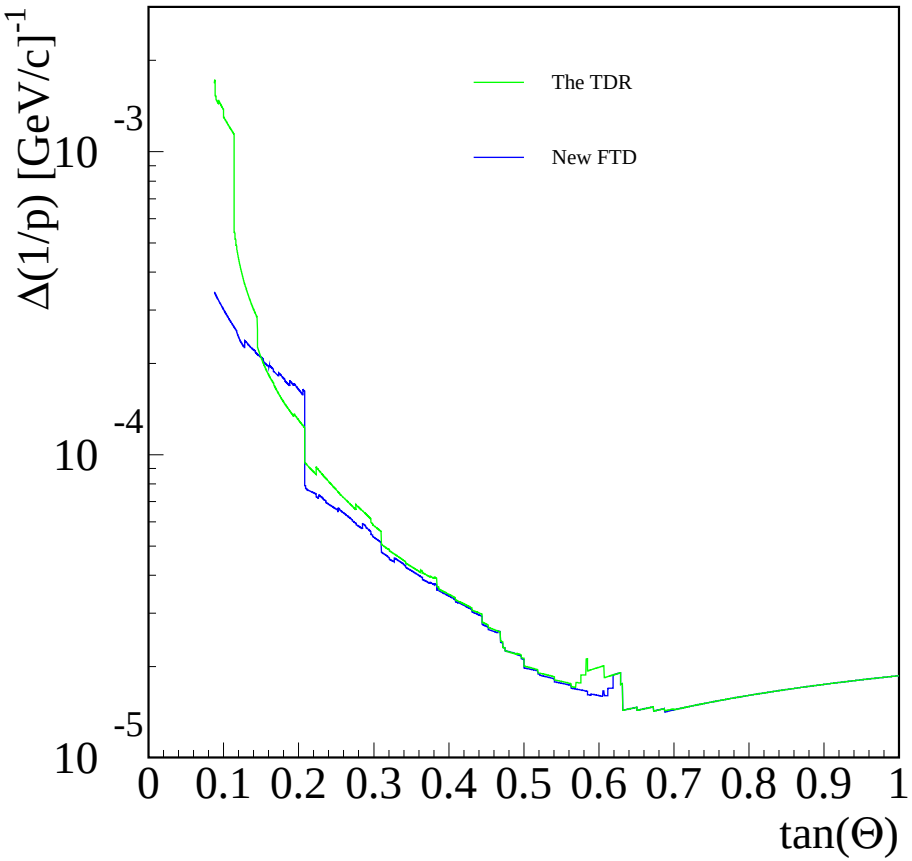
Ex: ILD tracker-system design (Vienna 2005).

- I: The divergence in the TDR: Once the last disk is hit, the $1/\sqrt{\tan\theta}$ is back !
- II: The step: End of The Vertex Detector
- Remedy I:Add *disks* all the way to the end of the TPC (5 more strip-disks)
- Remedy II: Add a *pixel disk* with $\sigma_{point} = 4\mu$ just outside the VD.



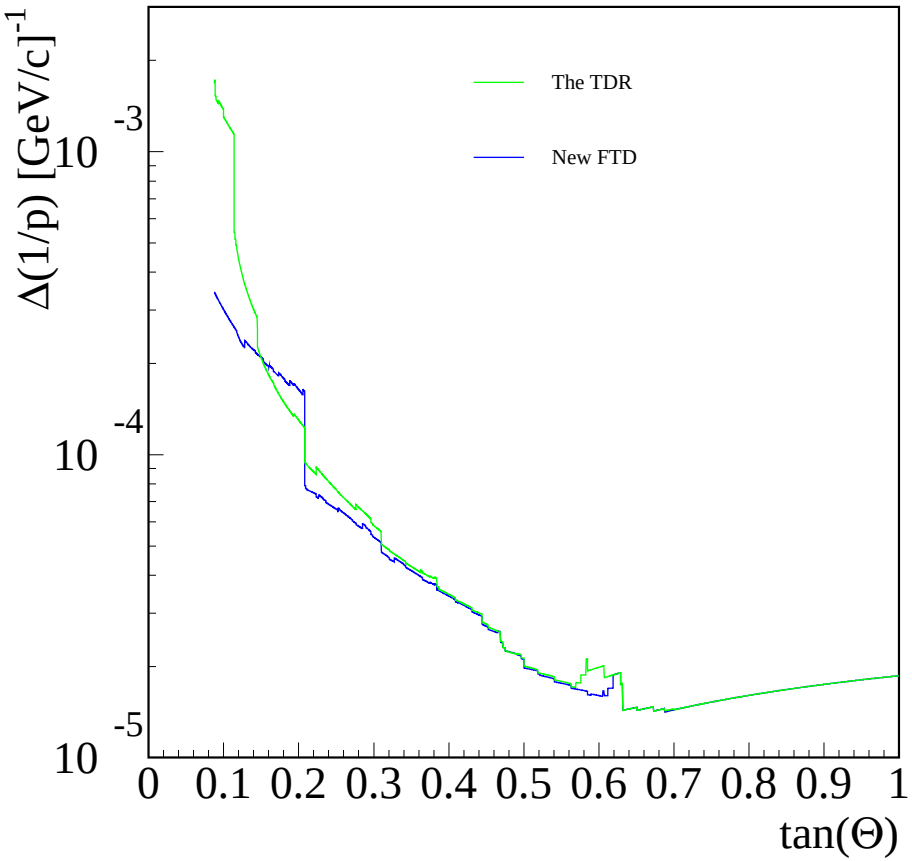
Ex: ILD tracker-system design (Vienna 2005).

- I: The divergence in the TDR: Once the last disk is hit, the $1/\sqrt{\tan\theta}$ is back !
- II: The step: End of The Vertex Detector
- Remedy I:Add *disks* all the way to the end of the TPC (5 more strip-disks)
- Remedy II: Add a *pixel disk* with $\sigma_{point} = 4\mu$ just outside the VD.



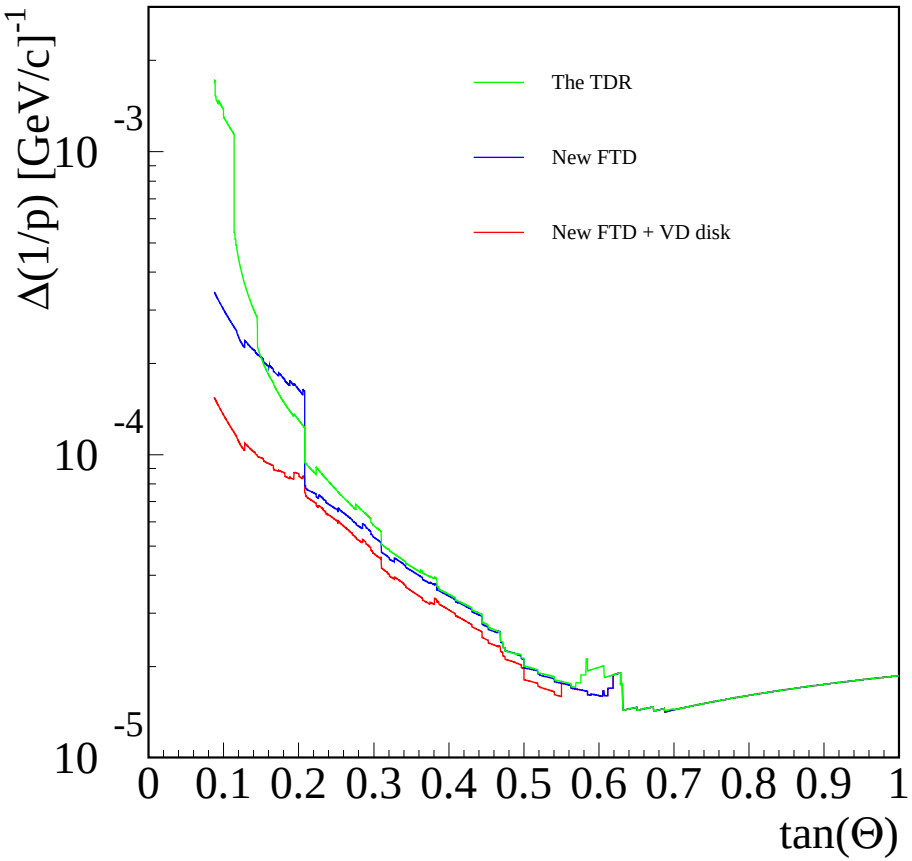
Ex: ILD tracker-system design (Vienna 2005).

- I: The divergence in the TDR: Once the last disk is hit, the $1/\sqrt{\tan\theta}$ is back !
- II: The step: End of The Vertex Detector
- Remedy I:Add *disks* all the way to the end of the TPC (5 more strip-disks)
- Remedy II: Add a *pixel disk* with $\sigma_{point} = 4\mu$ just outside the VD.

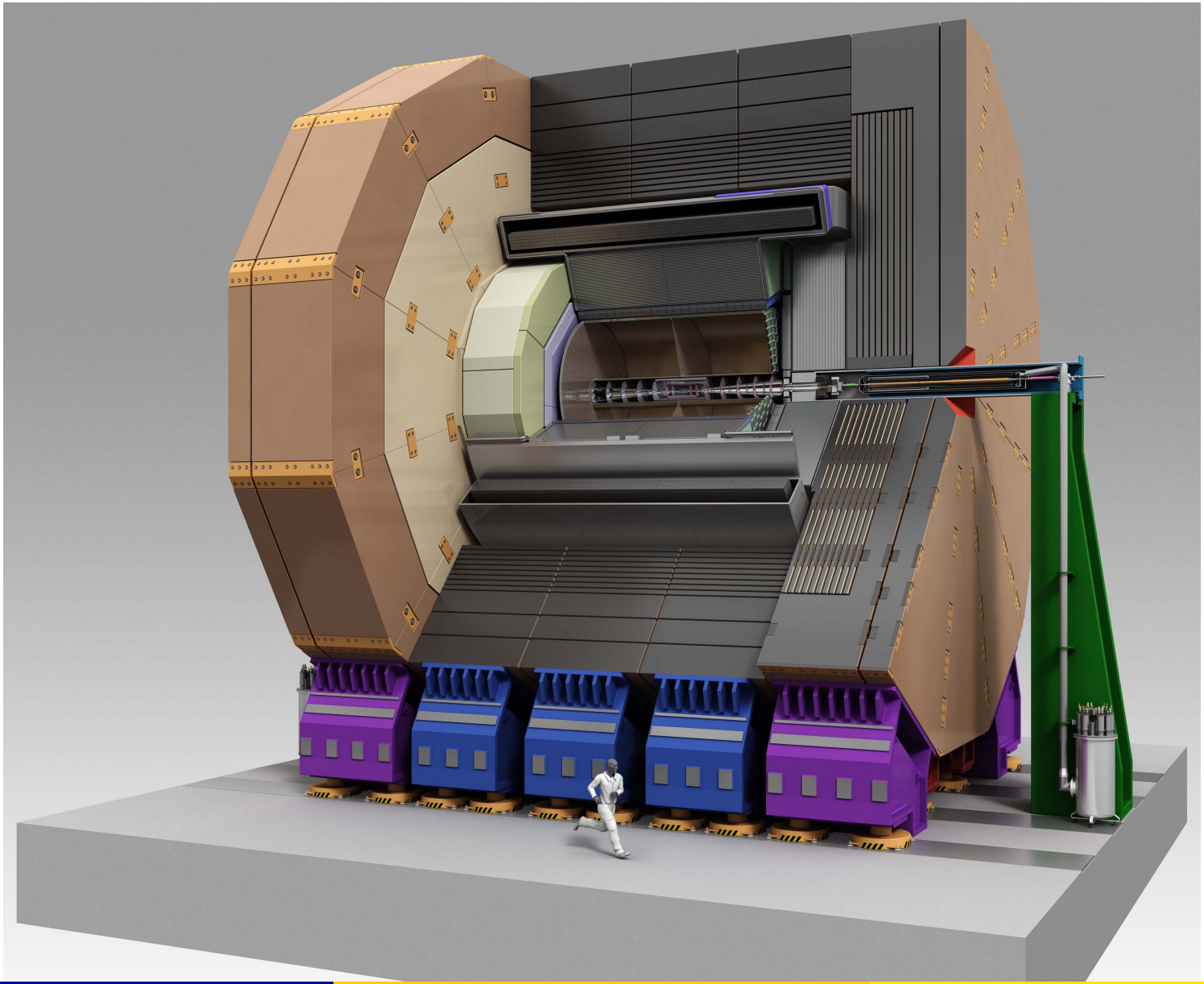


Ex: ILD tracker-system design (Vienna 2005).

- I: The divergence in the TDR: Once the last disk is hit, the $1/\sqrt{\tan\theta}$ is back !
- II: The step: End of The Vertex Detector
- Remedy I:Add *disks* all the way to the end of the TPC (5 more strip-disks)
- Remedy II: Add a *pixel disk* with $\sigma_{point} = 4\mu$ just outside the VD.

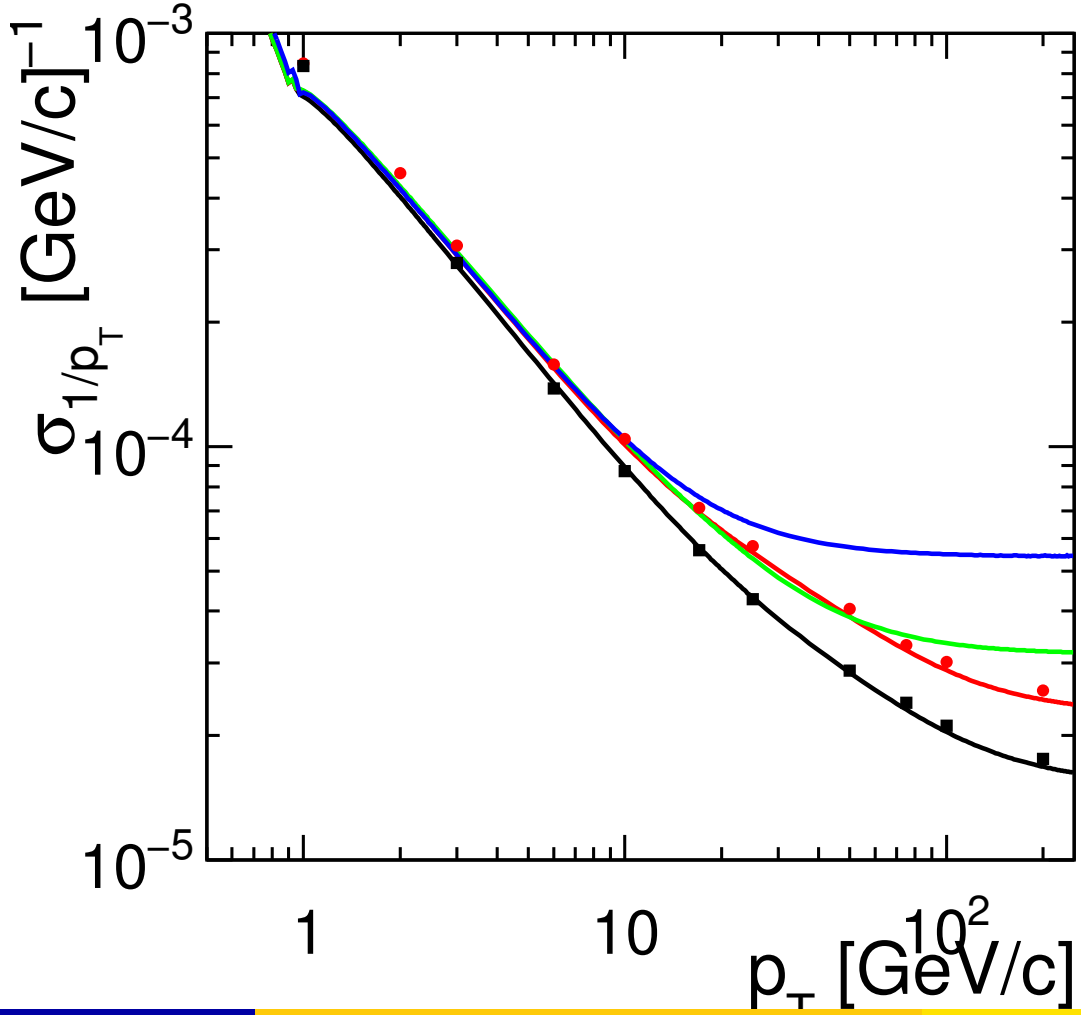


The ILD Detector



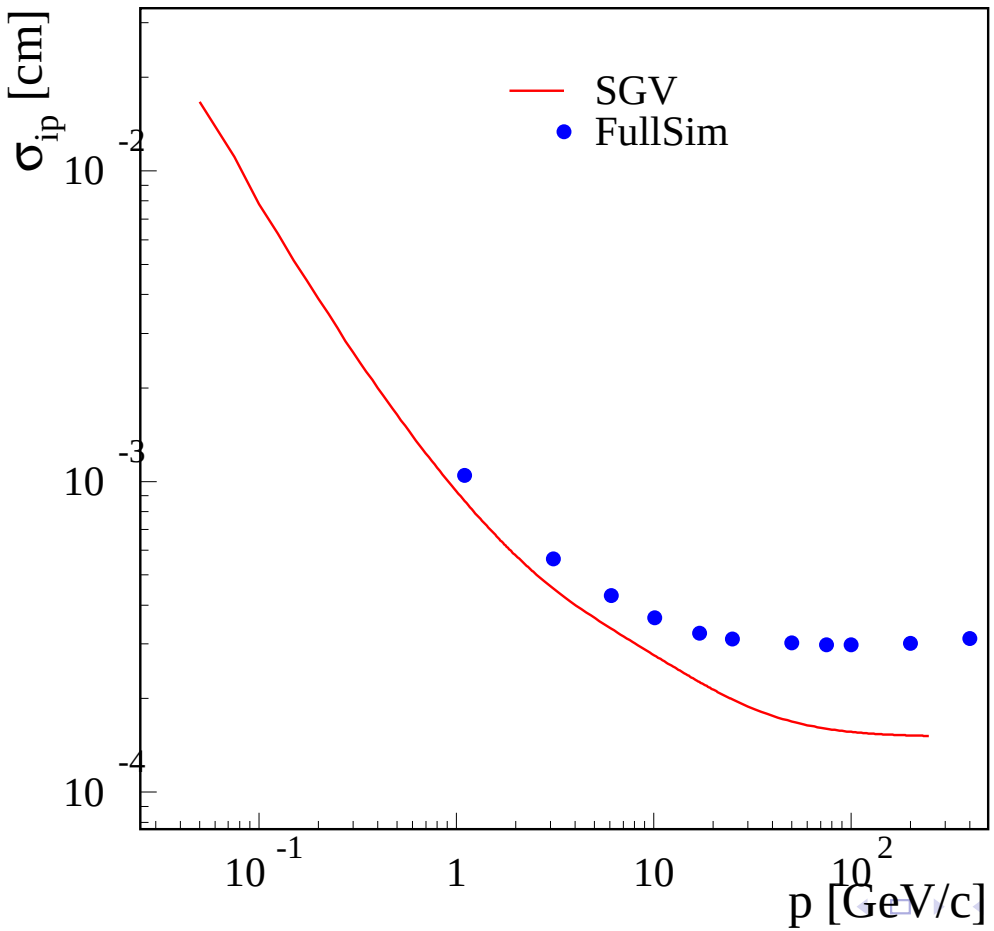
SGV and FullSim ILD: momentum resolution

Lines: SGV, dots: Mokka+Marlin



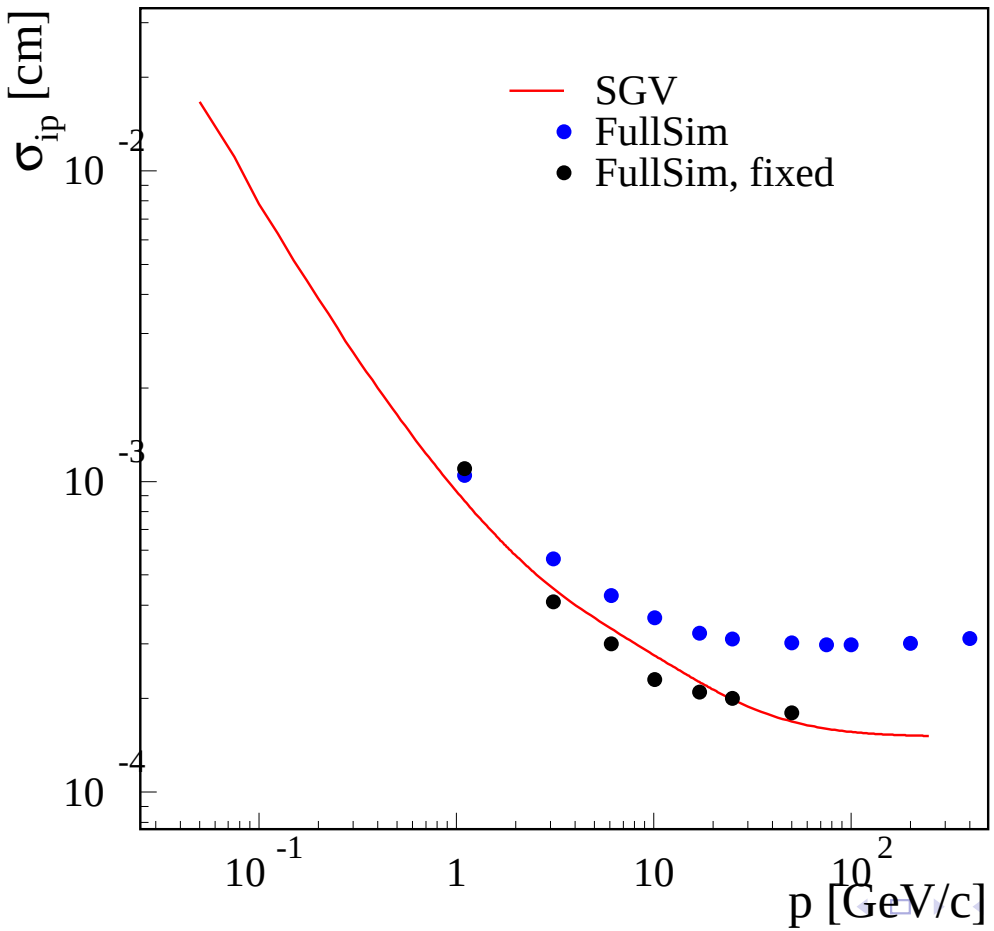
SGV and FullSim ILD: ip resolution vs P

Lines: SGV, dots: Mokka+Marlin



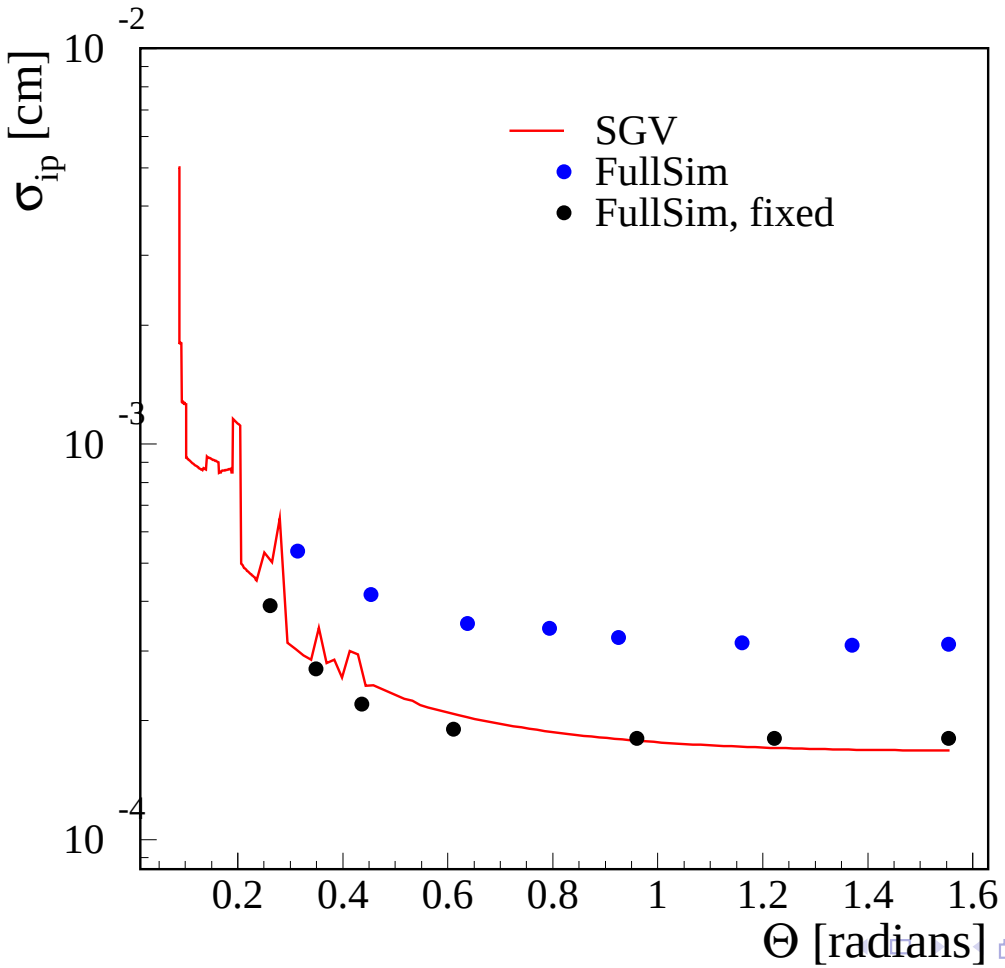
SGV and FullSim ILD: ip resolution vs P

Lines: SGV, dots: Mokka+Marlin



SGV and FullSim ILD: ip resolution vs angle

Lines: SGV, dots: Mokka+Marlin



SGV and FullSim ILD: Full system

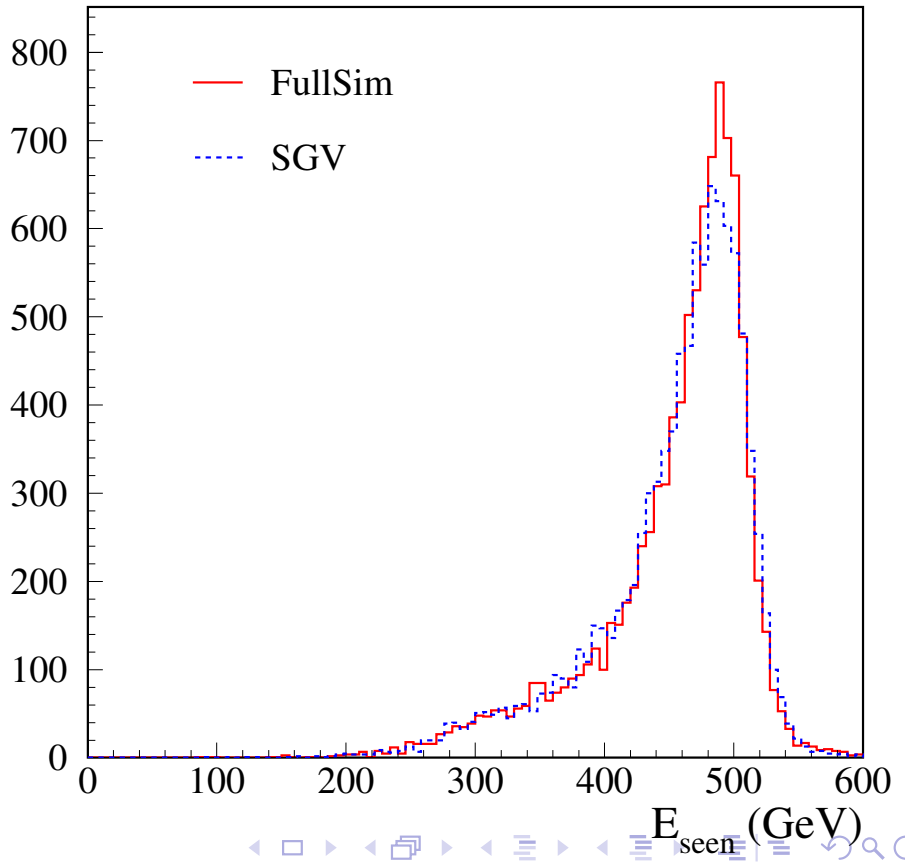
Feed exactly the same physics events through FullSim or SGV.

- 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
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !

SGV and FullSim ILD: Full system

Feed exactly the same physics events through FullSim or SGV.

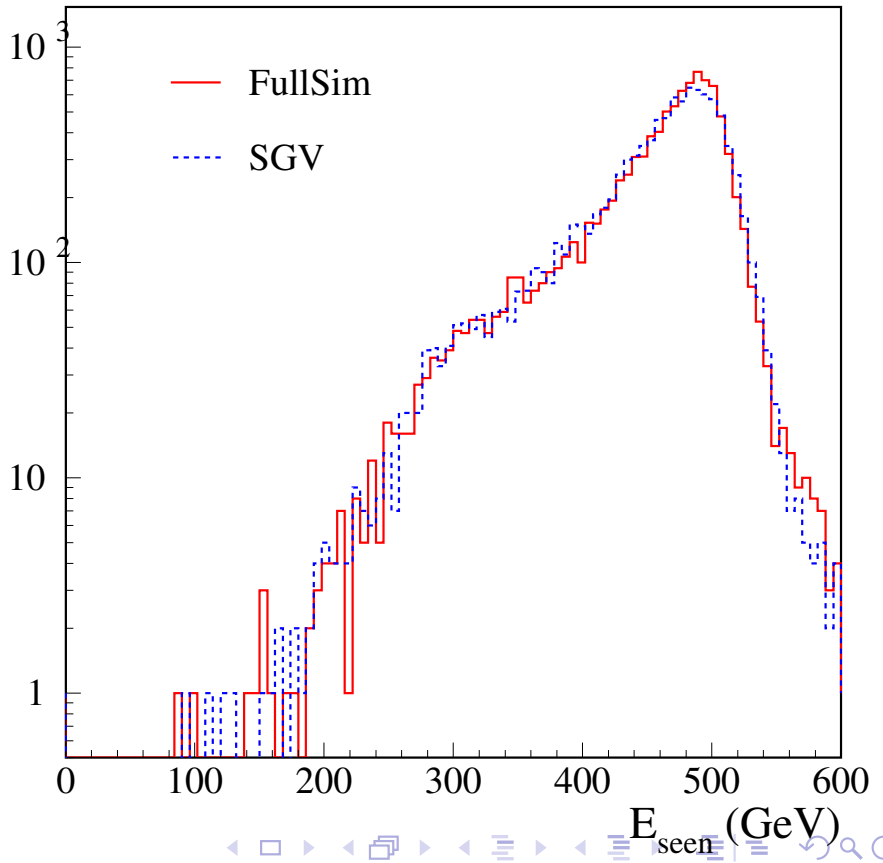
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !



SGV and FullSim ILD: Full system

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
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !



SGV and FullSim ILD: Full system

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
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !

SGV and FullSim ILD: Full system

Feed exactly the same physics events through FullSim or SGV.

- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True 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.

- b-tag
- Error on Higgs-selfcoupling:
  SGV 17% , FullSim 18 % !

SGV and FullSim ILD: Full system

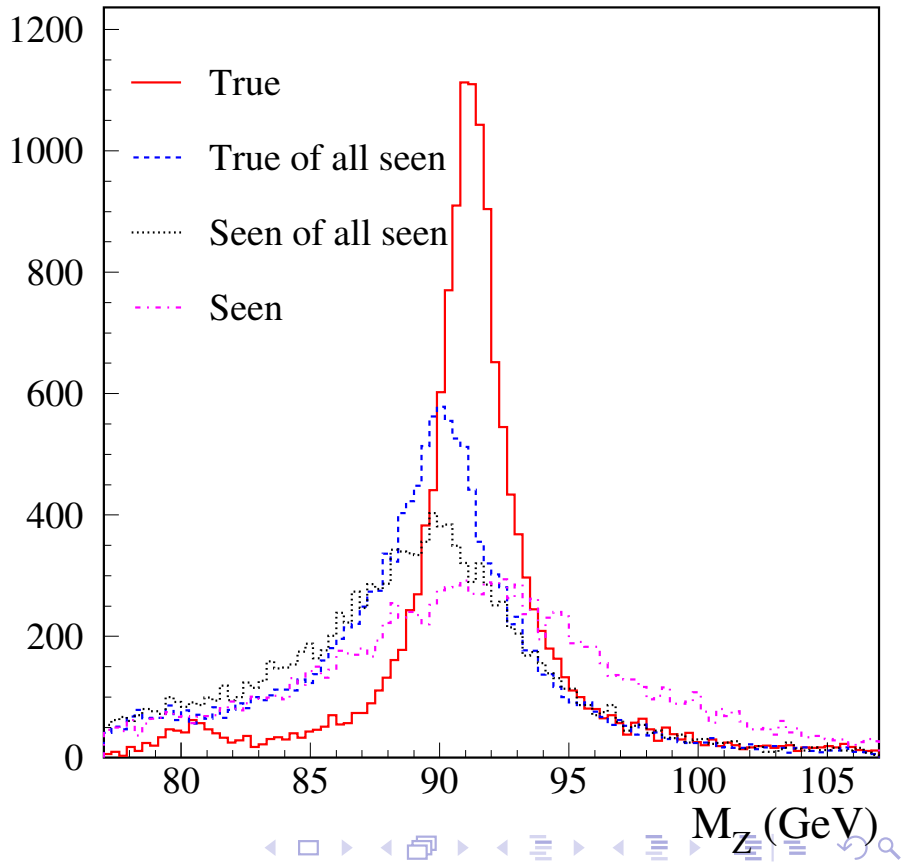
Feed exactly the same physics events through FullSim or SGV.

- 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
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !

SGV and FullSim ILD: Full system

Feed exactly the same physics events through FullSim or SGV.

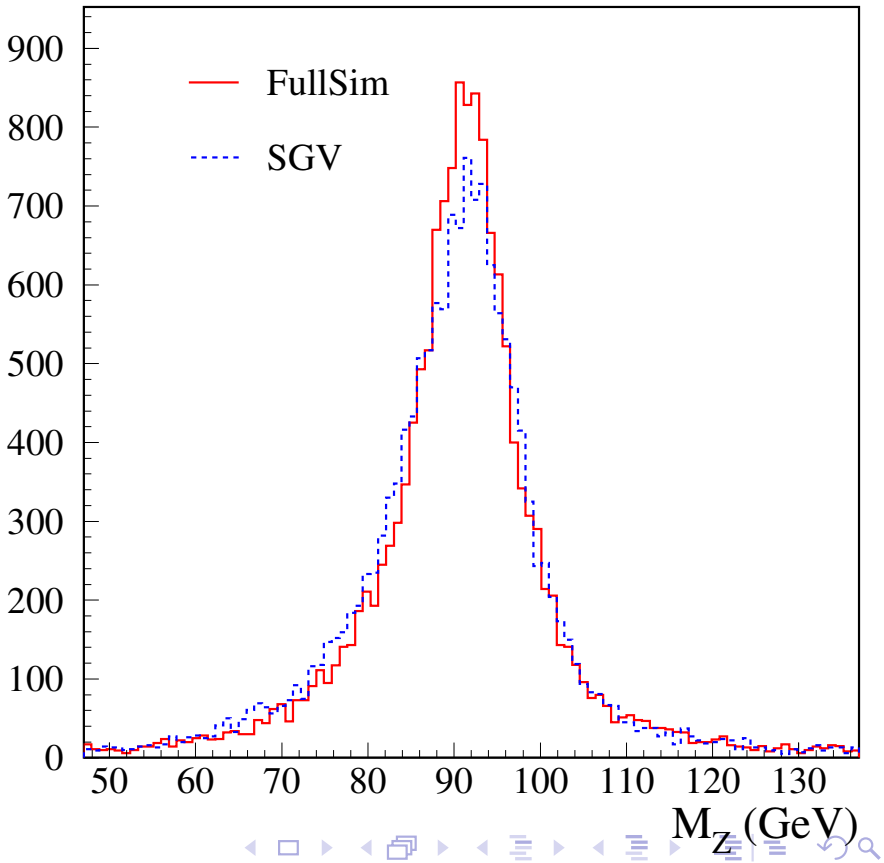
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !



SGV and FullSim ILD: Full system

Feed exactly the same physics events through FullSim or SGV.

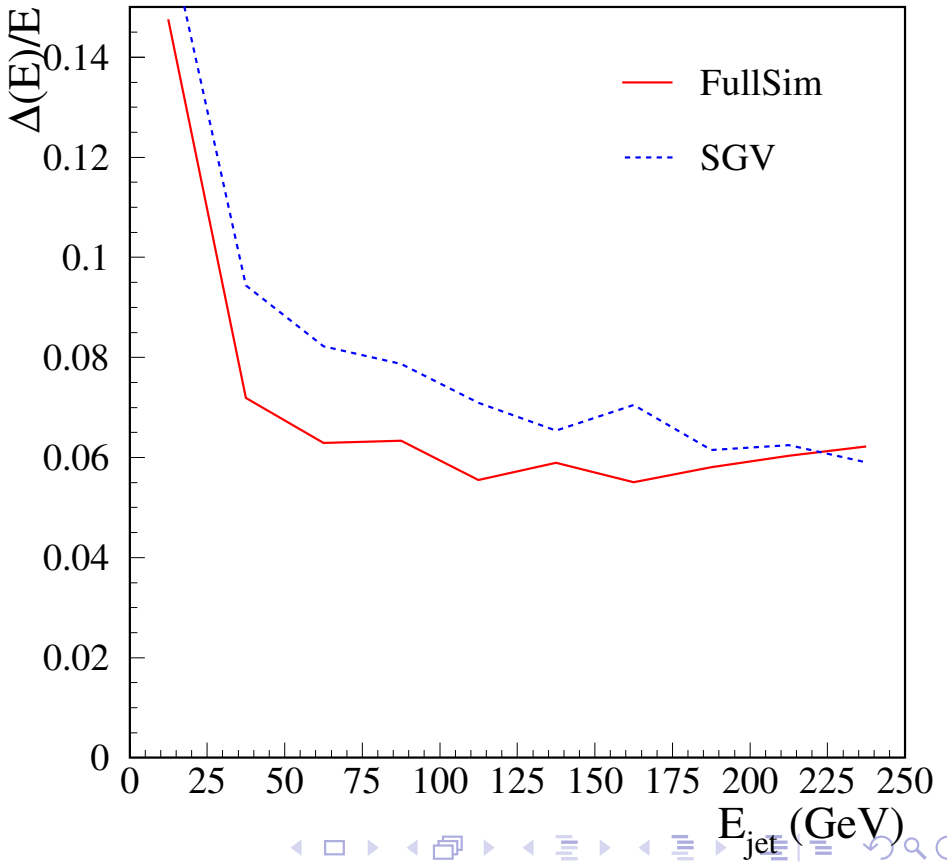
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !



SGV and FullSim ILD: Full system

Feed exactly the same physics events through FullSim or SGV.

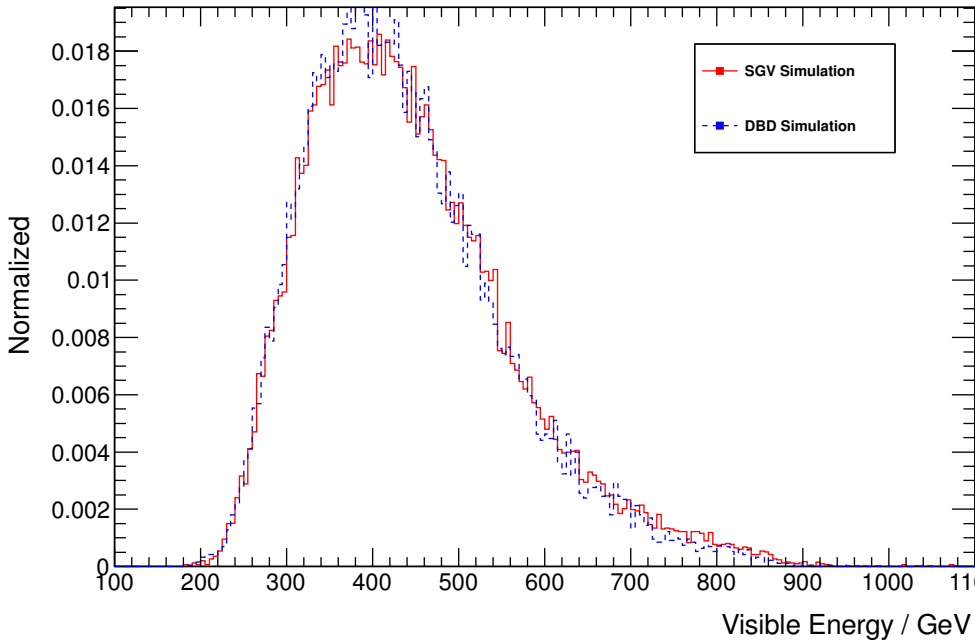
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !



SGV and FullSim ILD: Full system

Feed exactly the same physics events through FullSim or SGV.

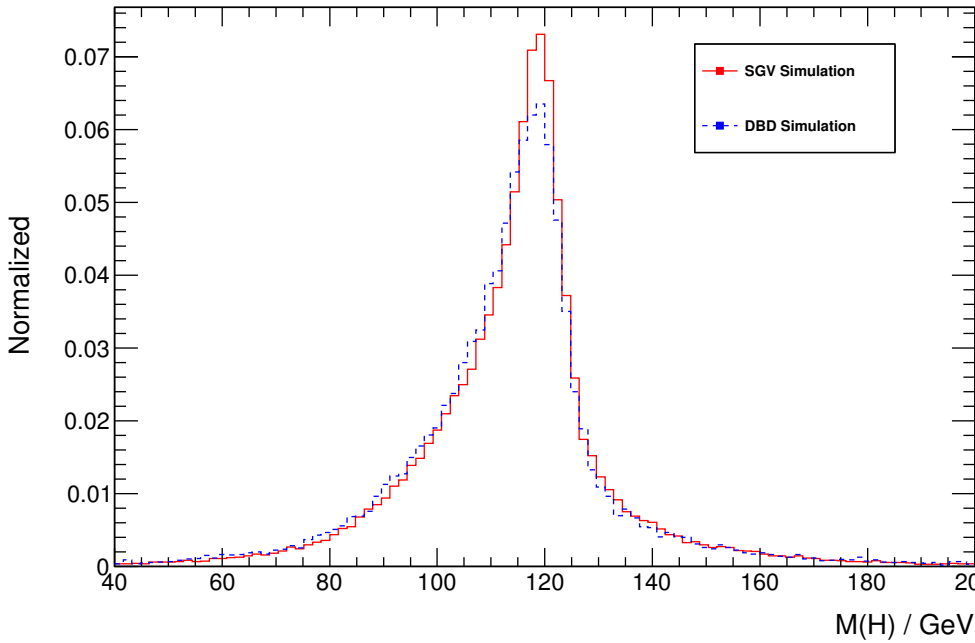
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !



SGV and FullSim ILD: Full system

Feed exactly the same physics events through FullSim or SGV.

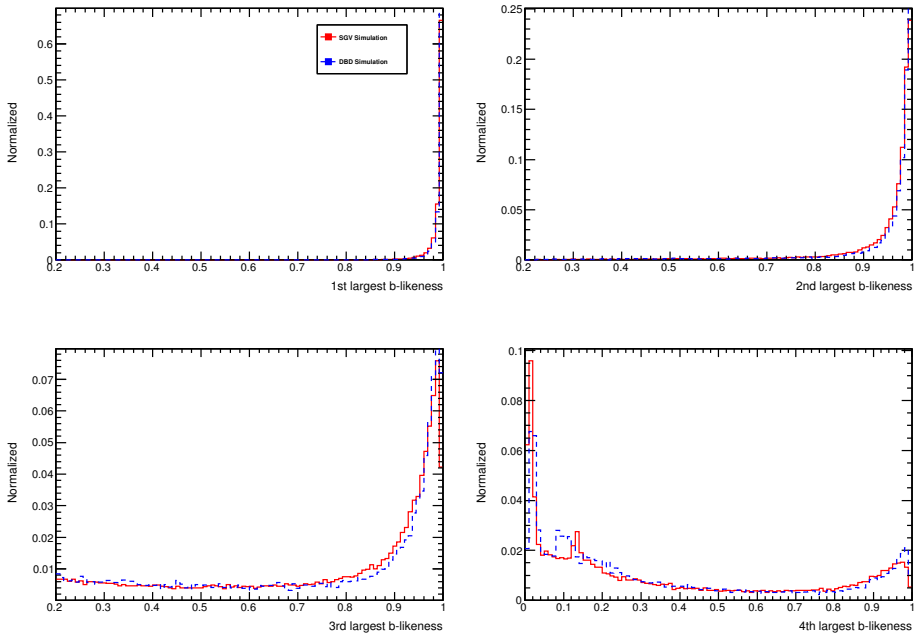
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !



SGV and FullSim ILD: Full system

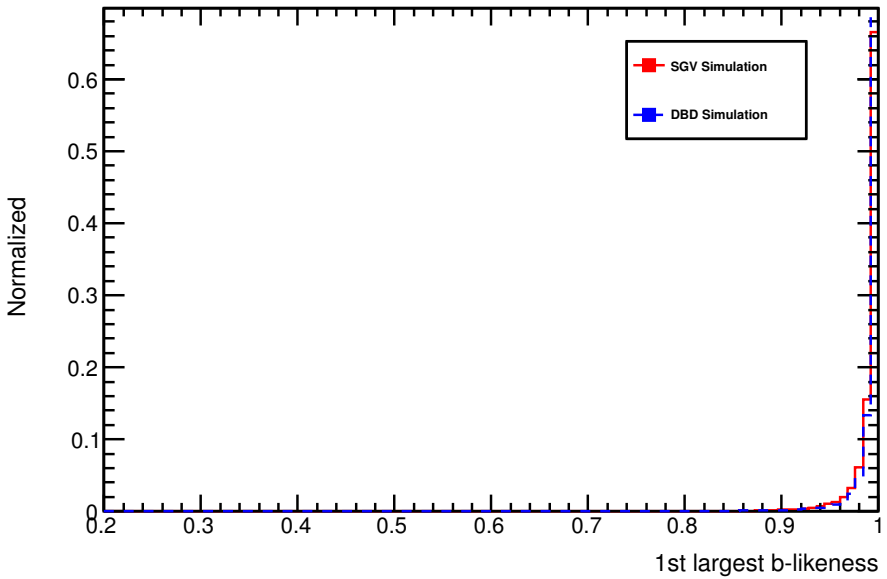
Feed exactly the same physics events through FullSim or SGV.

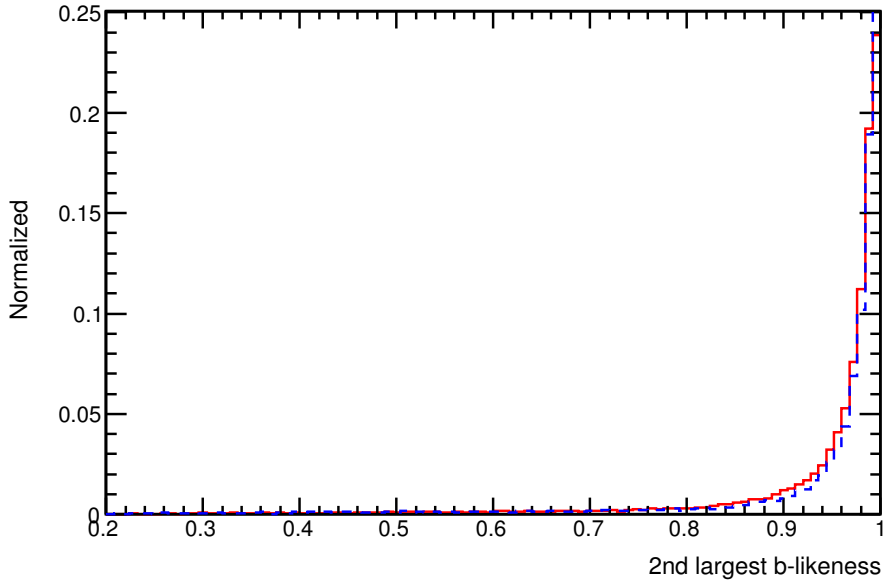
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !

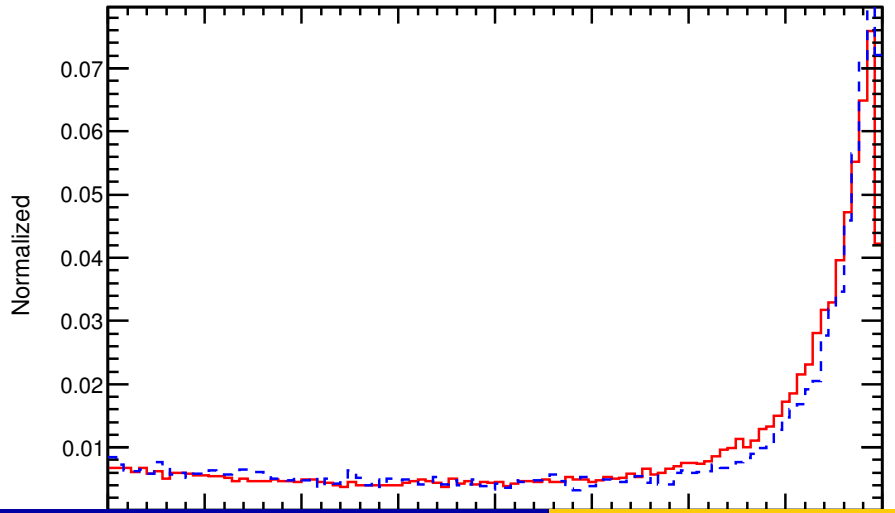


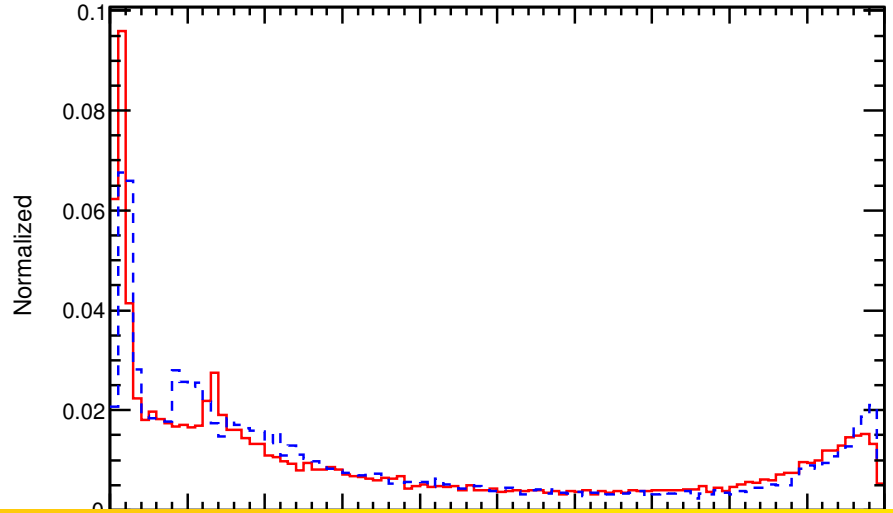
SGV and FullSim ILD: Full system

F





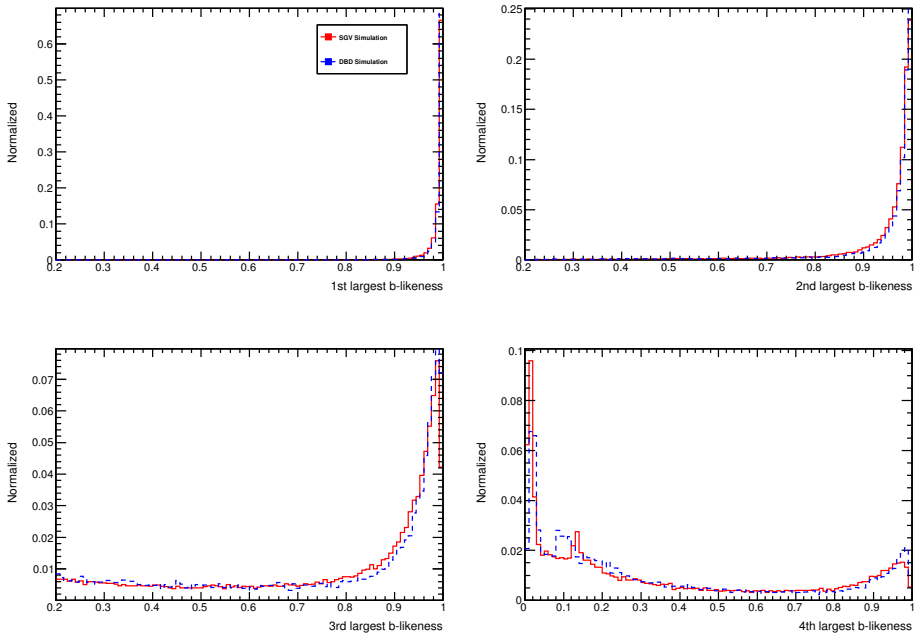




SGV and FullSim ILD: Full system

Feed exactly the same physics events through FullSim or SGV.

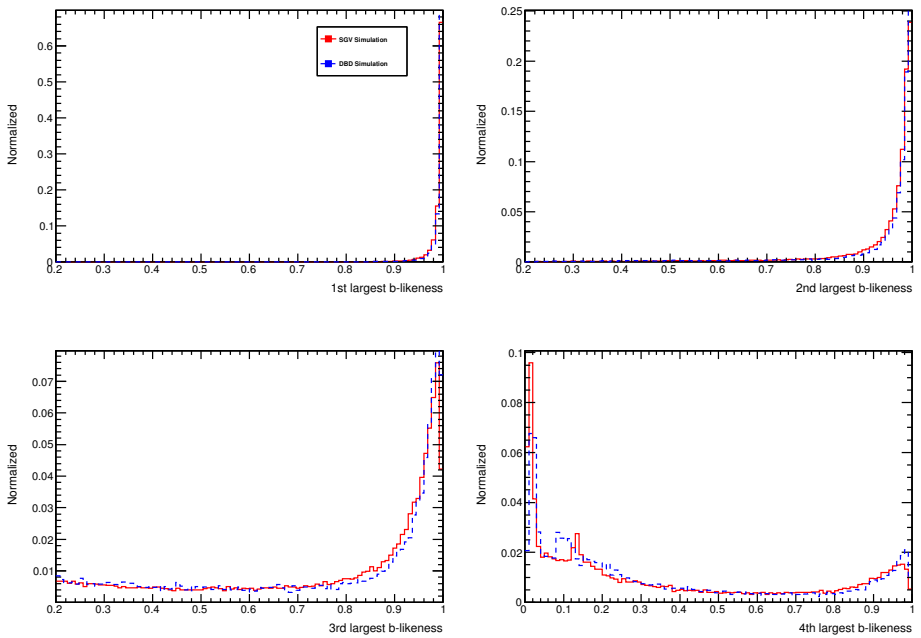
- Overall:
  - Total seen energy
- $e^+e^- \rightarrow ZZ \rightarrow$ four jets:
  - Make "True Jets":
  - Reconstructed M_Z at different stages in FullSim.
  - Seen Reconstructed M_Z, FullSim and SGV.
  - Jet-Energy resolution
- *Zhh* at 1 TeV:
  - Visible E
  - Higgs Mass
  - b-tag
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !



SGV and FullSim ILD: Full system

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
  - Error on Higgs-selfcoupling: SGV 17% , FullSim 18 % !



Near Future developments

- Finish up particle flow parametrisation:
  - Jet mass.
  - MIP signals.
  - Clusters from same true particle in both EM and hadronic calorimeters.
  - Cluster C.O.G (not just start point)
- Handling very large input and/or output files: File splitting etc.
- Handling very large *numbers* input files.
- 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

Near Future developments

- Finish up particle flow parametrisation:
  - Jet mass.
  - MIP signals.
  - Clusters from same true particle in both EM and hadronic calorimeters.
  - Cluster C.O.G (not just start point)
- Handling very large input and/or output files: File splitting etc.
- Handling very large *numbers* input files.
- 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

Near Future developments

- Finish up particle flow parametrisation:
  - Jet mass.
  - MIP signals.
  - Clusters from same true particle in both EM and hadronic calorimeters.
  - Cluster C.O.G (not just start point)
- Handling very large input and/or output files: File splitting etc.
- Handling very large *numbers* input files.
- 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

Near Future developments

- Finish up particle flow parametrisation:
  - Jet mass.
  - MIP signals.
  - Clusters from same true particle in both EM and hadronic calorimeters.
  - Cluster C.O.G (not just start point)
- Handling very large input and/or output files: File splitting etc.
- Handling very large *numbers* input files.
- 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

Not-So-Far 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.

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory, also at the ILC.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) will be explained by Madalina tomorrow, with more case-studies.
- Comparisons to Mokka/Marlin was shown to be quite good.
- SGV mass production works - do $\sim$ 0.7 TEvents in $\mathcal{O}(1)$ hour on NAF. (Details in backup slides)
- Many SGV-based analyses done in ILC/ILD, sometimes SGV only, sometimes mixed SGV-FullSim, sometimes developed with SGV, then done with FullSim when resources became available.

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory, also at the ILC.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) will be explained by Madalina tomorrow, with more case-studies.
- Comparisons to Mokka/Marlin was shown to be quite good.
- SGV mass production works - do $\sim$ 0.7 TEvents in $\mathcal{O}(1)$ hour on NAF. (Details in backup slides)
- Many SGV-based analyses done in ILC/ILD, sometimes SGV only, sometimes mixed SGV-FullSim, sometimes developed with SGV, then done with FullSim when resources became available.

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory, also at the ILC.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) will be explained by Madalina tomorrow, with more case-studies.
- Comparisons to Mokka/Marlin was shown to be quite good.
- SGV mass production works - do $\sim$ 0.7 TEvents in $\mathcal{O}(1)$ hour on NAF. (Details in backup slides)
- Many SGV-based analyses done in ILC/ILD, sometimes SGV only, sometimes mixed SGV-FullSim, sometimes developed with SGV, then done with FullSim when resources became available.

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory, also at the ILC.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) will be explained by Madalina tomorrow, with more case-studies.
- Comparisons to Mokka/Marlin was shown to be quite good.
- SGV mass production works - do $\sim$ 0.7 TEvents in $\mathcal{O}(1)$ hour on NAF. (Details in backup slides)
- Many SGV-based analyses done in ILC/ILD, sometimes SGV only, sometimes mixed SGV-FullSim, sometimes developed with SGV, then done with FullSim when resources became available.

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory, also at the ILC.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The method to emulate the performance of FullReco particle-flow (PandoraPFO) will be explained by Madalina tomorrow, with more case-studies.
- Comparisons to Mokka/Marlin was shown to be quite good.
- SGV mass production works - do $\sim$ 0.7 TEvents in $\mathcal{O}(1)$ hour on NAF. (Details in backup slides)
- Many SGV-based analyses done in ILC/ILD, sometimes SGV only, sometimes mixed SGV-FullSim, sometimes developed with SGV, then done with FullSim when resources became available.

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory, also at the ILC.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.

Installing SGV

svn co https://svnsrv.desy.de/public/sgv/trunk/ sgv/
Then
cd sgv ; . ./install

- SGV mass production works - do ~ 0.7 TEvents in $\mathcal{O}(1)$ hour on NAF. (Details in backup slides)
- Many SGV-based analyses done in ILC/ILD, sometimes SGV only, sometimes mixed SGV-FullSim, sometimes developed with SGV, then done with FullSim when resources became available.

Thank You !

Backup

BACKUP SLIDES

Some internals

Tracking:

- Module zt_tracking:
  - Higher-level functions: *"Particle with momentum $\bar{p}$ starts at $\bar{v}$ in detector i. Return helix-paramters and covariance at point $\bar{r}$"*
  - Initialises constants-of-motion for the track *once*.
  - Decides which helix-parametrisation is best at a given point, and transforms when needed.
  - Checks max and min layer that *could* be hit by particle, to avoid un-nessesary computations.
- Module zt_covaiance: The covariance machine:
  - Geometry worked out analytically.
  - Matrix algebra (inversions, multiplications, ...) solved in component-form.
- Module zt_access: "getters" of detector information.
- Routine zt_ini: Initialisation

Some internals

Tracking:

- Module zt_tracking:
  - Higher-level functions: *"Particle with momentum $\bar{p}$ starts at $\bar{v}$ in detector i. Return helix-paramters and covariance at point $\bar{r}$"*
  - Initialises constants-of-motion for the track *once*.
  - Decides which helix-parametrisation is best at a given point, and transforms when needed.
  - Checks max and min layer that *could* be hit by particle, to avoid un-nessesary computations.
- Module zt_covaiance: The covariance machine:
  - Geometry worked out analytically.
  - Matrix algebra (inversions, multiplications, ...) solved in component-form.
- Module zt_access: "getters" of detector information.
- Routine zt_ini: Initialisation

Some internals

Tracking:

- Module zt_tracking:
  - Higher-level functions: *"Particle with momentum $\bar{p}$ starts at $\bar{v}$ in detector i. Return helix-paramters and covariance at point $\bar{r}$"*
  - Initialises constants-of-motion for the track *once*.
  - Decides which helix-parametrisation is best at a given point, and transforms when needed.
  - Checks max and min layer that *could* be hit by particle, to avoid un-nessesary computations.
- Module zt_covaiance: The covariance machine:
  - Geometry worked out analytically.
  - Matrix algebra (inversions, multiplications, ...) solved in component-form.
- Module zt_access: "getters" of detector information.
- Routine zt_ini: Initialisation

Some internals

Detector simulation:

- Generated event should be loaded in Pythia:s /PYJETS/.
- Decide if particle is:
  - "Seeable", ie. if it possibly could reach any detector element.
  - "Storable", ie. if the user asked to keep it even is it is not "Seeable". By default: All stable particles + all particles with fly > 1 cm .

  This reduces the size of loops later.
- Apply B-field: move decay verticies, tilt $\bar{p}$ of decay products.
- Loop "seeable" twice:
  - First: simulate brems and pair-creation; remove daugthers of particles that hit a calorimeter before decaying.
  - Second: actually simulate the responce.
    - Tracking: Call tracking to get unsmeared helix and covariance, then smear, which smears it using Cholesky decomposition method.
    - Calorimetry: Call ... which calls ... to get path of/coordinates of trajectory intersection points with calo ... parametrically.

Some internals

Detector simulation:

- Generated event should be loaded in Pythia:s /PYJETS/.
- Decide if particle is:
  - “Seeable”, ie. if it possibly could reach any detector element.
  - “Storable”, ie. if the user asked to keep it even is it is not “Seeable”.
    By default: All stable particles + all particles with fly $>$ 1 cm .

  This reduces the size of loops later.
- Apply B-field: move decay verticies, tilt $\bar{p}$ of decay products.
- Loop “seeable” twice:
  - First: simulate brems and pair-creation; remove daugthers of particles that hit a calorimeter before decaying.
  - Second: actually simulate the responce.

Some internals

Detector simulation:

- Generated event should be loaded in Pythia:s /PYJETS/.
- Decide if particle is:
  - "Seeable", ie. if it possibly could reach any detector element.
  - "Storable", ie. if the user asked to keep it even is it is not "Seeable". By default: All stable particles + all particles with fly > 1 cm .

  This reduces the size of loops later.
- Apply B-field: move decay verticies, tilt $\bar{p}$ of decay products.
- Loop "seeable" twice:
  - First: simulate brems and pair-creation; remove daugthers of particles that hit a calorimeter before decaying.
  - Second: actually simulate the responce.
    - Tracking: Call tracking to get unsmeared helix and cov. mat , then zdsmtp, which smears it using Choleski decomposition method.
    - Calorimeter: Call zdcalo, which calls ztcalo to get path-ordered list of trajectory intersection points with calo:s. Response is then simulated paramtrically.

Some internals

Detector simulation:

- Generated event should be loaded in Pythia:s /PYJETS/.
- Decide if particle is:

Performance considerations

- Important issues in red.
- During f77 $\rightarrow$ Fortran95 transistion:
  - Avoid feature rot: Adding a cool feature that only add 10 % to the CPU time is OK, right ?
    - Well, yes, but ...
    - ... if you do that every two weeks ...
    - ... yor code runs 10 times slower after 1 year...
    - ... 100 times slower after 2 years ...
    - ... and 1000 slower times after 3 years !

zdsmtp, which smears it using Choleski decomposition method.

- Calorimeter: Call zdcalo, which calls ztcalo to get path-ordered list of trajectory intersection points with calo:s. Response is then simulated paramtrically.

Some internals

Detector simulation:

- Generated event should be loaded in Pythia:s /PYJETS/.
- Decide if particle is:

Performance considerations

- Important issues in red.
- During f77 $\rightarrow$ Fortran95 transistion:
  - Avoid feature rot: Adding a cool feature that only add 10 % to the CPU time is OK, right ?
    - Well, yes, but ...
    - ... if you do that every two weeks ...
    - ... yor code runs 10 times slower after 1 year...
    - ... 100 times slower after 2 years ...
    - ... and 1000 slower times after 3 years !

zdsmtp, which smears it using Choleski decomposition method.

- Calorimeter: Call zdcalo, which calls ztcalo to get path-ordered list of trajectory intersection points with calo:s. Response is then simulated paramtrically.

Some internals

Detector simulation:

- Generated event should be loaded in Pythia:s /PYJETS/.
- Decide if particle is:

Performance considerations

- Important issues in red.
- During f77 $\rightarrow$ Fortran95 transistion:
  - Avoid feature rot: Adding a cool feature that only add 10 % to the CPU time is OK, right ?
    - Well, yes, but ...
    - ... if you do that every two weeks ...
    - ... yor code runs 10 times slower after 1 year...
    - ... 100 times slower after 2 years ...
    - ... and 1000 slower times after 3 years !

zdsmtp, which smears it using Choleski decomposition method.

- Calorimeter: Call zdcalo, which calls ztcalo to get path-ordered list of trajectory intersection points with calo:s. Response is then simulated paramtrically.

Some internals

Detector simulation:

- Generated event should be loaded in Pythia:s /PYJETS/.
- Decide if particle is:

Performance considerations

- Important issues in red.
- During f77 $\rightarrow$ Fortran95 transistion:
  - Avoid feature rot: Adding a cool feature that only add 10 % to the CPU time is OK, right ?
    - Well, yes, but ...
    - ... if you do that every two weeks ...
    - ... yor code runs 10 times slower after 1 year...
    - ... 100 times slower after 2 years ...
    - ... and 1000 slower times after 3 years !

zdsmtp, which smears it using Choleski decomposition method.

- Calorimeter: Call zdcalo, which calls ztcalo to get path-ordered list of trajectory intersection points with calo:s. Response is then simulated paramtrically.

Some internals

Detector simulation:

- Generated event should be loaded in Pythia:s /PYJETS/.
- Decide if particle is:

Performance considerations

- Important issues in red.
- During f77 $\rightarrow$ Fortran95 transistion:
  - Avoid feature rot: Adding a cool feature that only add 10 % to the CPU time is OK, right ?
    - Well, yes, but ...
    - ... if you do that every two weeks ...
    - ... yor code runs 10 times slower after 1 year...
    - ... 100 times slower after 2 years ...
    - ... and 1000 slower times after 3 years !

zdsmtp, which smears it using Choleski decomposition method.

- Calorimeter: Call zdcalo, which calls ztcalo to get path-ordered list of trajectory intersection points with calo:s. Response is then simulated paramtrically.

Some internals

Detector simulation:

- Generated event should be loaded in Pythia:s /PYJETS/.
- Decide if particle is:

Performance considerations

- Important issues in red.
- During f77 $\rightarrow$ Fortran95 transistion:
  - Avoid feature rot: Adding a cool feature that only add 10 % to the CPU time is OK, right ?
    - Well, yes, but ...
    - ... if you do that every two weeks ...
    - ... yor code runs 10 times slower after 1 year...
    - ... 100 times slower after 2 years ...
    - ... and 1000 slower times after 3 years !

zdsmtp, which smears it using Choleski decomposition method.

- Calorimeter: Call zdcalo, which calls ztcalo to get path-ordered list of trajectory intersection points with calo:s. Response is then simulated paramtrically.

Use-cases at the ILC

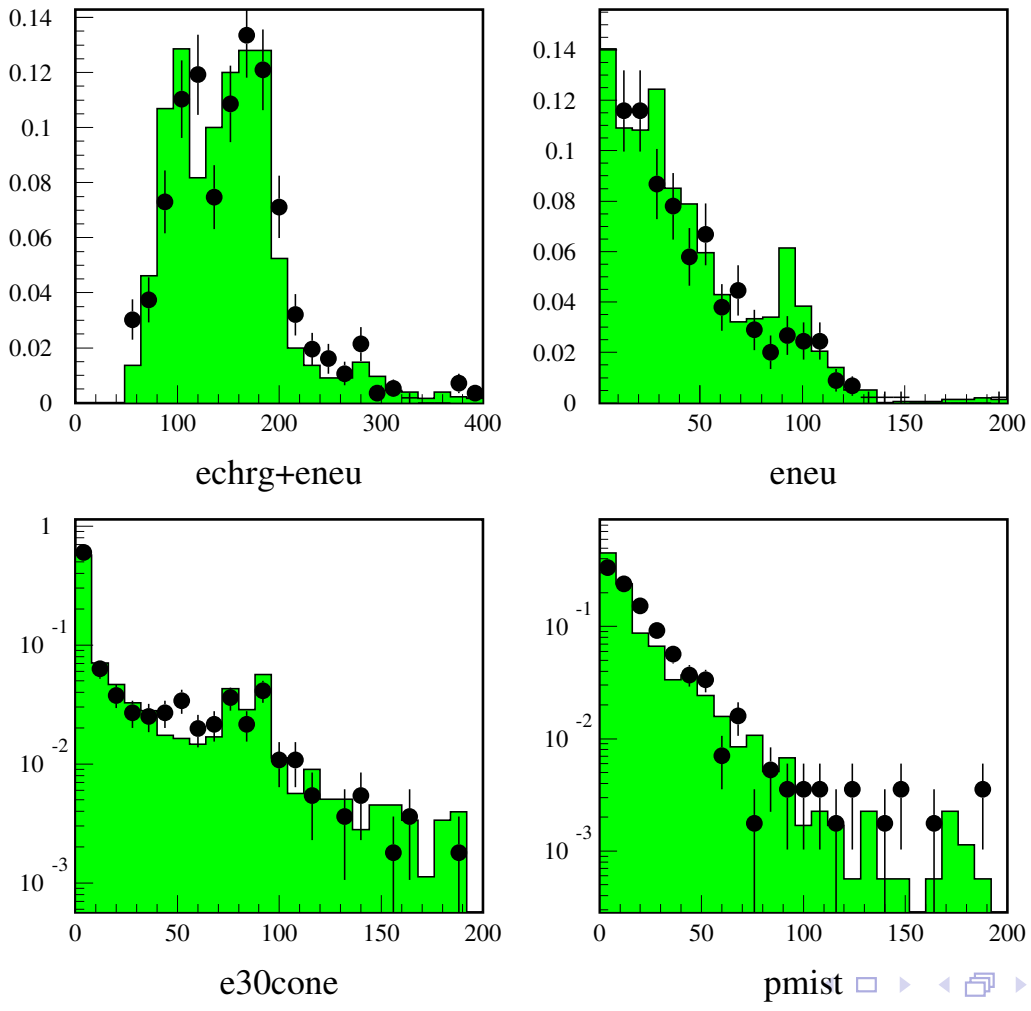
- Used for fastsim physics studies, eg. arXiv:hep-ph/0510088, arXiv:hep-ph/0508247, arXiv:hep-ph/0406010, arXiv:hep-ph/9911345 and arXiv:hep-ph/9911344.
- Used for flavour-tagging training.
- Used for overall detector optimisation, see Eg. Vienna ECFA WS (2007), See Ilcagenda > Conference and Workshops > 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 physics performance

SGV and real data from DELPHI: Global variables

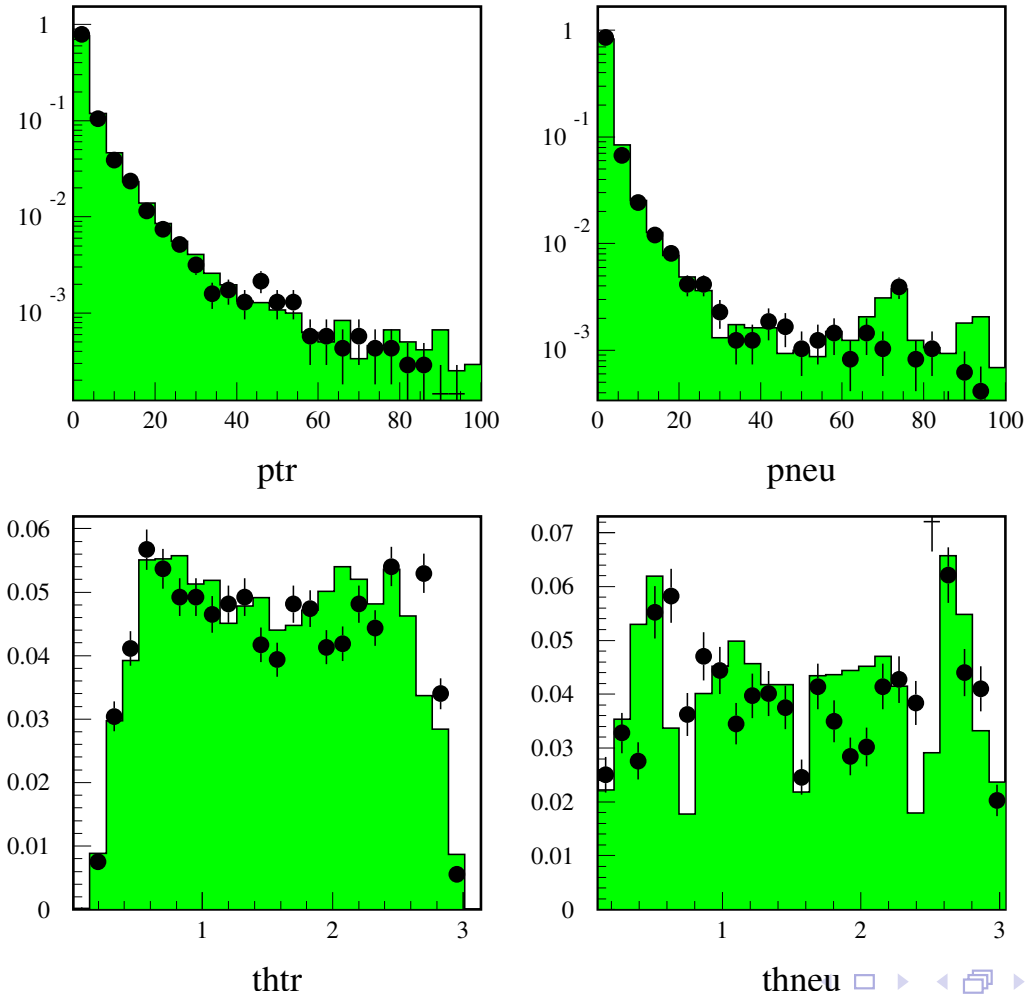
Histogram: SGV, Points: DELPHI data



SGV physics performance: DELPHI

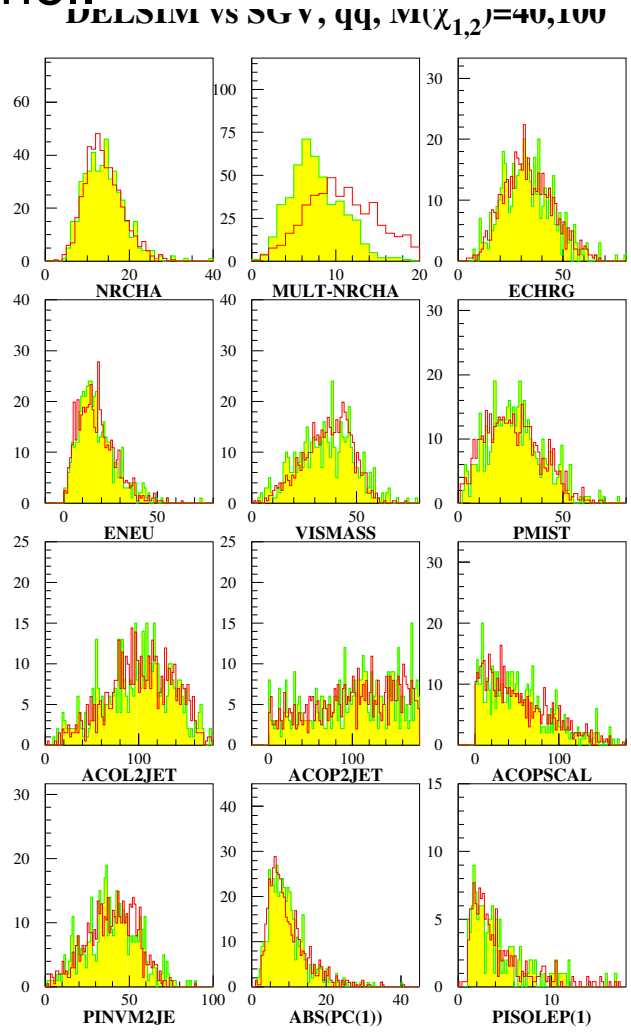
SGV and real data from DELPHI: Particle variables

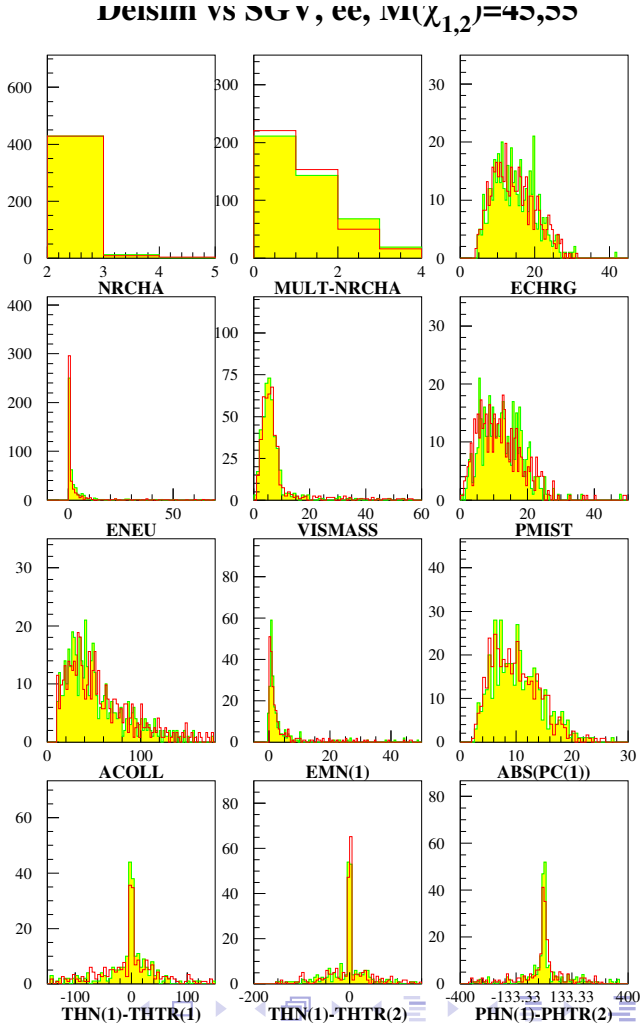
Histogram: SGV, Points: DELPHI data



SGV physics performance: DELPHI

SGV and full simulation from DELPHI: Neutralino search in hadronic channel.



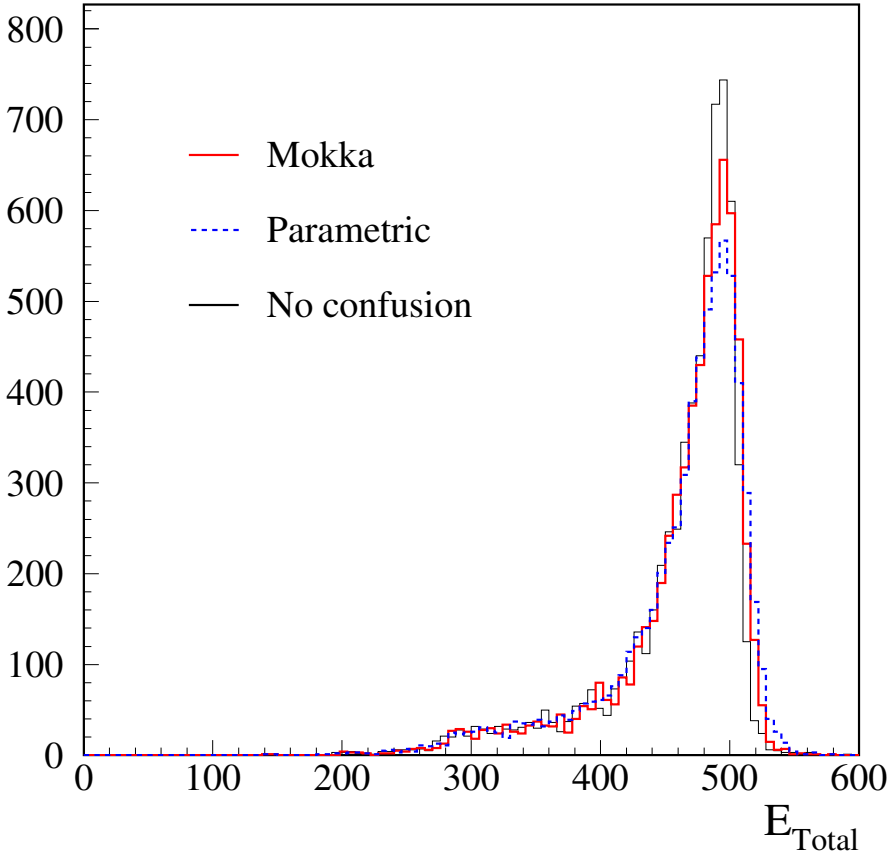


Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag

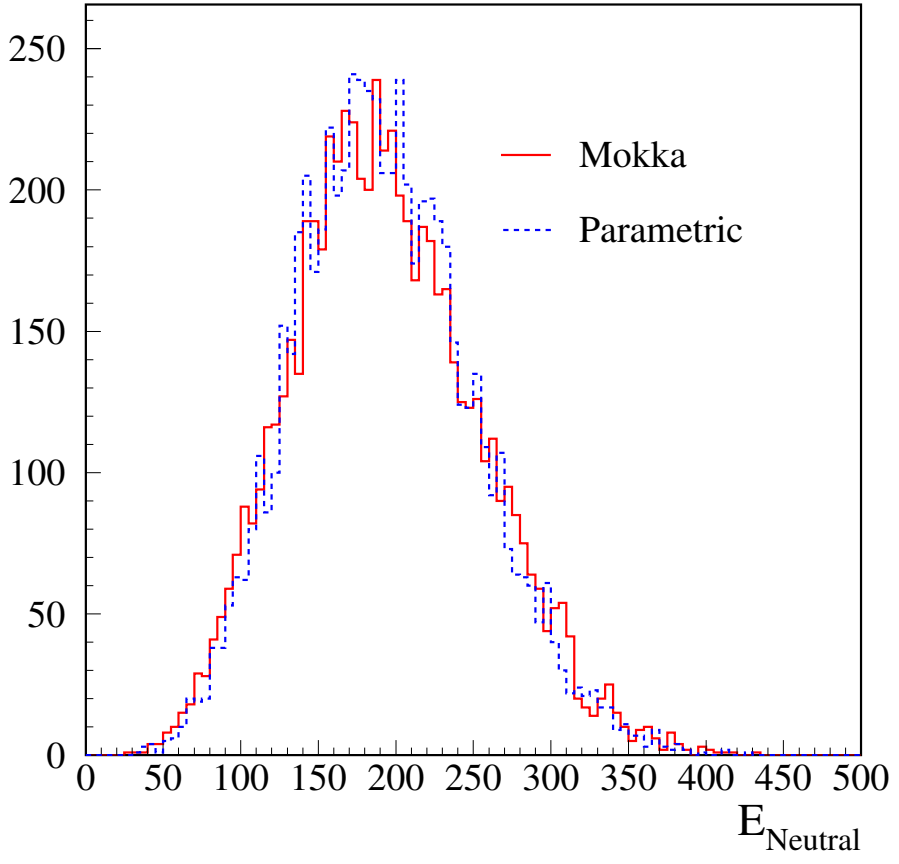
Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag



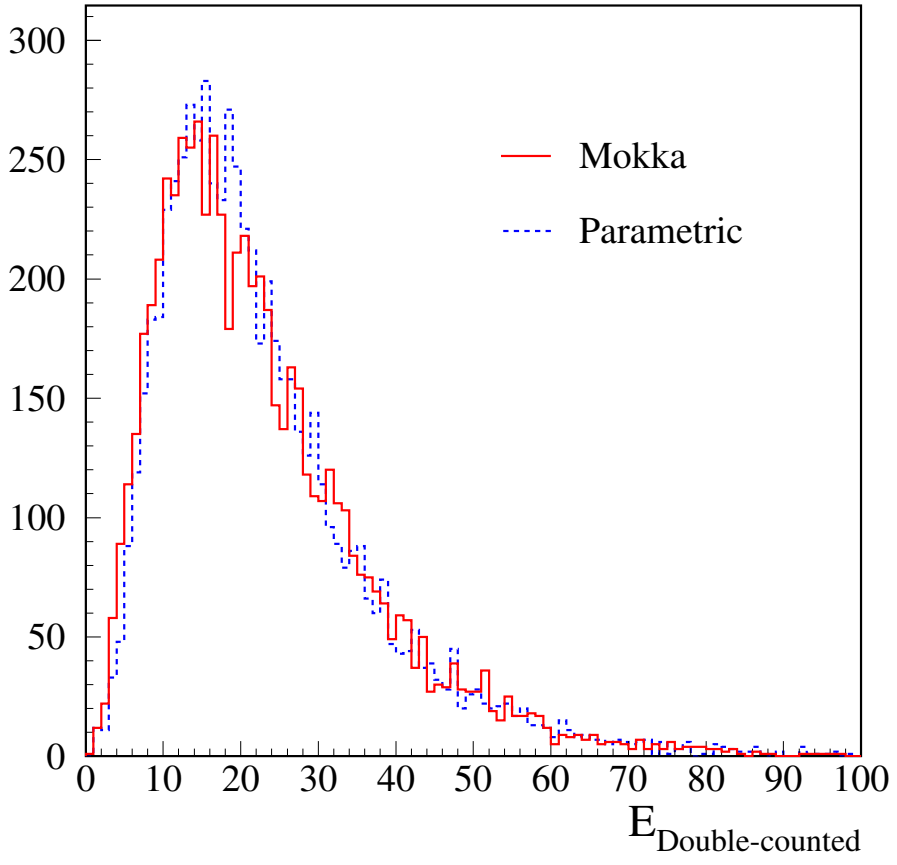
Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag



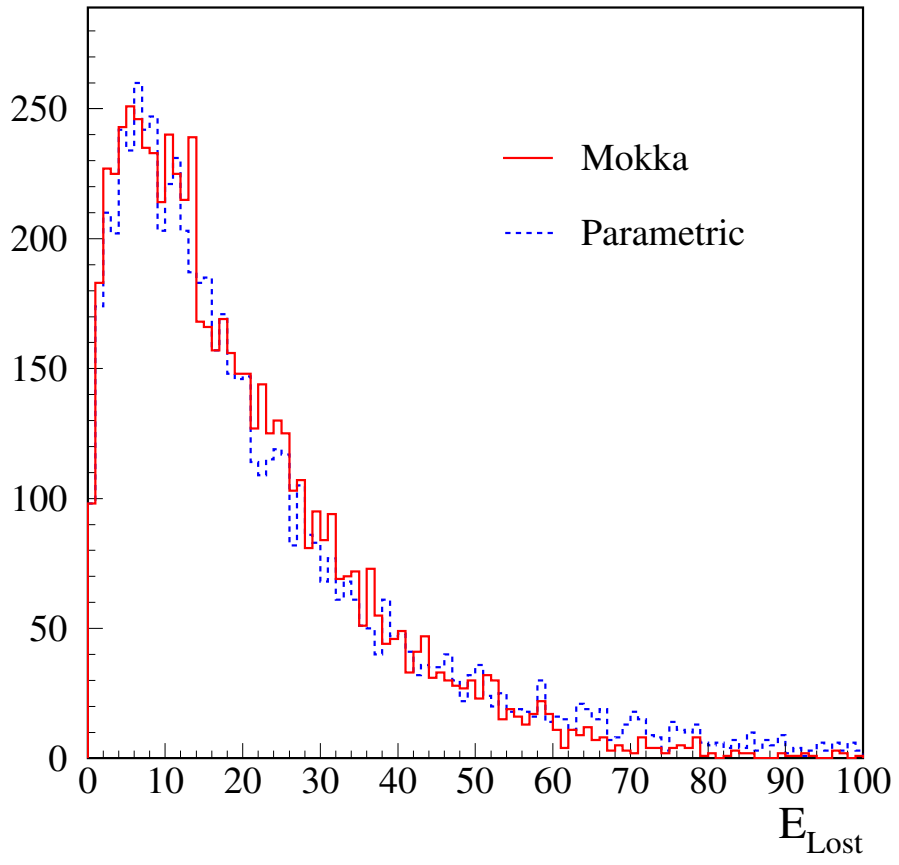
Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag



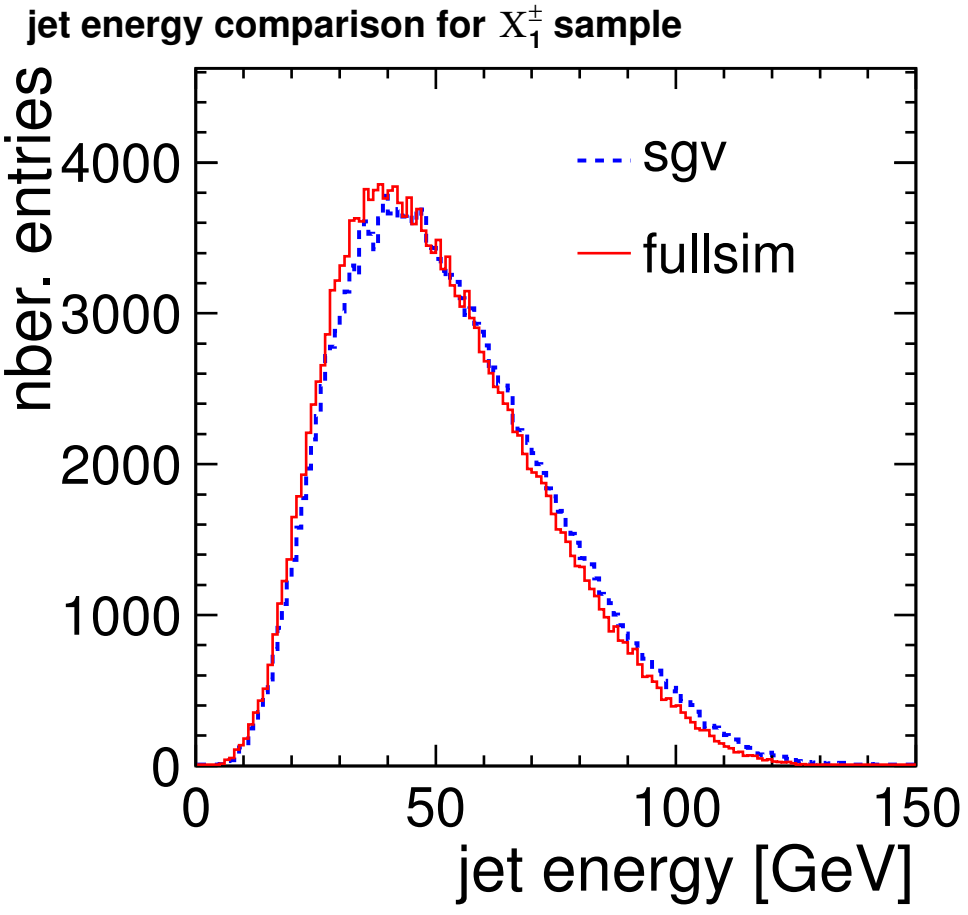
Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag



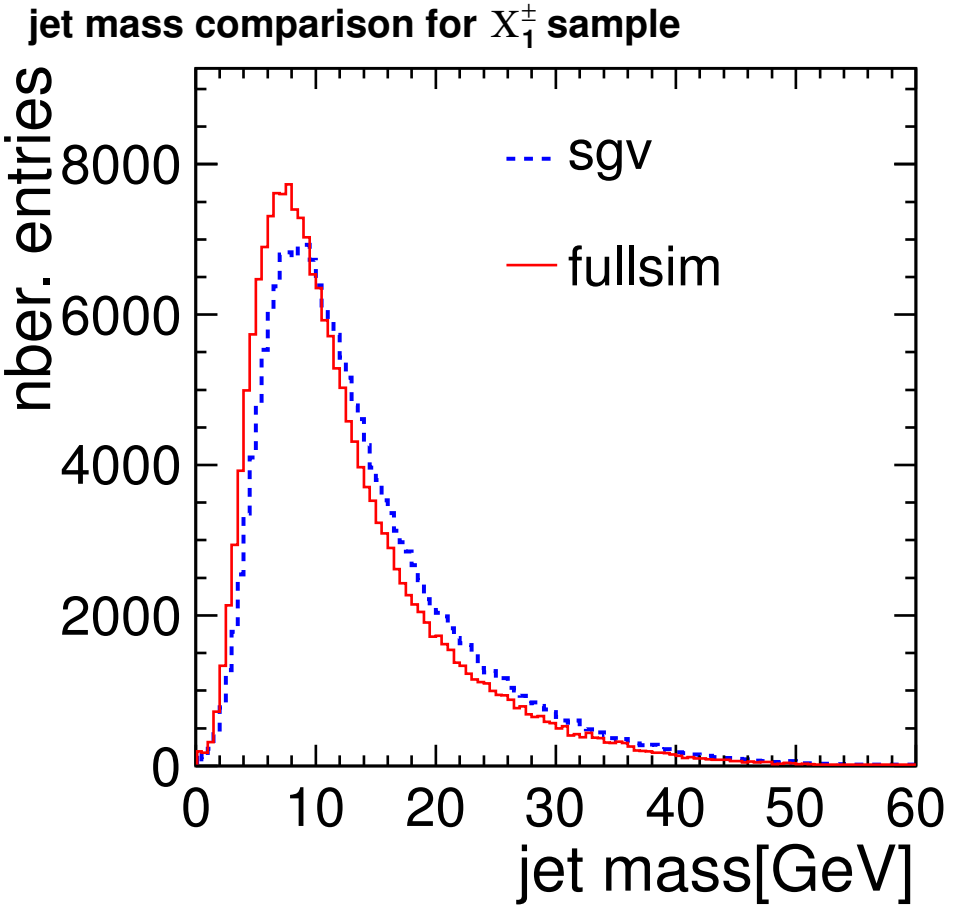
Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag



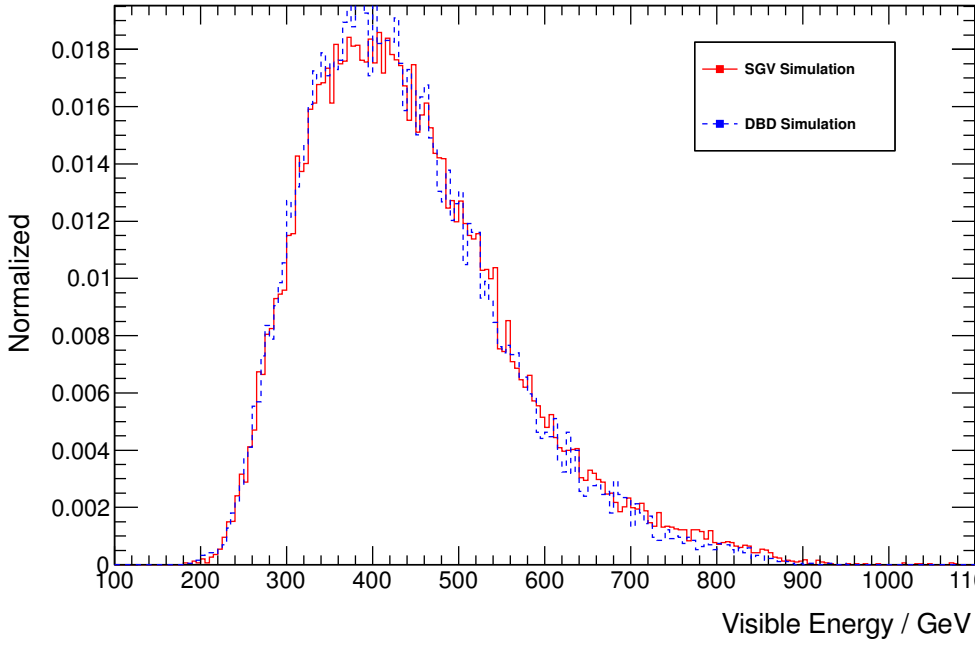
Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag



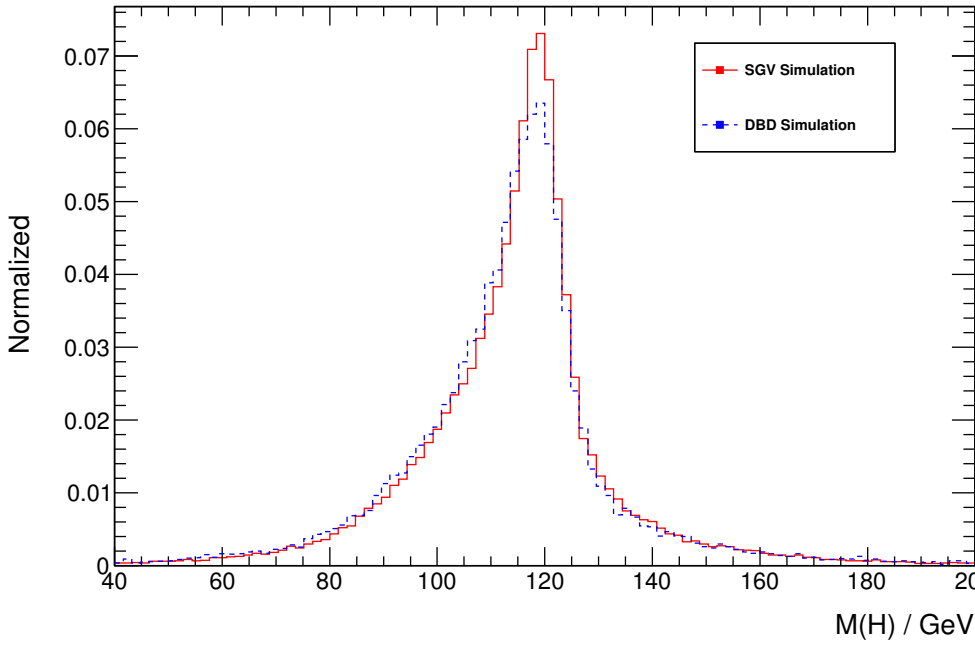
Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag



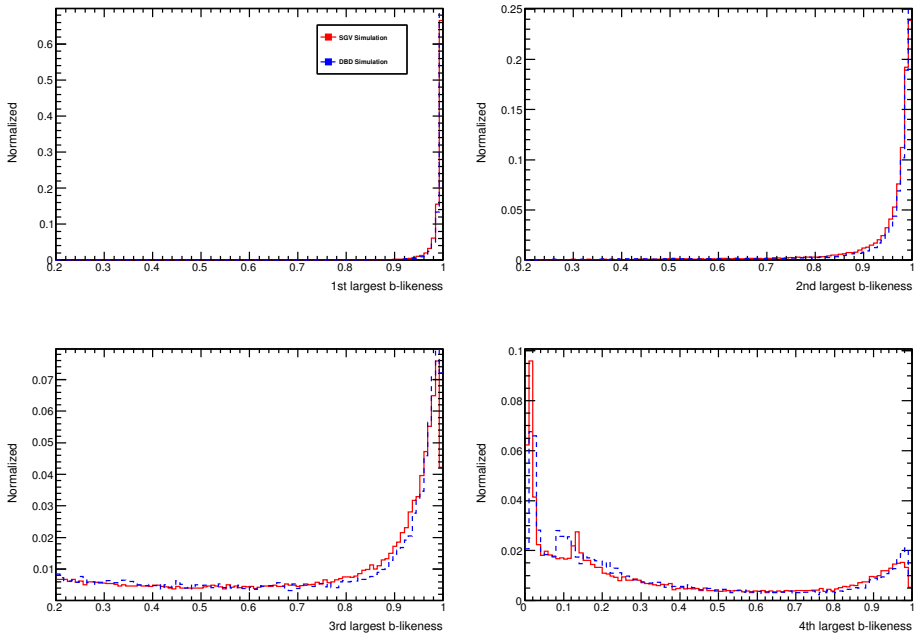
Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag

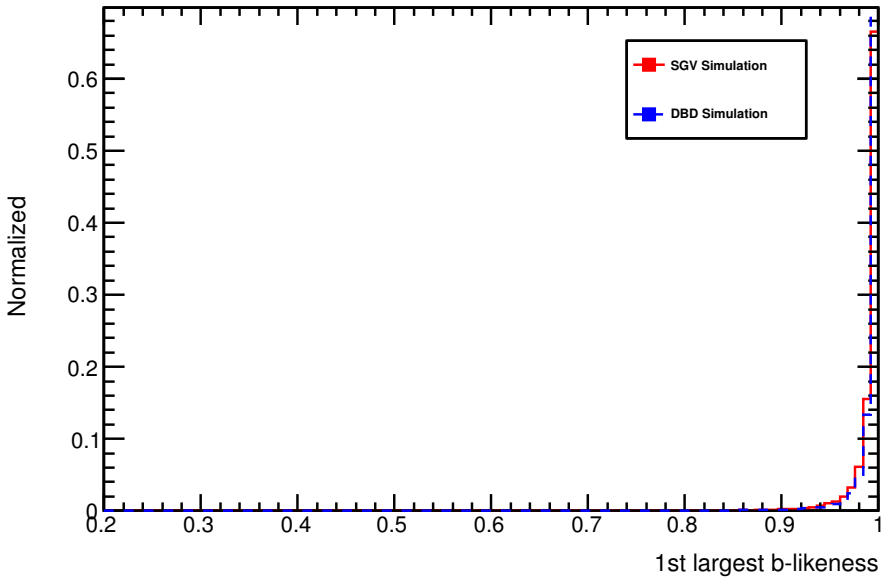


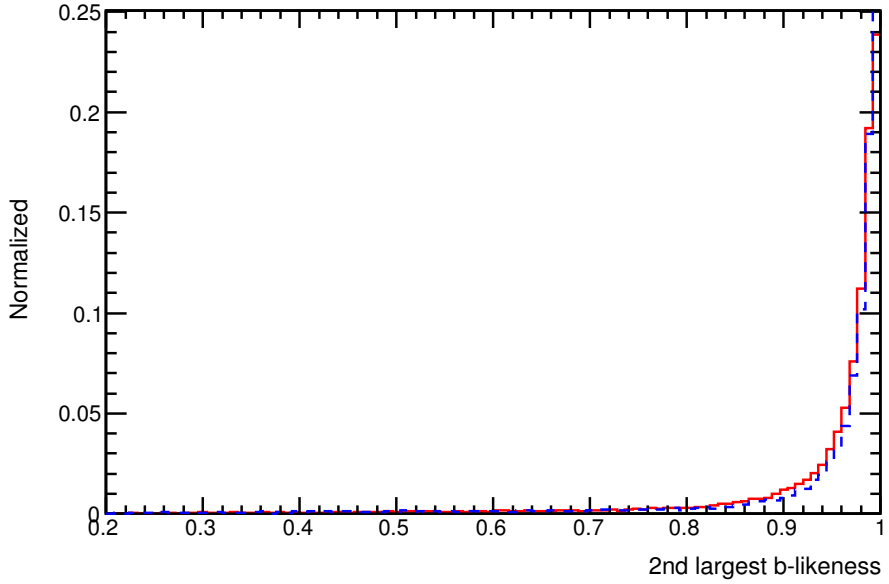
Checking the parametrisation

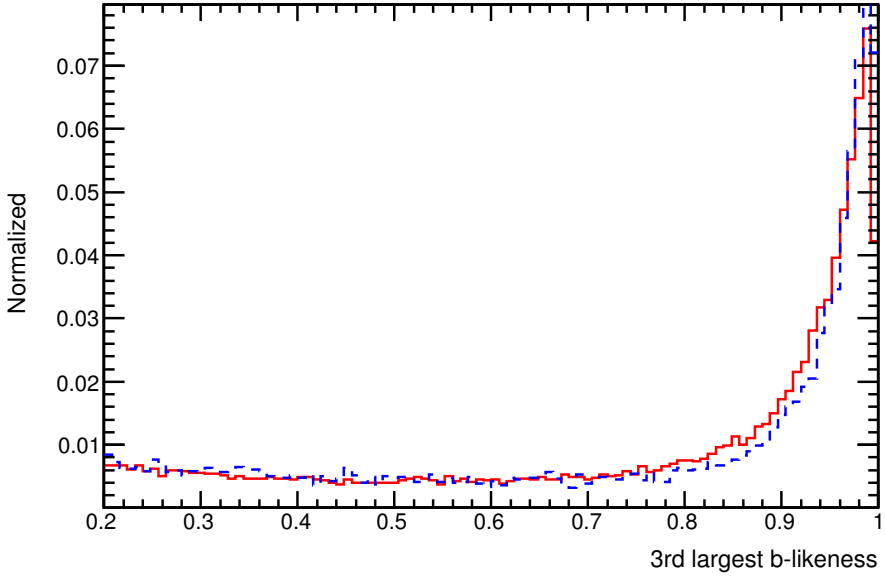
- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag

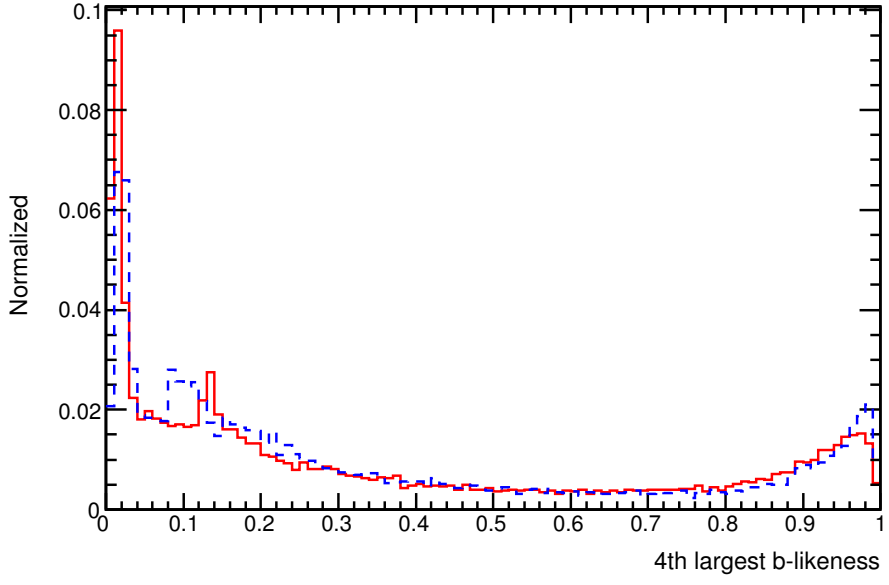


C


SGV Simulation
DBD Simulation
Normalized
1st largest b-likeness

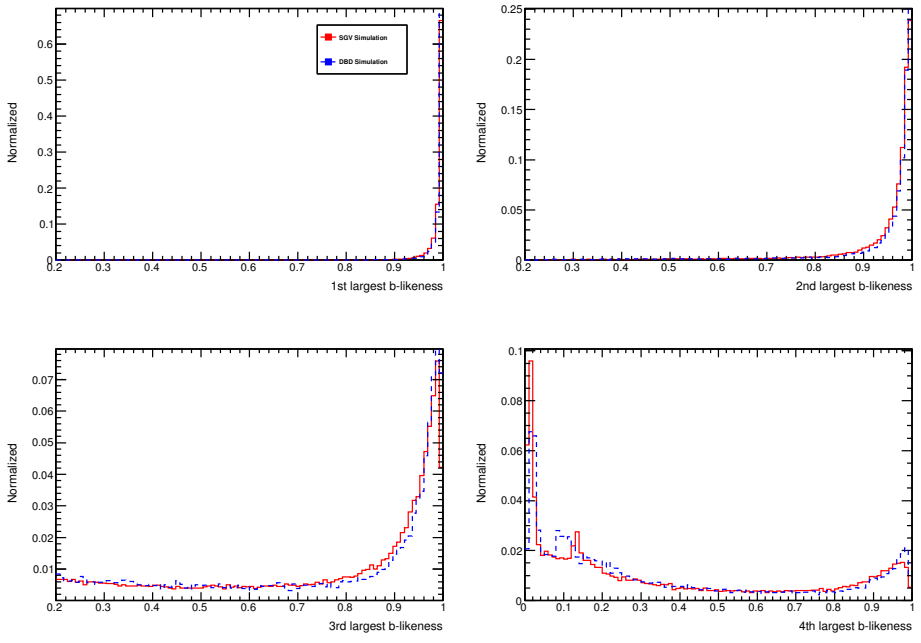

Normalized
2nd largest b-likeness


Normalized
3rd largest b-likeness


Normalized
4th largest b-likeness

Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet properties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
- *Zhh* at 1 TeV:
  - Vissible E
  - Higgs Mass
  - b-tag



LCIO DST mass-production

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

- *usesgvlcio.F95* in the *samples/lcio* directory extracts SGV data and fills all LCIO collections on the ILD DST:s
  - Clusters:
    - Are done with the Pandora confusion parametrisation on.
    - Expect $\sim$ correct dispersion of jet energy, but a few % to high central value of jet masses.
  - Navigators
    - All the navigators that the TruthLinker processor makes when all flags are switched on are created.
  - Secondary vertices:
    - Use true information to find all secondary vertices.
    - For all vertices with ≥ 2 seen charged tracks: do vertex fit.
    - Expect $\sim$ correct vertex fit-parameters, but too good vertex finding.

Mass production

- Done almost the full DBD samples - several times.
  - 34 Mevents.
  - $\sim$ 1 hour of wall-clock time (first submit to last completed) on the German NAF.
- $\gamma\gamma$ still missing: Logistics to figure out (Many thousands of input files !)

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.
- Consequence:
  - 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.