New developments in SGV - a fast detector simulation

Mikael Berggren[1]

[1]DESY, Hamburg

LCWS12, Arlington, TX, Oct 2012

Outline

$\gamma\gamma$ background

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

- $\int \mathcal{L}dt$ = 500 fb^{-1} $\rightarrow$ 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 $e^+e^- \rightarrow \gamma\gamma e^+e^- \rightarrow q\bar{q}e^+e^-$: 35 nb (PYTHIA)

- $\int \mathcal{L}dt$ = 500 fb^{-1} $\rightarrow$ 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 $e^+e^- \rightarrow \gamma\gamma e^+e^- \rightarrow q\bar{q}e^+e^-$: 35 nb (PYTHIA)

- $\int \mathcal{L}dt$ = 500 fb^{-1} $\rightarrow$ 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...

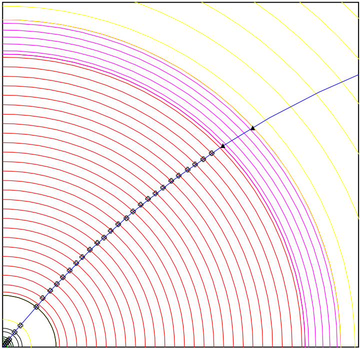
Slower to generate and simulate than $\gamma\gamma$ events

Also here: CPU millenniums with full simulation

SGV: How tracking works

"la Simulation à Grande Vitesse", SGV. is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.

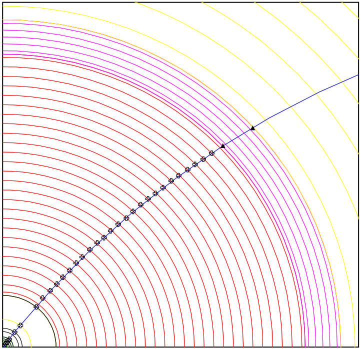


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Information on hits accessible to analysis.

SGV: How tracking works

"la Simulation à Grande Vitesse", SGV. is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.

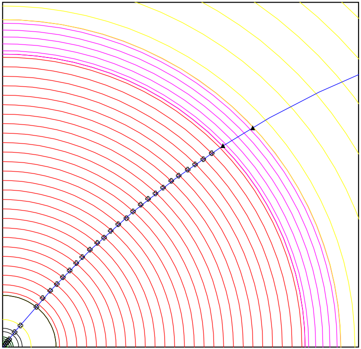


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Information on hits accessible to analysis.

SGV: How tracking works

"la Simulation à Grande Vitesse", SGV. is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.

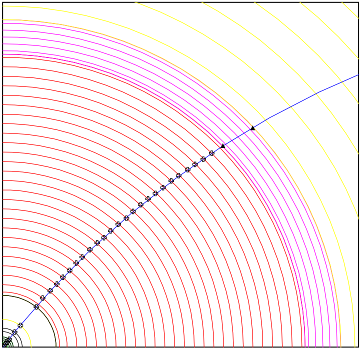


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Information on hits accessible to analysis.

SGV: How tracking works

"la Simulation à Grande Vitesse", SGV. is a machine to calculate covariance matrices

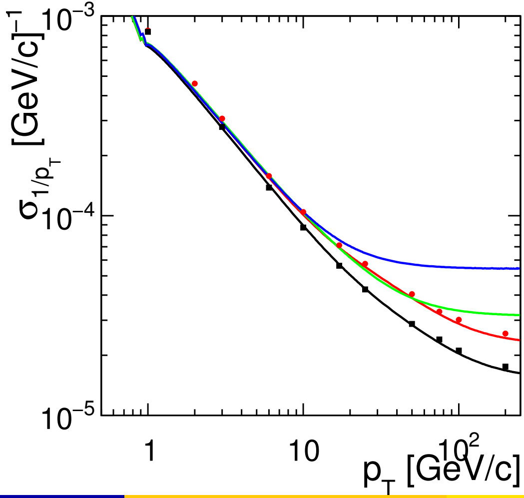
Tracking: Follow track-helix through the detector.



- 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)
- Information on hits accessible to analysis.

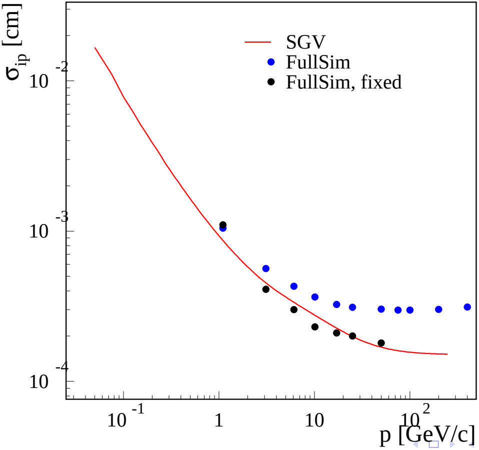
SGV and FullSim LDC/ILD: momentum resolution

Lines: SGV, dots: Mokka+Marlin



SGV and FullSim LDC/ILD: ip resolution vs P

Lines: SGV, dots: Mokka+Marlin



SGV: How the rest works

Calorimeters:

- Follow particle to intersection with calorimeters. Simulate:
  - Response type: MIP, EM-shower, hadronic shower, below threshold, etc.
  - Simulate response from parameters.

Other stuff:

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

SGV: How the rest works

Calorimeters:

- Follow particle to intersection with calorimeters. Simulate:
  - Response type: MIP, EM-shower, hadronic shower, below threshold, etc.
  - Simulate response from parameters.

Other stuff:

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

SGV: Technicalities

Features:

- Written in Fortran 95: 20k lines + 10k lines of comments.
  - Some CERNLIB dependence.
  - Re-write of battle-tested f77 SGV 2-series (LEP, Tesla, LOI, ...)
- Managed in SVN.Install script included.
- Callable PYTHIA, Whizard or input from PYJETS or stdhep.
- Using PYTHIA, including beam spectrum and and/or SUSY is just a question of steering-settings.
- Output of generated event to PYJETS or stdhep.
- samples subdirectory with READMEs, steering and code.
- output LCIO DST.
- Typical generation+simulation+reconstruction time $\mathcal{O}(10)$ ms.
- Timing verified to be faster (by 15%) than the f77 version.

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

Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Will depend on Energy, on distance to neighbor, 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 assoiated with wrong track ....) are of less importance.

Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Will depend on Energy, on distance to neighbor, 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 assoiated with wrong track ....) are of less importance.

Calorimeter simulation

The issues:

- Clearly: Random E, shower position, shower shape.
- But also association errors:
  - Clusters might merge.
  - Clusters might split.
  - Clusters might get wrongly associated to tracks.
- Will depend on Energy, on distance to neighbor, 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 assoiated with wrong track ....) are of less importance.

Tuning to Mokka+Marlin

- Use LOI sample (8k udsc), re-reconstructed with PandoraNew. Compare PandoraPFO:s to MCParticles.
- Replace SGV:s detector simulation by Mokka:
  - From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
  - 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.
  - Looking at how Pandora has associated tracks and clusters: link MCParticle -> Track and/or true cluster -> Seen cluster.
  - Analyse this in SGV.

Tuning to Mokka+Marlin

- Use LOI sample (8k udsc), re-reconstructed with PandoraNew. Compare PandoraPFO:s to MCParticles.
- Replace SGV:s detector simulation by Mokka:
  - From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
  - 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.
  - Looking at how Pandora has associated tracks and clusters: link MCParticle -> Track and/or true cluster -> Seen cluster.
  - Analyse this in SGV.

Tuning to Mokka+Marlin

- Use LOI sample (8k udsc), re-reconstructed with PandoraNew. Compare PandoraPFO:s to MCParticles.
- Replace SGV:s detector simulation by Mokka:
  - From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
  - 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.
  - Looking at how Pandora has associated tracks and clusters: link MCParticle -> Track and/or true cluster -> Seen cluster.
  - Analyse this in SGV.

Tuning to Mokka+Marlin

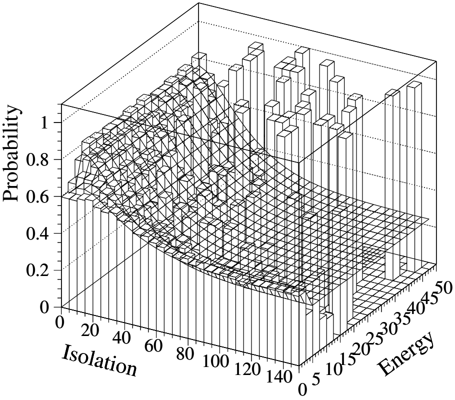
- Use LOI sample (8k udsc), re-reconstructed with PandoraNew. Compare PandoraPFO:s to MCParticles.
- Replace SGV:s detector simulation by Mokka:
  - From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
  - Create true clusters:
    - Each MCParticle is connected to a set of clusters made of CalorimeterHits created by this true particle only.
    - Each of these clusters contribute to only one Pandora cluster.
  - Looking at how Pandora has associated tracks and clusters: link MCParticle -> Track and/or true cluster -> Seen cluster.
  - Analyse this in SGV.

Tuning to Mokka+Marlin

- Use LOI sample (8k udsc), re-reconstructed with PandoraNew. Compare PandoraPFO:s to MCParticles.
- Replace SGV:s detector simulation by Mokka:
  - From LCIO : MCParticles, Tracks, CalorimeterHits, Clusters, PFO's.
  - 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.
  - Looking at how Pandora has associated tracks and clusters: link MCParticle -> Track and/or true cluster -> Seen cluster.
  - Analyse this in SGV.

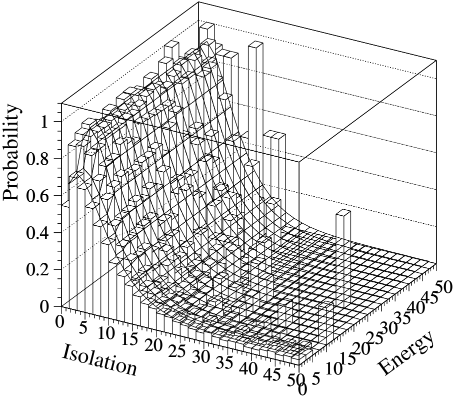
Observed distributions

- 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.



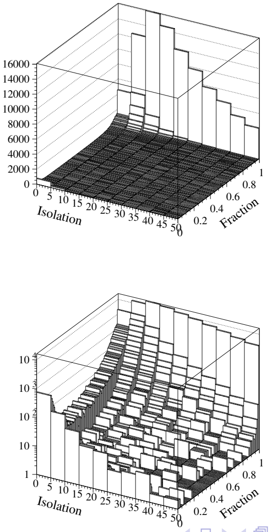
Observed distributions

- 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.



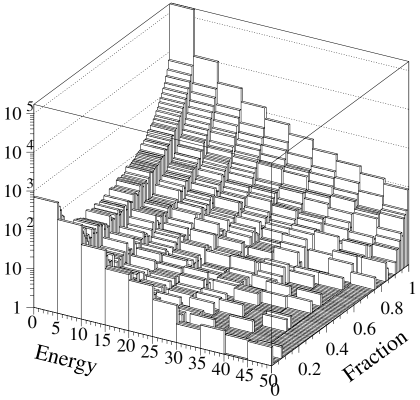
Observed distributions

- 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.



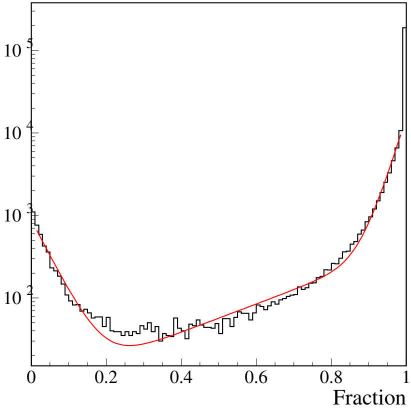
Observed distributions

- 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.



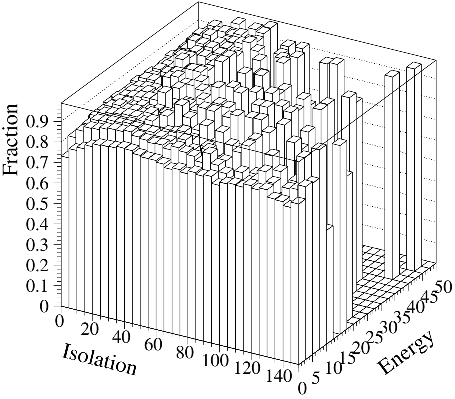
Observed distributions

- 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.



Observations

- Identify and factorise:
  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.
- Observations:
  1. Depnds on the isolation - strongly for merging, slightly for splitting - but can be treated in two energy bins with no energy dependence in the bin. %5 over-all dependence on barrel/endcap.
  2. Depends only on energy. Is small for splitting, important for merging at low E.
  3. Depends on both energy and isolation (very little for splitting), but only via the average.
- 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 × 4 cases (em/had × double-counting/loss)

Observations

- Identify and factorise:
  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.
- Observations:
  1. Depnds on the isolation - strongly for merging, slightly for splitting - but can be treated in two energy bins with no energy dependence in the bin. %5 over-all dependence on barrel/endcap.
  2. Depends only on energy. Is small for splitting, important for merging at low E.
  3. Depends on both energy and isolation (very little for splitting), but only via the average.
- 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 × 4 cases (em/had × double-counting/loss)

Observations

- Identify and factorise:
  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.
- Observations:
  1. Depnds on the isolation - strongly for merging, slightly for splitting - but can be treated in two energy bins with no energy dependence in the bin. %5 over-all dependence on barrel/endcap.
  2. Depends only on energy. Is small for splitting, important for merging at low E.
  3. Depends on both energy and isolation (very little for splitting), but only via the average.
- 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 × 4 cases (em/had × double-counting/loss)

Observations

- Identify and factorise:
  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.
- Observations:
  1. Depnds on the isolation - strongly for merging, slightly for splitting - but can be treated in two energy bins with no energy dependence in the bin. %5 over-all dependence on barrel/endcap.
  2. Depends only on energy. Is small for splitting, important for merging at low E.
  3. Depends on both energy and isolation (very little for splitting), but only via the average.
- 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 × 4 cases (em/had × double-counting/loss)

Observations

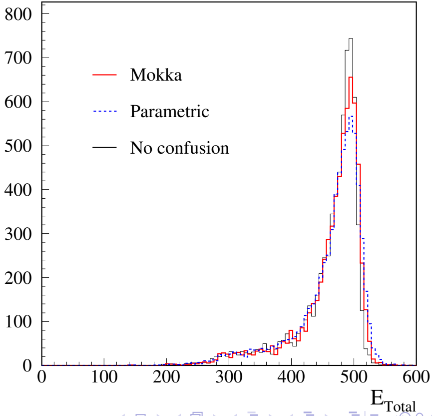
- Identify and factorise:
  1. Probability to split
  2. If split, probability to split off/merge the entire cluster.
  3. If split, but not 100 %: Form of the p.d.f. of the fraction split off.
- Observations:
  1. Depnds on the isolation - strongly for merging, slightly for splitting - but can be treated in two energy bins with no energy dependence in the bin. %5 over-all dependence on barrel/endcap.
  2. Depends only on energy. Is small for splitting, important for merging at low E.
  3. Depends on both energy and isolation (very little for splitting), but only via the average.
- 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

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

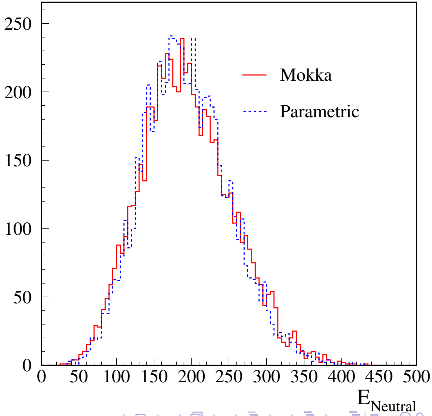
Checking the parametrisation

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



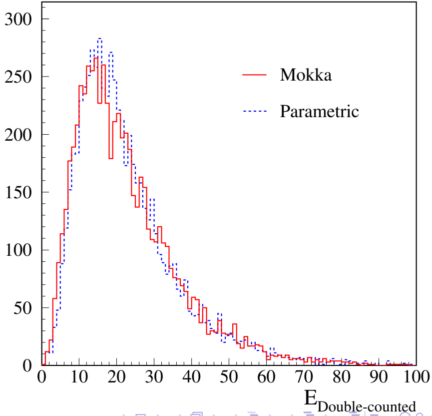
Checking the parametrisation

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



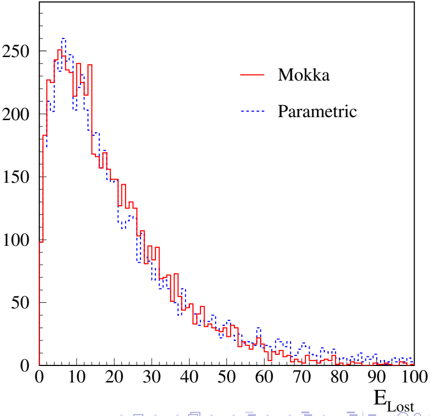
Checking the parametrisation

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



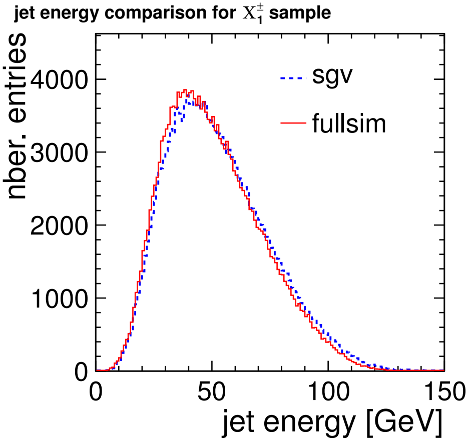
Checking the parametrisation

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



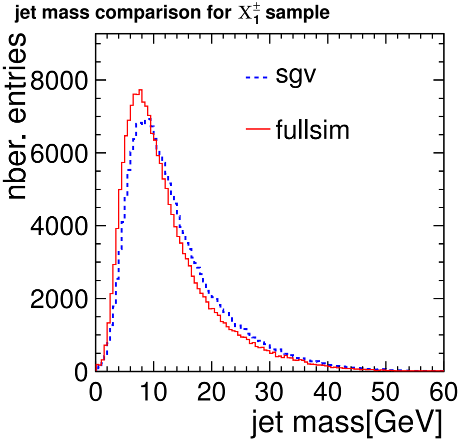
Checking the parametrisation

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



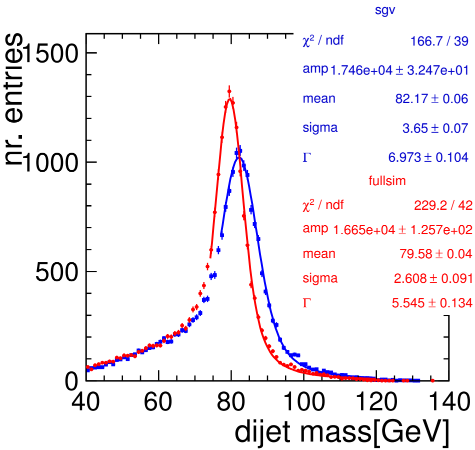
Checking the parametrisation

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



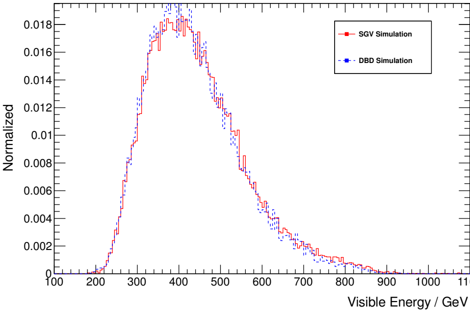
Checking the parametrisation

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



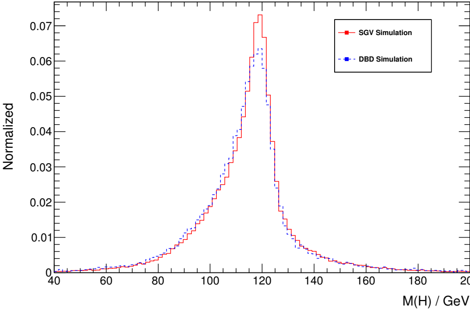
Checking the parametrisation

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



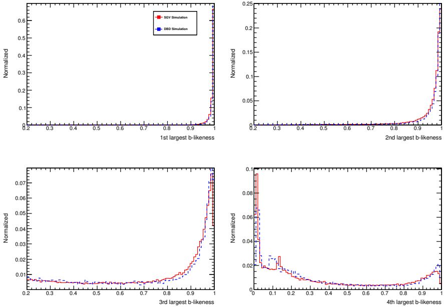
Checking the parametrisation

- Some overall distributions:
  - Total seen energy
  - Total neutral energy
  - Lost and double counted energy.
- Jet propoerties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
  - Di-jet Mass (expect: M_W).
- $\nu\nu h$ benchmark point 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 propoerties in the SUSY benchmark point 5 ($\tilde{\chi}_1^{\pm}$):
  - Jet Energy
  - Jet Mass
  - Di-jet Mass (expect: M_W).
- $\nu\nu h$ benchmark point 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 $\geq$ 2 seen charged tracks: do vertex fit.
    - Expect $\sim$ correct vertex fit-parameteters, 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 !)

On the grid under

lfn:/grid/ilc/users/berggren/mc-dbd/sgv-dst_y/zzz1000-B1b_ws/xxx

(xxx= 2f, 4f, ... , zzz= 1000-B1b_ws, 500-TDR_ws, ...
y is 6 right now. Always use the latest !)

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 !)

On the grid under

lfn:/grid/ilc/users/berggren/mc-dbd/sgv-dst_y/zzz1000-B1b_ws/xxx

(xxx= 2f, 4f, ... , zzz= 1000-B1b_ws, 500-TDR_ws, ...
y is 6 right now. Always use the latest !)

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The method to emulate the performance of PandoraPFO was explained.
- Comparisions to Mokka/Marlin was shown to be quite good.
- SGV mass production works
  - Is done in $\mathcal{O}(1)$ hour.
  - Almost All the bench-marks with background have been processed.
  - Only high cross-section $\gamma\gamma$ events are still to be done.

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The method to emulate the performance of PandoraPFO was explained.
- Comparisions to Mokka/Marlin was shown to be quite good.
- SGV mass production works
  - Is done in $\mathcal{O}(1)$ hour.
  - Almost All the bench-marks with background have been processed.
  - Only high cross-section $\gamma\gamma$ events are still to be done.

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The method to emulate the performance of PandoraPFO was explained.
- Comparisions to Mokka/Marlin was shown to be quite good.
- SGV mass production works
  - Is done in $\mathcal{O}(1)$ hour.
  - Almost All the bench-marks with background have been processed.
  - Only high cross-section $\gamma\gamma$ events are still to be done.

Summary

- The need for FastSim was reviewed:
- Large cross-sections ($\gamma\gamma$), or large parameter-spaces (SUSY) makes such programs obligatory.
- The SGV program was presented, and (I hope) was shown to be up to the job, both in physics and computing performance.
- The method to emulate the performance of PandoraPFO was explained.
- Comparisions to Mokka/Marlin was shown to be quite good.
- SGV mass production works
  - Is done in $\mathcal{O}(1)$ hour.
  - Almost All the bench-marks with background have been processed.
  - Only high cross-section $\gamma\gamma$ events are still to be done.

Summary

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

Installing SGV

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

Then

cd sgv ; . ./install

- Only high cross-section $\gamma\gamma$ events are still to be done.

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
- Finish up particle flow parametrisation:
  - Jet mass.
  - MIP signals.
  - Clusters form 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.

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
- Finish up particle flow parametrisation:
  - Jet mass.
  - MIP signals.
  - Clusters form 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.

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
- Finish up particle flow parametrisation:
  - Jet mass.
  - MIP signals.
  - Clusters form 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.

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
- Finish up particle flow parametrisation:
  - Jet mass.
  - MIP signals.
  - Clusters form 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.

Thank You !

Backup

BACKUP SLIDES

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.
- Further reduce CERNLIB dependence - at the cost of backward

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.
- Further reduce CERNLIB dependence - at a the cost of backward