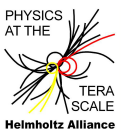
SGV fast detector simulation program : Tutorial

Mikael Berggren[1]

[1]DESY, Hamburg

IDT-WG3 SGV tutorial, Online, 13 October, 2021





Outline

Fast simulation types, and the choice for ILC

Different types, with increasing level of sophistication:

- 4-vector smearing.
- Parametric, needing input from FullSim: Delphes
  - Ignores correlations, eg. between p measurement and ip:s.
  - Hard to handle confusion in high granularity calorimeters.
  - No dE/dx, secondary vertices, effect of hit level inefficiencies ....
  - But very fast, and very condensed output.
  - By theoreticians, for theoreticians.
- Covariance matrix machines, not needing input from FullSim:SGV
  - Full covariance matrix available for each track-helix.
  - Individual shower shape and position generated $\Rightarrow$ can approximate confusion.
  - Hit patterns known $\Rightarrow$ dE/dX and hit-level efficiencies doable.
  - Covariance matrices available $\Rightarrow$ vertex fitting.
  - Anything up to DST-level detail can be output.
  - As fast as Delphes.
  - By experimetalists, for both exprimetalists and theoreticians.

Fast simulation types, and the choice for ILC

Different types, with increasing level of sophistication:

- 4-vector smearing.
- Parametric, needing input from FullSim: Delphes
  - Ignores correlations, eg. between p measurement and ip:s.
  - Hard to handle confusion in high granularity calorimeters.
  - No dE/dx, secondary vertices, effect of hit level inefficiencies ....
  - But very fast, and very condensed output.
  - By theoreticians, for theoreticians.
- Covariance matrix machines, not needing input from FullSim:SGV
  - Full covariance matrix available for each track-helix.
  - Individual shower shape and position generated ⇒ can approximate confusion.
  - Hit patterns known ⇒ dE/dX and hit-level efficiencies doable.
  - Covariance matrices available ⇒ vertex fitting.
  - Anything up to DST-level detail can be output.
  - As fast as Delphes.
  - By experimetalists, for both exprimetalists and theoreticians.

Fast simulation types, and the choice for ILC

Different types, with increasing level of sophistication:

- 4-vector smearing.
- Parametric, needing input from FullSim: Delphes
  - Ignores correlations, eg. between p measurement and ip:s.
  - Hard to handle confusion in high granularity calorimeters.
  - No dE/dx, secondary vertices, effect of hit level inefficiencies ....
  - But very fast, and very condensed output.
  - By theoreticians, for theoreticians.
- Covariance matrix machines, not needing input from FullSim:SGV
  - Full covariance matrix available for each track-helix.
  - Individual shower shape and position generated $\Rightarrow$ can approximate confusion.
  - Hit patterns known $\Rightarrow$ dE/dX and hit-level efficiencies doable.
  - Covariance matrices available $\Rightarrow$ vertex fitting.
  - Anything up to DST-level detail can be output.
  - As fast as Delphes.
  - By experimetalists, for both exprimetalists and theoreticians.

Fast simulation types, and the choice for ILC

Different types, with increasing level of sophistication:

- 4-vector smearing.
- Parametric, needing input from FullSim: Delphes
  - Ignores correlations, eg. between p measurement and ip:s.

For ILC:

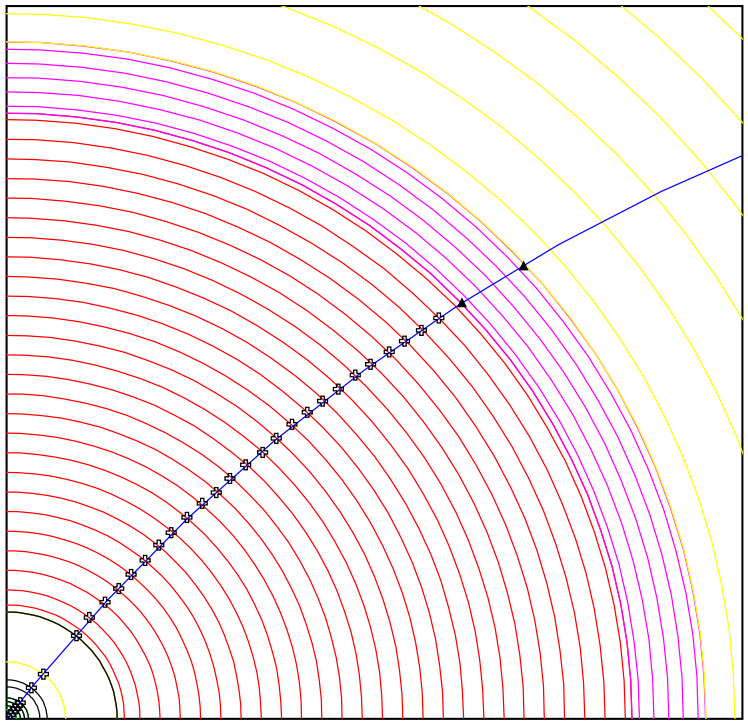
Only Covariance matrix machines have sufficient detail. Here, I'll cover "la Simulation à Grande Vitesse", SGV.

- Covariance matrix machines, not needing input from FullSim:SGV
  - Full covariance matrix available for each track-helix.
  - Individual shower shape and position generated $\Rightarrow$ can approximate confusion.
  - Hit patterns known $\Rightarrow$ dE/dX and hit-level efficiencies doable.
  - Covariance matrices available $\Rightarrow$ vertex fitting.
  - Anything up to DST-level detail can be output.
  - As fast as Delphes.
  - By experimetalists, for both exprimetalists and theoreticians.

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.

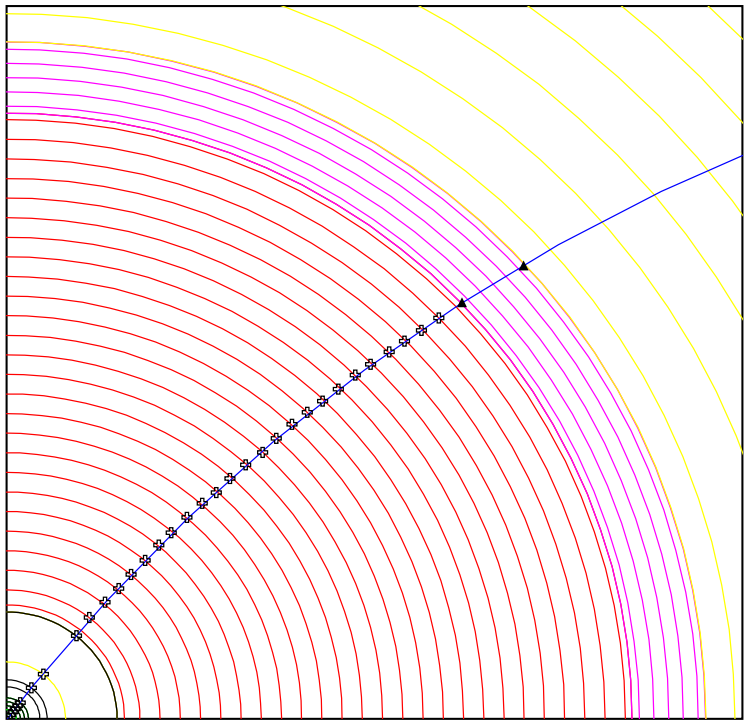


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your Kalman filter does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: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

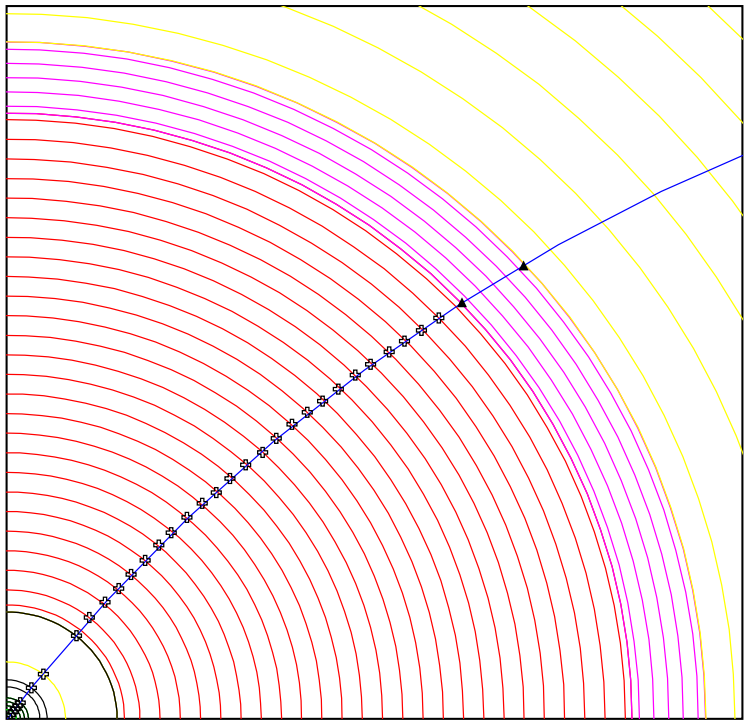


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your Kalman filter does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: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

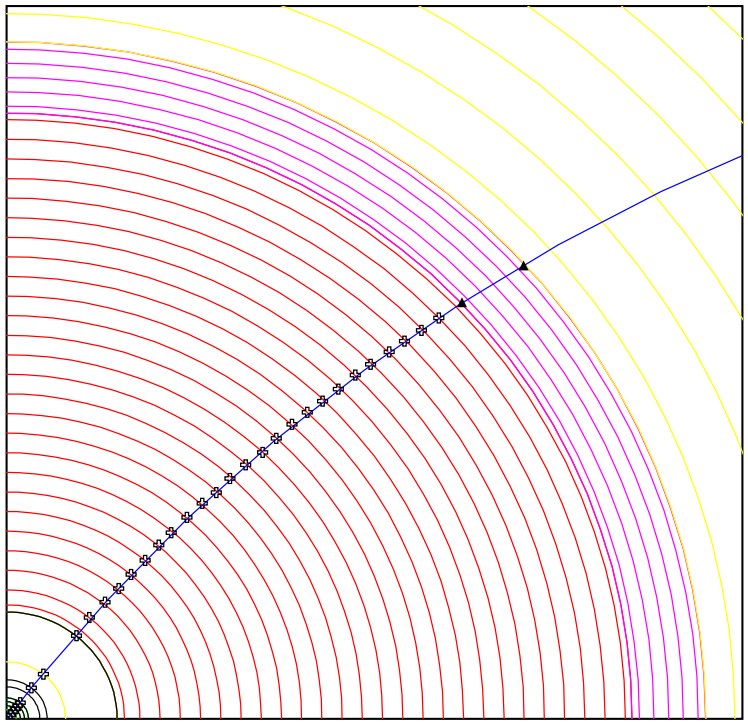


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your Kalman filter does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: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

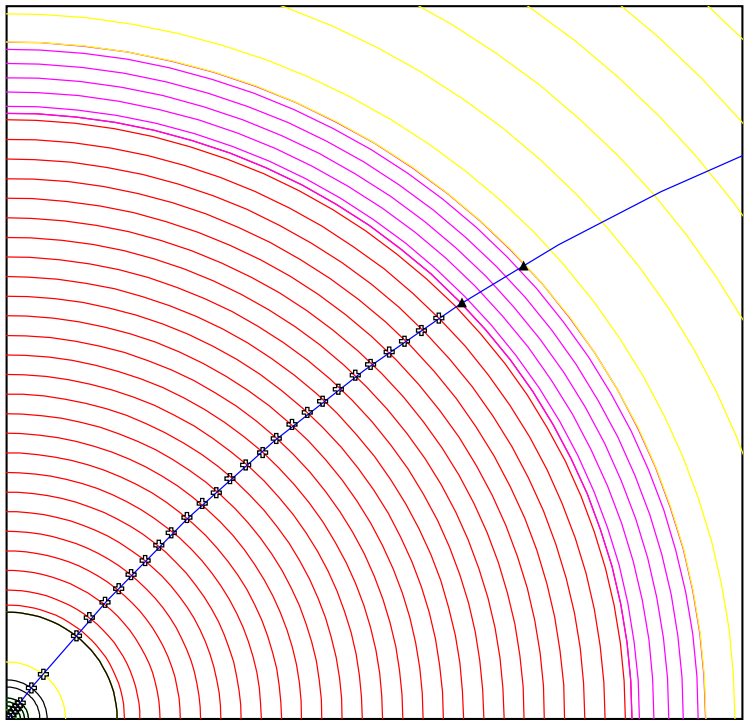


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your Kalman filter does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Helix *parameters* exactly calculated, *errors* with one approximation: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

SGV: How the rest works

SGV is a machine to calculate covariance matrices

Calorimeters: Follow particle to intersection with calorimeters.



- Response type: MIP, EM or hadronic shower, below threshold, etc.
- Simulate single particle response from parameters.
- Easy to plug in more sophisticated shower-simulation.

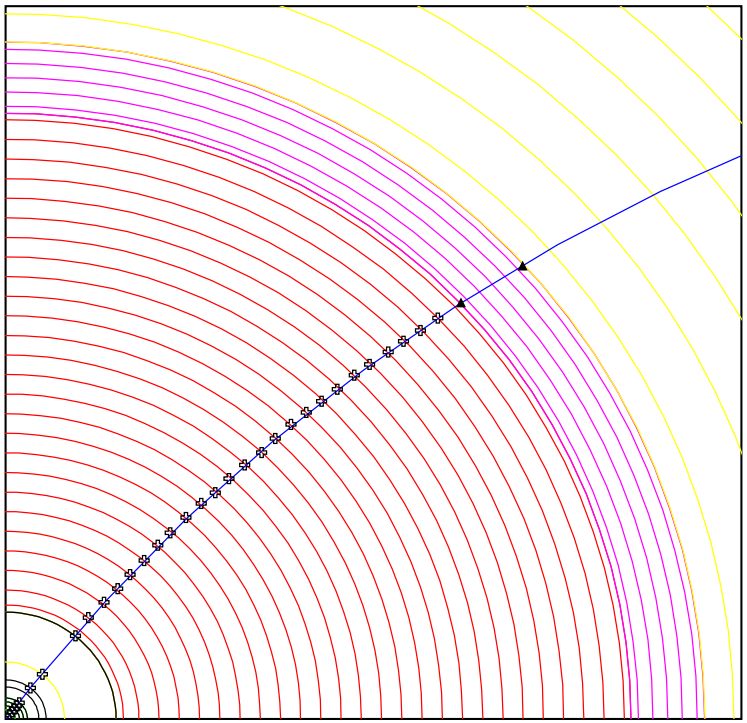
Other stuff:

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

SGV: How the rest works

SGV is a machine to calculate covariance matrices

Calorimeters: Follow particle to intersection with calorimeters.



- Response type: MIP, EM or hadronic shower, below threshold, etc.
- Simulate single particle response from parameters.
- Easy to plug in more sophisticated shower-simulation.

Other stuff:

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

SGV: How the rest works

- 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.
- 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.

SGV: How the rest works

- 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.
- 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.

SGV at work

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

- 43 Mevents.
- $\sim$ 1 hour of wall-clock time (first submit to last completed) on the German NAF.
- SUSY slepton scan: Generate 1000 events at all possible slepton and LSP masses on a 1 GeV $\times$ 1 GeV grid @ E_{CMS} 500 GeV (=31 000 points) $\Rightarrow$ 31 Mevents.
- Used to filter out the *seeable pairs* in 100 000 bunch-crossings from GuineaPig.

More performance in talks at eg.

KeyForHEP | LCWS2021

SGV at work

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

- 43 Mevents.
- $\sim$ 1 hour of wall-clock time (first submit to last completed) on the German NAF.
- SUSY slepton scan: Generate 1000 events at all possible slepton and LSP masses on a 1 GeV $\times$ 1 GeV grid @ E_{CMS} 500 GeV (=31 000 points) $\Rightarrow$ 31 Mevents.
- Used to filter out the *seeable pairs* in 100 000 bunch-crossings from GuineaPig.

More performance in talks at eg.

KeyForHEP | LCWS2021

SGV at work

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

- 43 Mevents.
- $\sim$ 1 hour of wall-clock time (first submit to last completed) on the German NAF.
- SUSY slepton scan: Generate 1000 events at all possible slepton and LSP masses on a 1 GeV $\times$ 1 GeV grid @ E_{CMS} 500 GeV (=31 000 points) $\Rightarrow$ 31 Mevents.
- Used to filter out the *seeable pairs* in 100 000 bunch-crossings from GuineaPig.

More performance in talks at eg.

KeyForHEP | LCWS2021

SGV at work

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

- 43 Mevents.
- $\sim$ 1 hour of wall-clock time (first submit to last completed) on the German NAF.
- SUSY slepton scan: Generate 1000 events at all possible slepton and LSP masses on a 1 GeV $\times$ 1 GeV grid @ E_{CMS} 500 GeV (=31 000 points) $\Rightarrow$ 31 Mevents.
- Used to filter out the *seeable pairs* in 100 000 bunch-crossings from GuineaPig.

More performance in talks at eg.

▸ KeyForHEP ▸ LCWS2021

Technicalities

- Written in Fortran 08, a re-write of the Fortran77-based SGV2 series.Managed in SVN.Install script included.
- Requires:
  - Fortran compiler, e.g. gfortran - any version between 4.7 and 10 ($\Rightarrow$ future-proof!).
  - Standard Linux math: `blas` and `lapack`.
  - PYTHIA vers 6 (even if not needed for event-generation: SGV uses many PYTHIA6 extras).
  - To produce the doc's during installation: `TexLive` and `pandoc`
- Features:
  - On-demand memory allocation.
  - Callable PYTHIA, Whizard 1.x, ...
  - Input from Hepevt/PYJETS, stdhep, slcio, HepMC2/3, GuineaPig.
  - Output of generated event to PYJETS, stdhep, slcio, or HepMC2/3.
  - No imposed format of reconstructed events. Code to make LCIO-DSTs or ROOT trees supplied.
- Typical generation+simulation+reconstruction time $\mathcal{O}(1)$ ms.

Technicalities

- Written in Fortran 08, a re-write of the Fortran77-based SGV2 series.Managed in SVN.Install script included.
- Requires:
  - Fortran compiler, e.g. gfortran - any version between 4.7 and 10 ($\Rightarrow$ future-proof!).
  - Standard Linux math: `blas` and `lapack`.
  - PYTHIA vers 6 (even if not needed for event-generation: SGV uses many PYTHIA6 extras).
  - To produce the doc's during installation: `TexLive` and `pandoc`
- Features:
  - On-demand memory allocation.
  - Callable PYTHIA, Whizard 1.x, ...
  - Input from Hepevt/PYJETS, stdhep, slcio, HepMC2/3, GuineaPig.
  - Output of generated event to PYJETS, stdhep, slcio, or HepMC2/3.
  - No imposed format of reconstructed events. Code to make LCIO-DSTs or ROOT trees supplied.
- Typical generation+simulation+reconstruction time $\mathcal{O}(1)$ ms.

Technicalities

- Written in Fortran 08, a re-write of the Fortran77-based SGV2 series.Managed in SVN.Install script included.
- Requires:
  - Fortran compiler, e.g. gfortran - any version between 4.7 and 10 ($\Rightarrow$ future-proof!).
  - Standard Linux math: `blas` and `lapack`.
  - PYTHIA vers 6 (even if not needed for event-generation: SGV uses many PYTHIA6 extras).
  - To produce the doc's during installation: `TexLive` and `pandoc`
- Features:
  - On-demand memory allocation.
  - Callable PYTHIA, Whizard 1.x, ...
  - Input from Hepevt/PYJETS, stdhep, slcio, HepMC2/3, GuineaPig.
  - Output of generated event to PYJETS, stdhep, slcio, or HepMC2/3.
  - No imposed format of reconstructed events. Code to make LCIO-DSTs or ROOT trees supplied.
- Typical generation+simulation+reconstruction time $\mathcal{O}(1)$ ms.

Technicalities

- Written in Fortran 08, a re-write of the Fortran77-based SGV2 series.Managed in SVN.Install script included.
- Requires:
  - Fortran compiler, e.g. gfortran - any version between 4.7 and 10 ($\Rightarrow$ future-proof!).
  - Standard Linux math: `blas` and `lapack`.
  - PYTHIA vers 6 (even if not needed for event-generation: SGV uses many PYTHIA6 extras).
  - To produce the doc's during installation: `TexLive` and `pandoc`
- Features:
  - On-demand memory allocation.
  - Callable PYTHIA, Whizard 1.x, ...
  - Input from Hepevt/PYJETS, stdhep, slcio, HepMC2/3, GuineaPig.
  - Output of generated event to PYJETS, stdhep, slcio, or HepMC2/3.
  - No imposed format of reconstructed events. Code to make LCIO-DSTs or ROOT trees supplied.
- Typical generation+simulation+reconstruction time $\mathcal{O}(1)$ ms.

Steering SGV

Two steering files...

- Program steering:
  - Single file, with sections for general, generator, detector and analysis steering.
  - Many examples included.
  - Extensive comments in these.
- Geometry description:
  - Described by cylinders and planes.
  - Attach material properties (rad. length, material, int. length, ...)
  - Attach measurement capabilities (quantities measured, dependence on local angles, ...)
  - Several commented examples included.
  - For all details: Chapter 6 in sgv_ug.pdf (created during install)

Steering SGV

Two steering files...

- Program steering:
  - Single file, with sections for general, generator, detector and analysis steering.
  - Many examples included.
  - Extensive comments in these.
- Geometry description:
  - Described by cylinders and planes.
  - Attach material properties (rad. length, material, int. length, ...)
  - Attach measurement capabilities (quantities measured, dependence on local angles, ...)
  - Several commented examples included.
  - For all details: Chapter 6 in sgv_ug.pdf (created during install)

Steering SGV

Two steering files...

- Program steering:
  - Single file, with sections for general, generator, detector and analysis steering.
  - Many examples included.
  - Extensive comments in these.
- Geometry description:
  - Described by cylinders and planes.
  - Attach material properties (rad. length, material, int. length, ...)
  - Attach measurement capabilities (quantities measured, dependence on local angles, ...)
  - Several commented examples included.
  - For all details: Chapter 6 in sgv_ug.pdf (created during install)

The User's Guide

All the details can be found in:

https://www.desy.de/~berggren/sgv_ug/

Summary

- SGV is a full-blown fast detector simulation, not just a parameterised four-vector smearer
  - Comparisons to FullSim can be shown to be quite good, also for complicated features like h.f. tagging.
  - A pre-existing full simulation is not needed to get realistic results. Descriptions of e^+e^- detectors available after installation.
  - Still: SGV is as fast as eg. `Delphes`.
- Many input-methods (internal and external). No output format imposed on the user, plugins for LCIO and Root output available.
- A covariance-machine like SGV is needed not to be cornered between the systematic errors of a parameterised fast-sim, and the statistical errors of FullSiM

Summary

- SGV is a full-blown fast detector simulation, not just a parameterised four-vector smearer
  - Comparisons to FullSim can be shown to be quite good, also for complicated features like h.f. tagging.
  - A pre-existing full simulation is not needed to get realistic results. Descriptions of e^+e^- detectors available after installation.
  - Still: SGV is as fast as eg. `Delphes`.
- Many input-methods (internal and external). No output format imposed on the user, plugins for LCIO and Root output available.
- A covariance-machine like SGV is needed not to be cornered between the systematic errors of a parameterised fast-sim, and the statistical errors of FullSiM

Summary

- SGV is a full-blown fast detector simulation, not just a parameterised four-vector smearer
  - Comparisons to FullSim can be shown to be quite good, also for complicated features like h.f. tagging.
  - A pre-existing full simulation is not needed to get realistic results. Descriptions of e^+e^- detectors available after installation.
  - Still: SGV is as fast as eg. `Delphes`.
- Many input-methods (internal and external). No output format imposed on the user, plugins for LCIO and Root output available.
- A covariance-machine like SGV is needed not to be cornered between the systematic errors of a parameterised fast-sim, and the statistical errors of FullSiM

Summary

- SGV is a full-blown fast detector simulation, not just a parameterised four-vector smearer
  - Comparisons to FullSim can be shown to be quite good, also for complicated features like h.f. tagging.
  - A pre-existing full simulation is not needed to get realistic results. Descriptions of e^+e^- detectors available after installation.
  - Still: SGV is as fast as eg. `Delphes`.
- Many input-methods (internal and external). No output format imposed on the user, plugins for LCIO and Root output available.
- A covariance-machine like SGV is needed not to be cornered between the systematic errors of a parameterised fast-sim, and the statistical errors of FullSiM

Hands-on

If you have checked the system requirements explained on Indico, you can now get your hands on SGV. Now do this:

- Download `tutorial-step-by-step` from Indico.
- Start viewing the file in a separate window - that can be with `more, less, cat` - whatever. Or in the editor.
- ∼Everything in the file that is flush-left can be cut'n'pasted to the command line.
- I will share my desktop and go through the steps with you ...
  - My desktop is a brand-new Ububtu 20.04, running in VirtualBox.
  - I only did the explained preliminaries - including LCIO and root setup.
  - Apart from that, it's an out-of-the-box installation.

Hands-on

If you have checked the system requirements explained on Indico, you can now get your hands on SGV. Now do this:

- Download `tutorial-step-by-step` from Indico.
- Start viewing the file in a separate window - that can be with `more, less, cat` - whatever. Or in the editor.
- ∼Everything in the file that is flush-left can be cut'n'pasted to the command line.
- I will share my desktop and go through the steps with you ...
  - My desktop is a brand-new Ububtu 20.04, running in VirtualBox.
  - I only did the explained preliminaries - including LCIO and root setup.
  - Apart from that, it's an out-of-the-box installation.

Let's go !

Gotchas, updates, out-look

- Stay up-to-date ! Regularly do
  - `svn update`
  - `makesgvlibs (library)`
  - `. ./sgvenv.sh`
    - If `(library)` is an existing one, only changed files, and possible dependencies, will be recompiled. If it is not, a new library is created, and added as the last definition of `SGV_LIB` in `sgvenv.sh`
- `sgvenv.sh` issues
  - `sgvenv.sh` sets the environment *as it was when* `install` *was run*.
  - If you do `makesgvlibs [a new directory]`, the new directory will be appended to `sgvenv.sh`
  - But only that ! If you run `install` without eg. LCIO, and you then install LCIO, follow the instructions in `samples/lcio/00_README`, and do `makesgvlibs` again, the LCIO enabled library *is* installed, but `sgvenv.sh` still will unset LCIO ⇒ You need to edit `sgvenv.sh`.

Gotchas, updates, out-look

- Stay up-to-date ! Regularly do
  - `svn update`
  - `makesgvlibs (library)`
  - `. ./sgvenv.sh`
    - If `(library)` is an existing one, only changed files, and possible dependencies, will be recompiled. If it is not, a new library is created, and added as the last definition of `SGV_LIB` in `sgvenv.sh`
- `sgvenv.sh` issues
  - `sgvenv.sh` sets the environment *as it was when* `install` *was run*.
  - If you do `makesgvlibs [a new directory]`, the new directory will be appended to `sgvenv.sh`
  - But only that ! If you run `install` without eg. LCIO, and you then install LCIO, follow the instructions in `samples/lcio/00_README`, and do `makesgvlibs` again, the LCIO enabled library *is* installed, but `sgvenv.sh` still will unset LCIO ⇒ You need to edit `sgvenv.sh`.

Gotchas, updates, out-look

- Stay up-to-date ! Regularly do
  - `svn update`
  - `makesgvlibs (library)`
  - `. ./sgvenv.sh`
    - If `(library)` is an existing one, only changed files, and possible dependencies, will be recompiled. If it is not, a new library is created, and added as the last definition of `SGV_LIB` in `sgvenv.sh`
- `sgvenv.sh` issues
  - `sgvenv.sh` sets the environment *as it was when* `install` *was run*.
  - If you do `makesgvlibs [a new directory]`, the new directory will be appended to `sgvenv.sh`
  - But only that ! If you run `install` without eg. LCIO, and you then install LCIO, follow the instructions in `samples/lcio/00_README`, and do `makesgvlibs` again, the LCIO enabled library *is* installed, but `sgvenv.sh` still will unset LCIO ⇒ You need to edit `sgvenv.sh`.

Gotchas, updates, out-look

- Stay up-to-date ! Regularly do
  - `svn update`
  - `makesgvlibs (library)`
  - `. ./sgvenv.sh`
    - If `(library)` is an existing one, only changed files, and possible dependencies, will be recompiled. If it is not, a new library is created, and added as the last definition of `SGV_LIB` in `sgvenv.sh`
- `sgvenv.sh` issues
  - `sgvenv.sh` sets the environment *as it was when* `install` *was run*.
  - If you do `makesgvlibs [a new directory]`, the new directory will be appended to `sgvenv.sh`
  - But only that ! If you run `install` without eg. LCIO, and you then install LCIO, follow the instructions in `samples/ lcio/ 00_README`, and do `makesgvlibs` again, the LCIO enabled library *is* installed, but `sgvenv.sh` still will unset LCIO ⇒ You need to edit `sgvenv.sh`.

Gotchas, updates, out-look

- Things can always go wrong...
  - In my experience, crashes are in 90% of the cases due to wrong input: Eg. an slcio generator-file is read, but the steering says that the input is stdhep, or the input, steering or geometry file is missing.
  - $\Rightarrow$ Check your soft-links and steering file!
  - If that isn't the problem: re-compile with debug, to localise the point-of-failure:

```
cd $SGV_DIR
makesgvlibs [you library] debug
. ./sgvenv.sh d
cd [your working directory]
cresgvexe merge [your program] "debug [+ any other compile optinons]"
```

    - The `debug` argument to `makesgvlibs` sets the options, and also creates a *different* sub-directory for debug-libraries (by default `[you library]/deb`). The `d` as the second argument to `sgvenv.sh` will set `SGV_LIB` to this.

Gotchas, updates, out-look

- Things can always go wrong...
  - In my experience, crashes are in 90% of the cases due to wrong input: Eg. an slcio generator-file is read, but the steering says that the input is stdhep, or the input, steering or geometry file is missing.
  - $\Rightarrow$ Check your soft-links and steering file!
  - If that isn't the problem: re-compile with debug, to localise the point-of-failure:

```
cd $SGV_DIR
makesgvlibs [you library] debug
. ./sgvenv.sh d
cd [your working directory]
cresgvexe merge [your program] "debug [+ any other compile optinons]"
```

    - The `debug` argument to `makesgvlibs` sets the options, and also creates a *different* sub-directory for debug-libraries (by default `[you library]/deb`). The `d` as the second argument to `sgvenv.sh` will set `SGV_LIB` to this.

Gotchas, updates, out-look

- Things can always go wrong...
  - In my experience, crashes are in 90% of the cases due to wrong input: Eg. an slcio generator-file is read, but the steering says that the input is stdhep, or the input, steering or geometry file is missing.
  - $\Rightarrow$ Check your soft-links and steering file!
  - If that isn't the problem: re-compile with debug, to localise the point-of-failure:

```
cd $SGV_DIR
makesgvlibs [you library] debug
. ./sgvenv.sh d
cd [your working directory]
cresgvexe merge [your program] "debug [+ any other compile optinons]"
```

    - The `debug` argument to `makesgvlibs` sets the options, and also creates a *different* sub-directory for debug-libraries (by default `[you library]/deb`). The `d` as the second argument to `sgvenv.sh` will set `SGV_LIB` to this.

Gotchas, updates, out-look

- There is more features, not covered in this tutorial. Check the READMEs:
  - BCal: look in `samples/usercalo`.
  - HepMC: look in `samples/hepmc` (Note that HepMC itself needs a patch for this)
  - Confusion in the calorimeters: look in `samples/pflow`.
  - Whizard: look in `samples/whizard`. Only Whizard v1 for now ...
  - Filtering: Select generated events to output after generation, or simulation, or analysis. Look in the `global_generation_steering` section of any example steering file.
- To come: SGV4
  - Move to git (GitHub or GitLab).
  - Tidy up comments, docs, conditional code, scripts, ... from all CERNLIB/Fortran 77 references.
  - Modernise the steering state-structure

Gotchas, updates, out-look

- There is more features, not covered in this tutorial. Check the READMEs:
  - BCal: look in `samples/usercalo`.
  - HepMC: look in `samples/hepmc` (Note that HepMC itself needs a patch for this)
  - Confusion in the calorimeters: look in `samples/pflow`.
  - Whizard: look in `samples/whizard`. Only Whizard v1 for now ...
  - Filtering: Select generated events to output after generation, or simulation, or analysis. Look in the `global_generation_steering` section of any example steering file.
- To come: SGV4
  - Move to git (GitHub or GitLab).
  - Tidy up comments, docs, conditional code, scripts, ... from all CERNLIB/Fortran 77 references.
  - Modernise the steering state-structure

Gotchas, updates, out-look

- There is more features, not covered in this tutorial. Check the READMEs:
  - BCal: look in `samples/usercalo`.
  - HepMC: look in `samples/hepmc` (Note that HepMC itself needs a patch for this)
  - Confusion in the calorimeters: look in `samples/pflow`.
  - Whizard: look in `samples/whizard`. Only Whizard v1 for now ...
  - Filtering: Select generated events to output after generation, or simulation, or analysis. Look in the `global_generation_steering` section of any example steering file.
- To come: SGV4
  - Move to git (GitHub or GitLab).
  - Tidy up comments, docs, conditional code, scripts, ... from all CERNLIB/Fortran 77 references.
  - Modernise the steering state-structure

Gotchas, updates, out-look

- There is more features, not covered in this tutorial. Check the READMEs:
  - BCal: look in `samples/usercalo`.
  - HepMC: look in `samples/hepmc` (Note that HepMC itself needs a patch for this)
  - Confusion in the calorimeters: look in `samples/pflow`.
  - Whizard: look in `samples/whizard`. Only Whizard v1 for now ...
  - Filtering: Select generated events to output after generation, or simulation, or analysis. Look in the `global_generation_steering` section of any example steering file.
- To come: SGV4
  - Move to git (GitHub or GitLab).
  - Tidy up comments, docs, conditional code, scripts, ... from all CERNLIB/Fortran 77 references.
  - Modernise the steering state-structure

Gotchas, updates, out-look

- There is more features, not covered in this tutorial. Check the READMEs:
  - BCal: look in `samples/usercalo`.
  - HepMC: look in `samples/hepmc` (Note that HepMC itself needs a patch for this)
  - Confusion in the calorimeters: look in `samples/pflow`.
  - Whizard: ...rd v1 for now ...
  - Filtering: ...eneration, or simulatic... `global_`... example steering file.

Need a Fortran95 crash-course?
Here it is:

▸ WikiPedia

- To come: SGV4
  - Move to git (GitHub or GitLab).
  - Tidy up comments, docs, conditional code, scripts, ... from all CERNLIB/Fortran 77 references.
  - Modernise the steering state-structure

Reminder: Installing SGV

Do

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

Check that `lapack` and `blas` installed. If PYTHIA6 is not installed, run `bash sgv/install-pythia` to get it from HepForge.

Then

cd sgv ; . ./install[1]

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 ROOT interface set up, and produce you first ROOT tree.
  - Get the LCIO, HepMC, and/or STDHEP i/o set up.
  - Make you first LCIO DST.

[1] Under bash. Under c-shell, do `bash install ; source sgvenv.csh`

Reminder: Installing SGV

Do

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

Check that `lapack` and `blas` installed. If PYTHIA6 is not installed, run `bash sgv/install-pythia` to get it from HepForge.

Then

cd sgv ; . ./install[1]

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 ROOT interface set up, and produce you first ROOT tree.
  - Get the LCIO, HepMC, and/or STDHEP i/o set up.
  - Make you first LCIO DST.

[1] Under bash. Under c-shell, do `bash install ; source sgvenv.csh`

Reminder: Installing SGV

Do

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

Check that `lapack` and `blas` installed. If PYTHIA6 is not installed, run `bash sgv/install-pythia` to get it from HepForge.

Then

cd sgv ; . ./install[1]

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 ROOT interface set up, and produce you first ROOT tree.
  - Get the LCIO, HepMC, and/or STDHEP i/o set up.
  - Make you first LCIO DST.

[1] Under bash. Under c-shell, do `bash install ; source sgvenv.csh`

Thank You !

Backup

BACKUP SLIDES

... a tool for rapid LC studies?

Peer-reviewed papers using SGV

- Phys. Rev D101 (2020) 7, 075053
- ILD-PHYS-2019-001 (Accepted by Phys. ReV. D)
- Eur.Phys.J.C 76 (2016) 4,183
- Eur.Phys.J.C 75 (2015) 12, 617
- Phys. Rev D 91 (2015) 113007
- Phys. Rev D 90 (2014) 114029
- Phys. Rev D 89 (2014) 11, 113006
- Eur.Phys.J.C 73 (2013) 12,2660
- Eur.Phys.J.C 72 (2012) 2213
- Phys. Rev. D 82 (2010) 055016
- NIM A 579 (2007) 750
- Eur. Phys. J. C 31 (2003) 421
- Eur. Phys, J. direct (2000) 1

+ innumerable theses, reports, arXiv submissions and conference proceedings. Including the Tesla TDR, LoI, TDR, the IDR and ILC/ILD inputs to EPPSU and Snowmass 2013.

... a tool for rapid LC studies ?

SGV was used for

- Defining the forward tracking geometry of LDC:
  - Vienna 2005. ▸ LDC and ▸ tracking
- The utility (or not) of the silicon envelope
  - ▸ Valencia 2006
- Merge of LDC and GLD into ILD
  - ▸ Cambrige 2008
- Define the options for the IDR
  - ▸ KEK 2015
- Also: SGV is part of the FullSim machinery: It is used to select which part of the pairs-background to overlay.

... a tool for rapid LC studies ?

SGV was used for

- Defining the forward tracking geometry of LDC:
  - Vienna 2005. ▸ LDC and ▸ tracking
- The utility (or not) of the silicon envelope
  - ▸ Valencia 2006
- Merge of LDC and GLD into ILD
  - ▸ Cambrige 2008
- Define the options for the IDR
  - ▸ KEK 2015
- Also: SGV is part of the FullSim machinery: It is used to select which part of the pairs-background to overlay.

... a tool for rapid LC studies !

SGV was used for

- Defining the forward tracking geometry of LDC:
  - Vienna 2005. ▸ LDC and ▸ tracking
- The utility (or not) of the silicon envelope
  - ▸ Valencia 2006
- Merge of LDC and GLD into ILD
  - ▸ Cambrige 2008
- Define the options for the IDR
  - ▸ KEK 2015
- Also: SGV is part of the FullSim machinery: It is used to select which part of the pairs-background to overlay.

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

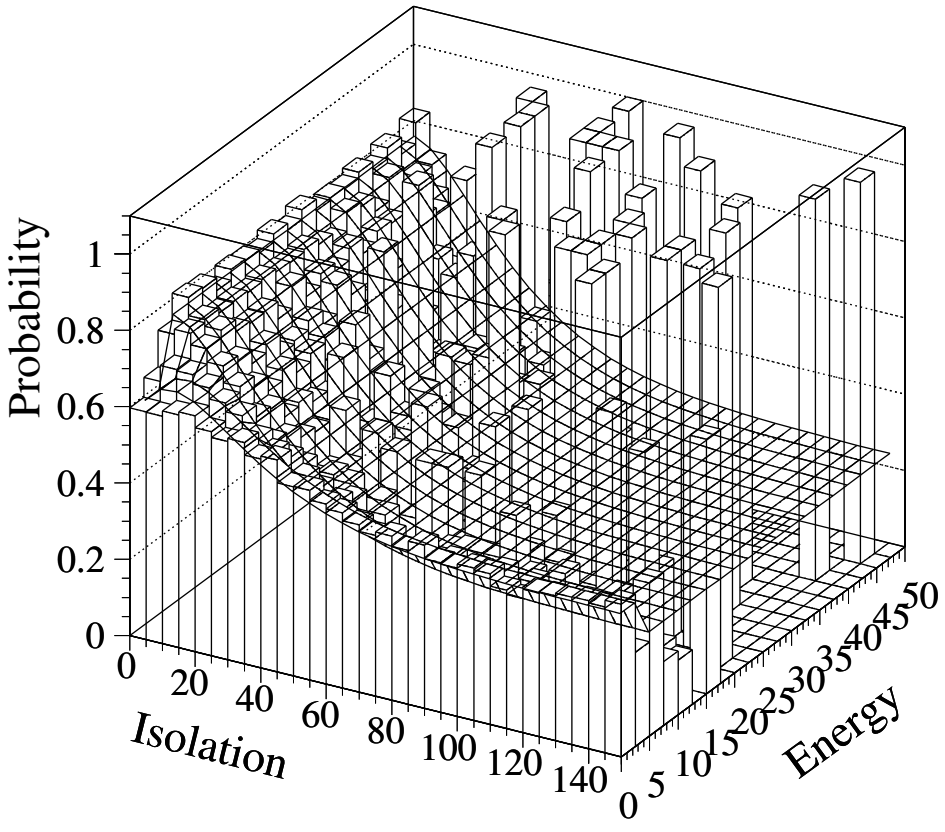
- Concentrate on what really matters:
  - True charged particles splitting off (a part of) their shower: double-counting.
  - True neutral particles merging (a part of) their shower with charged particles: enetgy loss.
- Don't care about neutral-neutral or charged-charged merging.
- Nor about multiple 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.

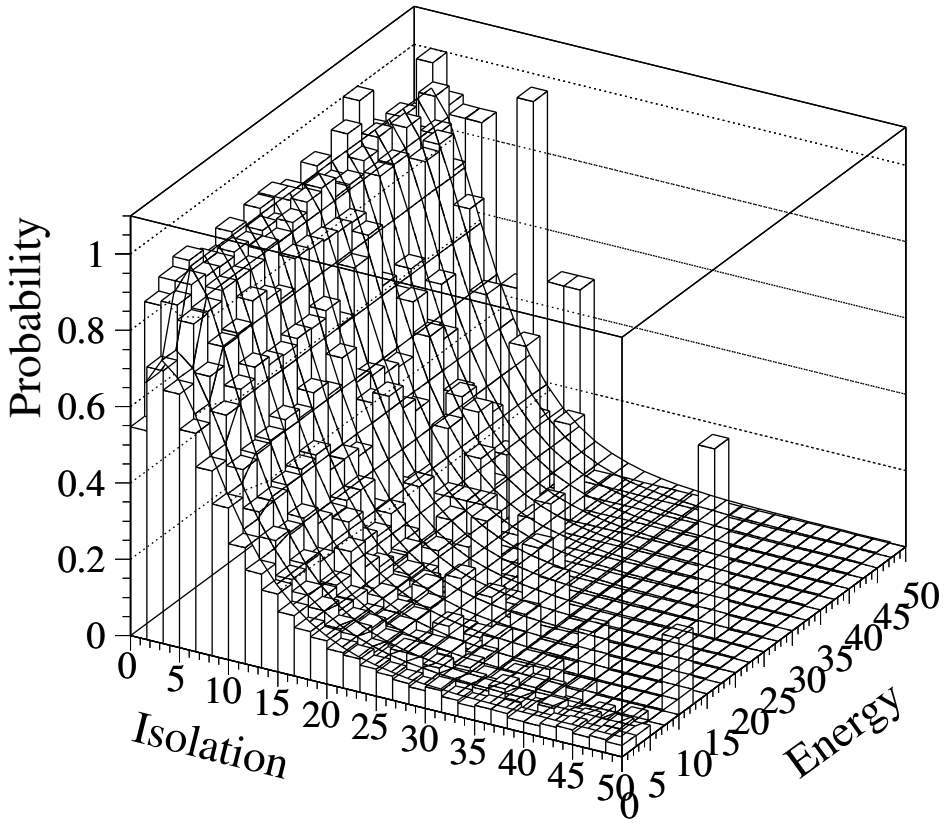
Observed distributions

- 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.



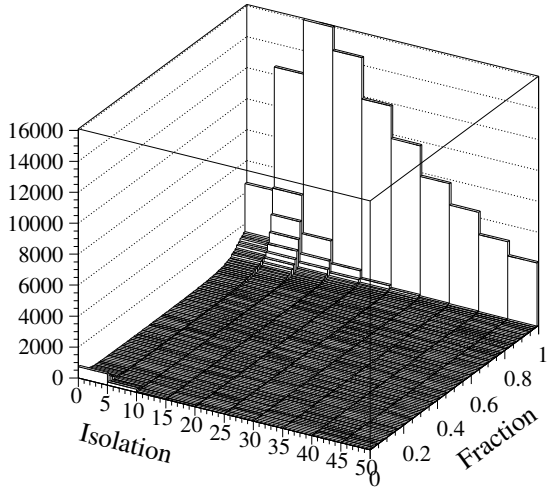
Observed distributions

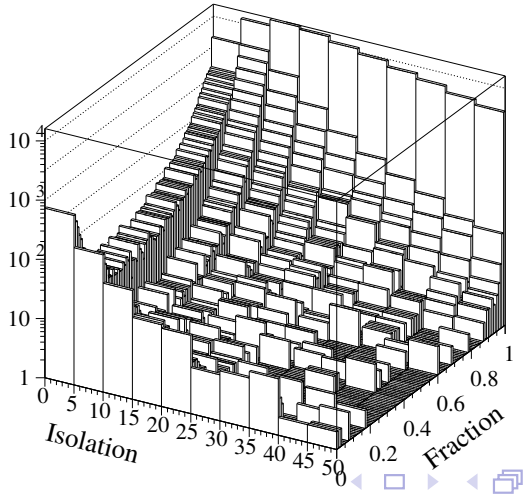
- Probability to split (charged had or γ)
- Fraction the energy vs distance
- ... and vs E
- Fit of the Distribution of the fraction
- Average fraction vs. E and distance.



Observed distributions

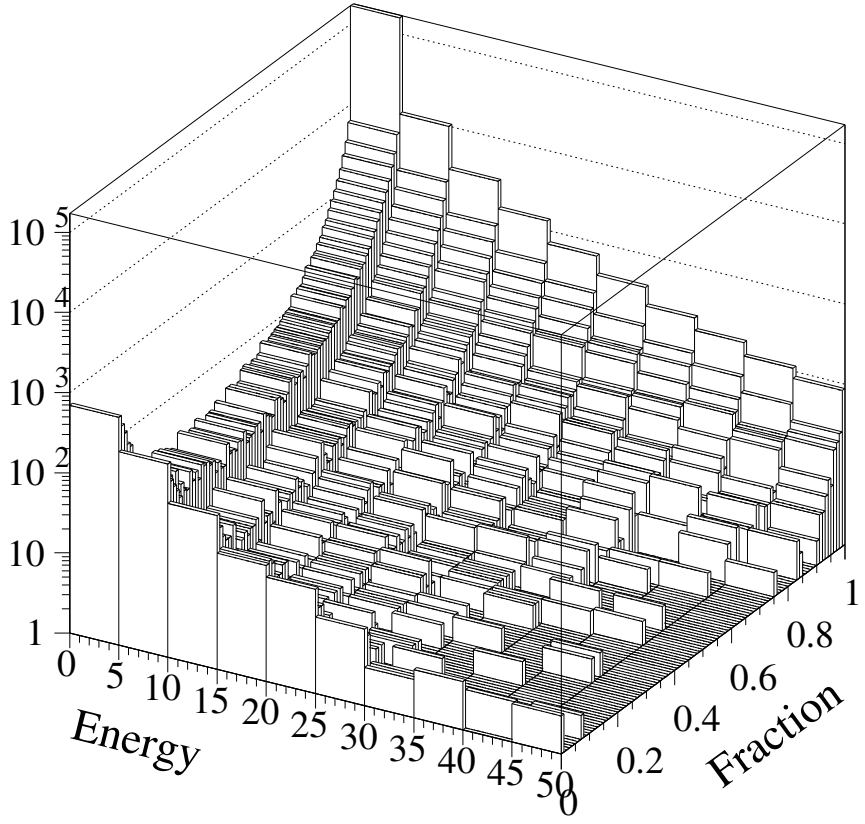
- Probability to split (charged had or γ)
- Fraction the energy vs distance
- ... and vs E
- Fit of the Distribution of the fraction
- Average fraction vs. E and distance.





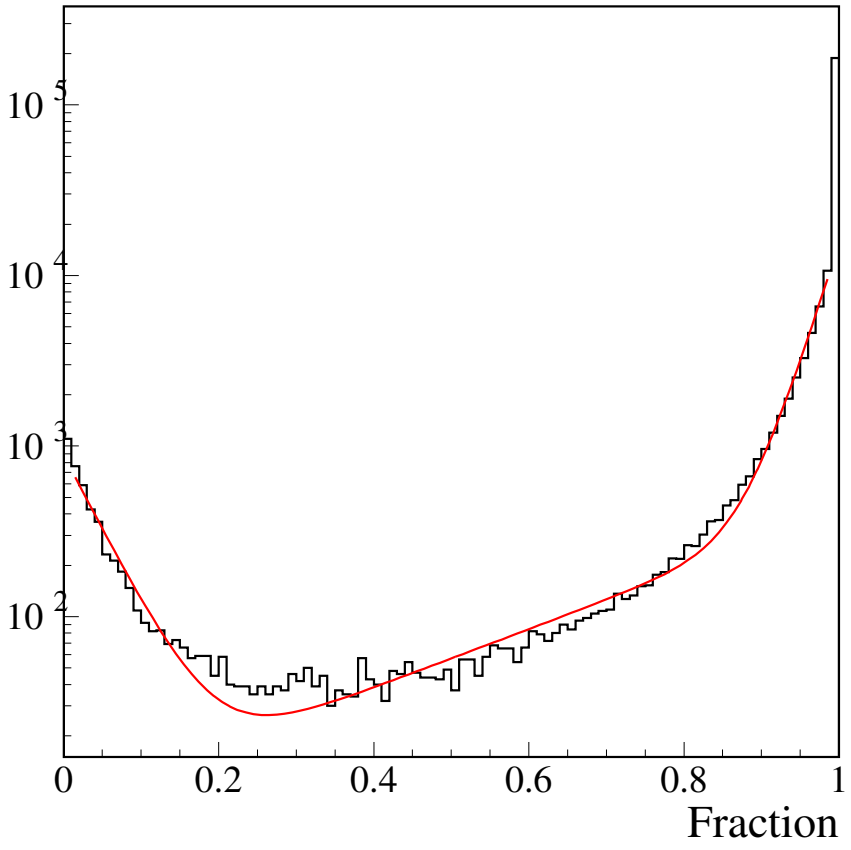
Observed distributions

- 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.



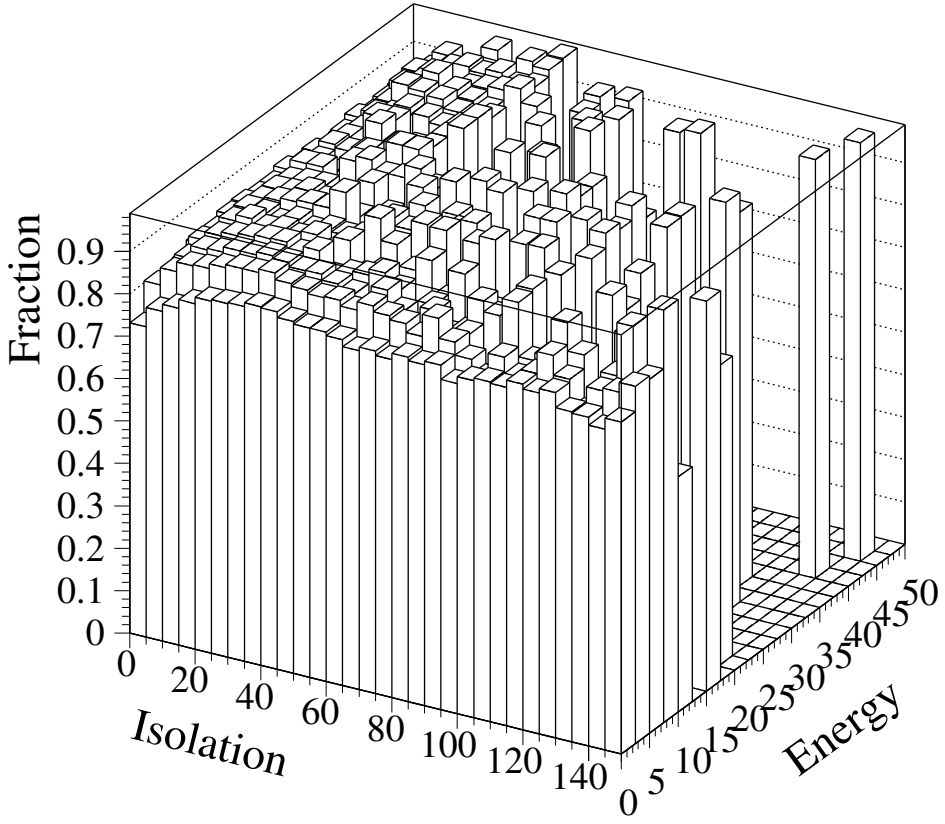
Observed distributions

- 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.



LCIO Collections with DST output

- 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.