SGV - fast simulation of ILC detectors for physics studies

Mikael Berggren[1]

[1] DESY, Hamburg

LCForum, DESY, Feb 7-9, 2012

Outline

1. The need for fast simulation
   - Ex1: $\gamma\gamma$ cross-sections
   - Ex2: SUSY scans
2. Fast simulation
   - SGV
3. Calorimeter simulation
4. Summary and Outlook

The need for fast simulation

- 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 LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

The need for fast simulation

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer 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 LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

The need for fast simulation

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer 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 LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

The need for fast simulation

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer 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 LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

The need for fast simulation

- We have very good full simulation now.
- So why bother about fast simulation ?
- Answer:
  - R. Heuer 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 LOI exercise showed that for physics, the fastSim studies were good enough.

But most of all:

Fast simulation is Fast !

So...

Why do we need speed ?

Cross-section and event-generation time

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

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

Cross-section and event-generation time

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.

Cross-section and event-generation time

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

- $\int \mathcal{L}dt = 500\ \mathrm{fb}^{-1} \to 18 \star 10^9$ events are expected.
- 10 ms to generate one event.
- 10 ms to fastsim (SGV) one event.

10^8 s of CPU time is needed, ie more than 3 years. But:This goes to 3000 years with full simulation.

SUSY parameter scans

Simple example:

- MSUGRA: 4 parameters + sign of μ
- Scan each in eg. 20 steps
- Eg. 5000 events per point (modest requirement: in sps1a' almost 1 million SUSY events are expected for 500 fb^{-1} !)
- = $20^4 \times 2 \times 5000 = 1.6 \times 10^9$ events to generate...

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

Also here: CPU millenniums with full simulation

SUSY parameter scans

Simple example:

- MSUGRA: 4 parameters + sign of μ
- Scan each in eg. 20 steps
- Eg. 5000 events per point (modest requirement: in sps1a' almost 1 million SUSY events are expected for 500 fb^{-1} !)
- = $20^4 \times 2 \times 5000 = 1.6 \times 10^9$ events to generate...

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

Also here: CPU millenniums with full simulation

Fast simulation

Different types, with increasing level of sophistication:

- 4-vector smearing.
- Parametric. Eg SIMDET
- Covariance matrix machines. Eg. LiCToy, SGV

Common for all:

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

I will talk about

"la Simulation à Grande Vitesse", SGV.

Fast simulation

Different types, with increasing level of sophistication:

- 4-vector smearing.
- Parametric. Eg SIMDET
- Covariance matrix machines. Eg. LiCToy, SGV

Common for all:

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

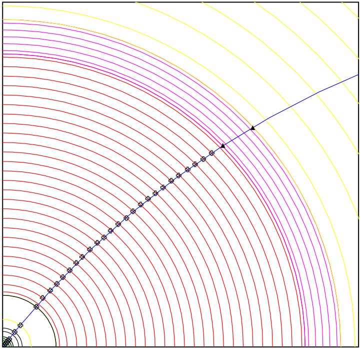
I will talk about

"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.

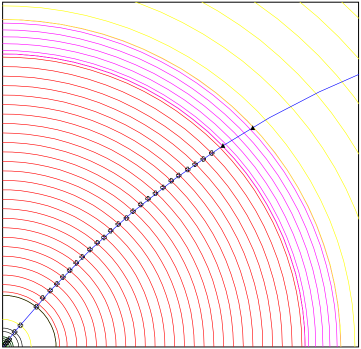


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Information on hit-pattern accessible to analysis. Co-ordinates of hits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

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.

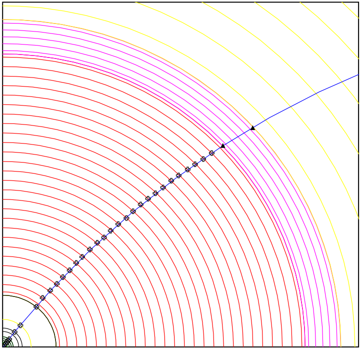


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Information on hit-pattern accessible to analysis. Co-ordinates of hits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

SGV: How tracking works

SGV is a machine to calculate covariance matrices

Tracking: Follow track-helix through the detector.

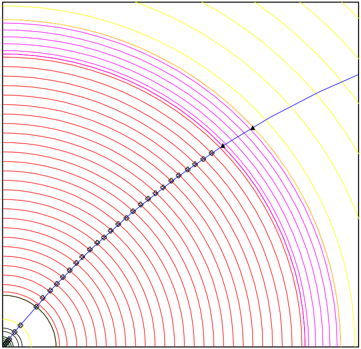


- Calculate cov. mat. at perigee, including material, measurement errors and extrapolation. NB: this is exactly what Your track fit does!
- Smear perigee parameters (Choleski decomposition: takes all correlations into account)
- Information on hit-pattern accessible to analysis. Co-ordinates of hits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

SGV: How tracking works

SGV is a machine to calculate covariance matrices

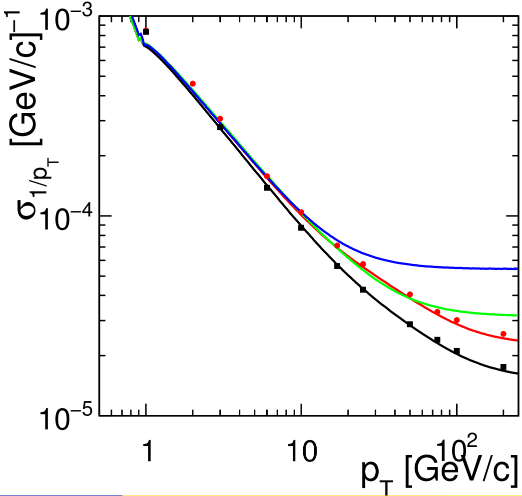
Tracking: Follow track-helix through the detector.



- 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)
- Information on hit-pattern accessible to analysis. Co-ordinates of hits accessible.

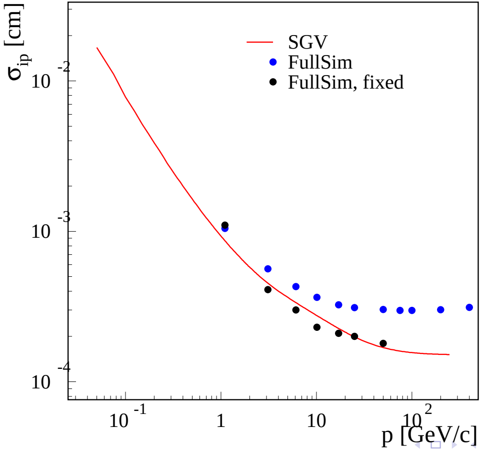
SGV and FullSim LDC/ILD: momentum resolution

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.
- 20 000 lines + 10 000 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.
- Output of generated event to PYJETS or stdhep.
- samples subdirectory with READMEs, steering and code.
- output LCIO DST.

Installing SGV

Do

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

Then

```
cd SGV-3.0rc1 ; bash install (+maybe ; bash makesgvlibs lib )
```

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

Then

```
cd SGV-3.0rc1 ; bash install (+maybe ; bash makesgvlibs lib )
```

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

Then

```
cd SGV-3.0rc1 ; bash install (+maybe ; bash makesgvlibs lib )
```

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.
- 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.
- 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.
- Consequences:
  - If a (part of) a neutral cluster associated to track $\rightarrow$ Energy is lost.
  - If a (part of) a charged cluster not associated to any track $\rightarrow$ Energy is double-counted.
  - Other errors (split neutral cluster, charged cluster assoiated with wrong track ....) are of less importance.

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.

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)

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

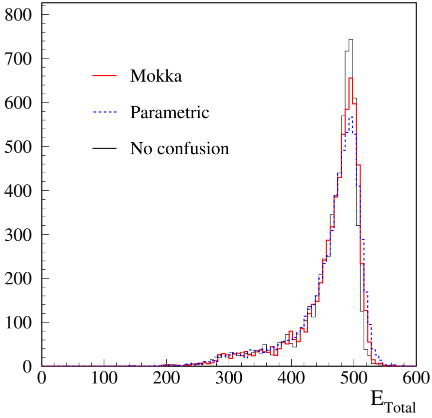
- When running over the fully simulated/reconstructed sample:
  - Use the 3 functions to simulate double - counting / loss for each true particle
  - Compare with full reco
- Twiddle knobs: E vs p, overall split - probability.
- Total seen energy
- Total neutral energy
- Lost and double counted energy.

Checking the parametrisation

- When running over the fully simulated/reconstructed sample:
  - Use the 3 functions to simulate double - counting / loss for each true particle
  - Compare with full reco
- Twiddle knobs: E vs p, overall split - probability.
- Total seen energy
- Total neutral energy
- Lost and double counted energy.

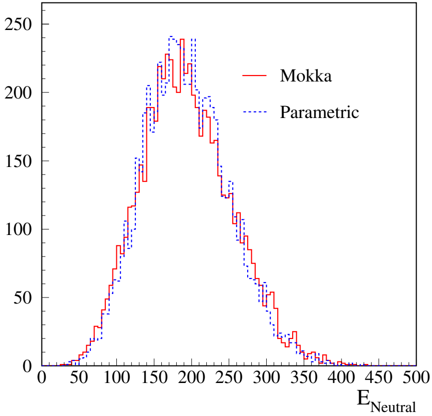
Checking the parametrisation

- When running over the fully simulated/reconstructed sample:
  - Use the 3 functions to simulate double - counting / loss for each true particle
  - Compare with full reco
- Twiddle knobs: E vs p, overall split - probability.
- Total seen energy
- Total neutral energy
- Lost and double counted energy.



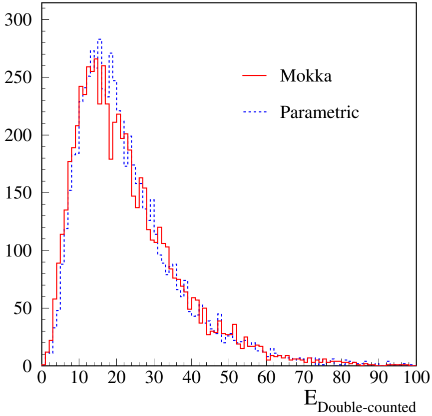
Checking the parametrisation

- When running over the fully simulated/reconstructed sample:
  - Use the 3 functions to simulate double - counting / loss for each true particle
  - Compare with full reco
- Twiddle knobs: E vs p, overall split - probability.
- Total seen energy
- Total neutral energy
- Lost and double counted energy.



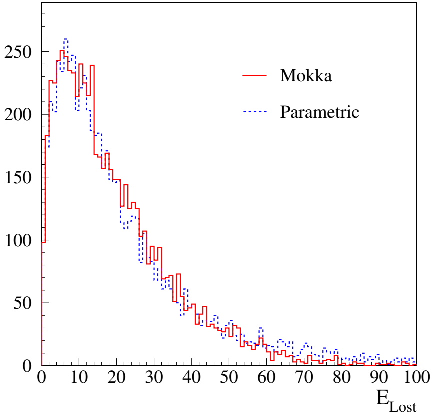
Checking the parametrisation

- When running over the fully simulated/reconstructed sample:
  - Use the 3 functions to simulate double - counting / loss for each true particle
  - Compare with full reco
- Twiddle knobs: E vs p, overall split - probability.
- Total seen energy
- Total neutral energy
- Lost and double counted energy.



Checking the parametrisation

- When running over the fully simulated/reconstructed sample:
  - Use the 3 functions to simulate double - counting / loss for each true particle
  - Compare with full reco
- Twiddle knobs: E vs p, overall split - probability.
- Total seen energy
- Total neutral energy
- Lost and double counted energy.



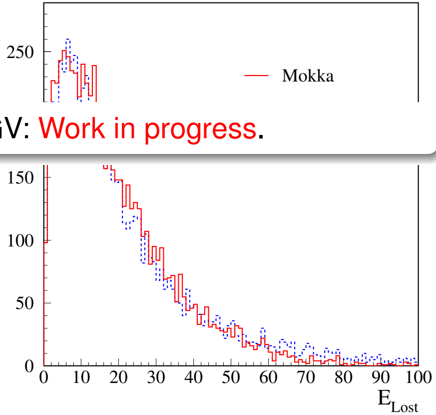
Checking the parametrisation

- When running over the fully simulated/reconstructed sample:
  - Use the 3 functions to simulate double - counting / loss for each true particle
  - Compare with full reco

Promising ! Will be integrated into SGV: Work in progress.

split - probability.

- Total seen energy
- Total neutral energy
- Lost and double counted energy.



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.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.

Installing SGV

svn export https://svnsrv.desy.de/public/sgv/tags/SGV-3.0rc1
Then
cd /SGV-3.0rc1 ; bash install

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.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.

Installing SGV

svn export https://svnsrv.desy.de/public/sgv/tags/SGV-3.0rc1
Then
cd /SGV-3.0rc1 ; bash install

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.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.

Installing SGV

svn export https://svnsrv.desy.de/public/sgv/tags/SGV-3.0rc1
Then
cd /SGV-3.0rc1 ; bash install

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.
- First comparisions to Mokka/Marlin with a first tentative tuning was shown to be promising.

Installing SGV

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

Then

cd /SGV-3.0rc1 ; bash install

Outlook

(On time-scale days to weeks)

- 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.
- Fix a few identfied issues, then Release SGV3.0 (no rc1).
- Produce LCIO DST:s for the DBD bench-marks: DBD analyses can start ≈ now, while waiting for full-sim.

Outlook

(On time-scale days to weeks)

- 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.
- Fix a few identfied issues, then Release SGV3.0 (no rc1).
- Produce LCIO DST:s for the DBD bench-marks: DBD analyses can start ≈ now, while waiting for full-sim.

Outlook

(On time-scale days to weeks)

- 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.
- Fix a few identfied issues, then Release SGV3.0 (no rc1).
- Produce LCIO DST:s for the DBD bench-marks: DBD analyses can start $\approx$ now, while waiting for full-sim.

Outlook

(On time-scale days to weeks)

- 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.
- Fix a few identfied issues, then Release SGV3.0 (no rc1).
- Produce LCIO DST:s for the DBD bench-marks: DBD analyses can start ≈ now, while waiting for full-sim.

Outlook

(On time-scale days to weeks)

- 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.
- Fix a few identfied issues, then Release SGV3.0 (no rc1).
- Produce LCIO DST:s for the DBD bench-marks: DBD analyses can start $\approx$ now, while waiting for full-sim.

Outlook

(On time-scale days to weeks)

- 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.
- Fix a few identfied issues, then Release SGV3.0 (no rc1).
- Produce LCIO DST:s for the DBD bench-marks: DBD analyses can start $\approx$ now, while waiting for full-sim.

Outlook

(On time-scale days to weeks)

- 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.
- Fix a few identfied issues, then Release SGV3.0 (no rc1).
- Produce LCIO DST:s for the DBD bench-marks: DBD analyses can start $\approx$ now, while waiting for full-sim.

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

- 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 compatibility on steering files ? HBOOK dependence will remain in the forseable future - but only for user convenience : SGV itself doesn't need it.