Update on EvtGen
and related simulation developments

Fernando Abudinén, John Back, Michal Kreps, Thomas Latham,

VINCIA: Giacomo Morgante, Peter Skands

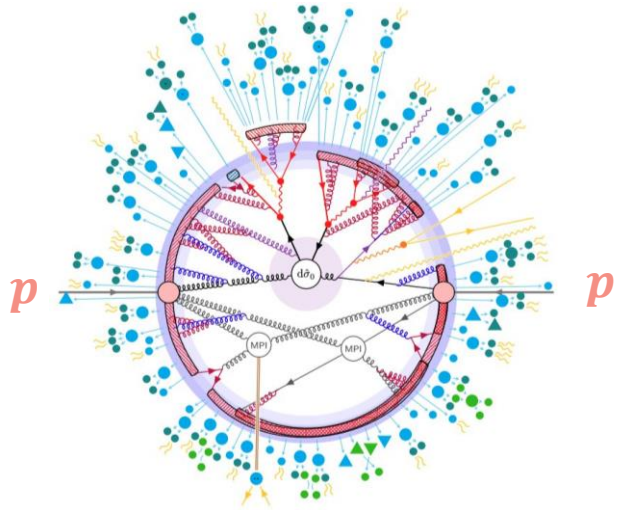
THE UNIVERSITY of EDINBURGH
School of Physics
& Astronomy

MONASH
University

MC Support Tools Workshop
March 17, 2026

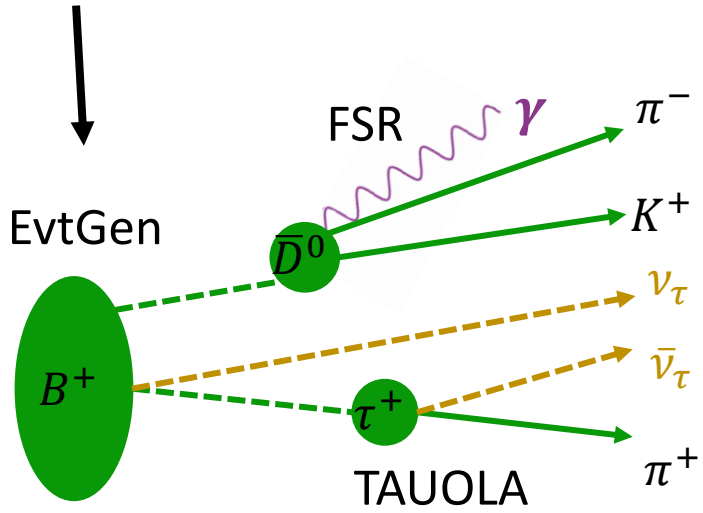
EvtGen in a nutshell

Example collision simulated by PYTHIA8



Simulation generator package specialised for decays of heavy particles containing b and c quarks.

- Contains 130 decay models for specific decays
- Maintains decay table with properties of $\sim 10^4$ explicit decays



External dependencies

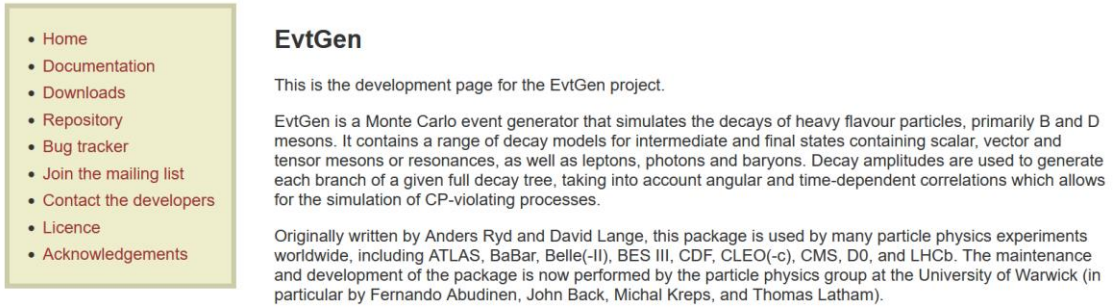
- Uses Pythia8 for decays of generic quark configurations
- Uses TAUOLA for decays of τ particles
- Relies on PHOTOS or PHOTONS++ for final-state photon radiation (FSR)

EvtGen status

- Developed in the 90's, stable over past 10 years
- Changes mostly additions of new models by different collaborators
- Maintained at Warwick since 2012 following merge of various experiment's forks by Anders Ryd
- Recently accomplished first **source-code modernisation** with help of research-software engineers at Warwick

⇒ Main goal of recent campaign: **enable thread safety**

https://evtgen.hepforge.org

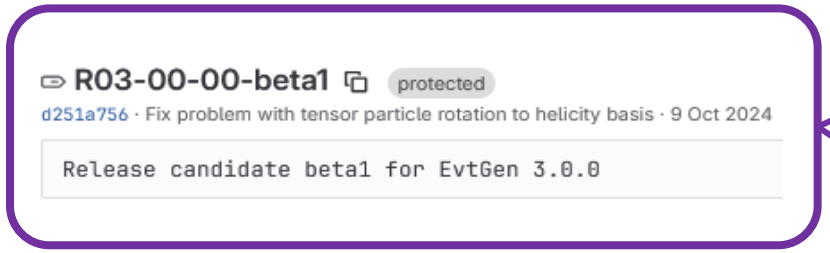


Mirror at https://gitlab.cern.ch/evtgen/evtgen
with continuous integration tests

Recent developments

- Propagated thread safety across the whole software framework
- Implemented general testing framework (with optional multi-threading)
- Fixed bug in tensor rotation
- Added SHERPA's PHOTONS++ for QED radiation

Included in



Since then

- VINCIA for QED radiation
- Bugfixes ($B \to X_s\gamma$ model, $B \to X_s\,\tau\ell$)
- Inclusion of Belle II decay table
- Refactoring of testing framework

Forwarded to ATLAS for testing

Started new project on decay table weighting

Incorporated into LHCb simulation

Recap: implementation of multithreading

With help of research-software engineers:
Heather Ratcliffe, Chris Brady

Challenges

- Internal: structural limitations
  - Global instance of random number generator
  - Global instance of particle properties and decay table
- External: limitations from dependences (look for alternatives) on TAUOLA and PHOTOS

Current solution

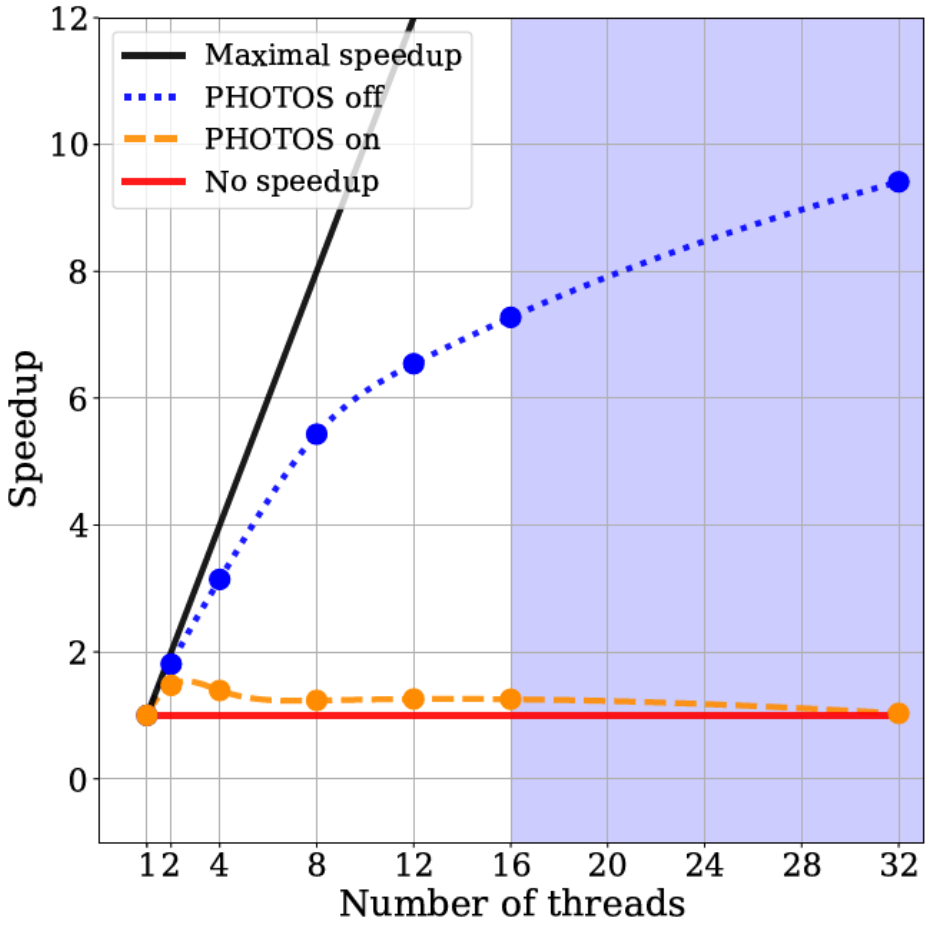
1st Campaign with SWIFTHEP support

- Static objects made constant or thread-local
- Global singleton objects made thread-local
- Serialized (mutexed) calls to PHOTOS and TAUOLA

⇒ Passes all tests

⇒ Performance limited by external dependencies

⇒ Study alternatives



Final-state radiation in EvtGen

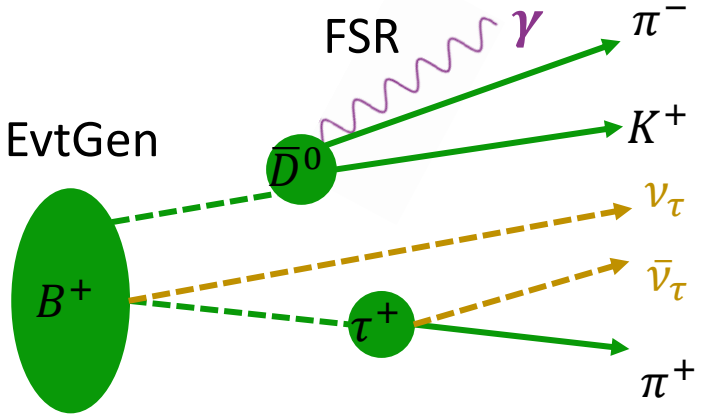
- EvtGen relies on external specialised generators to add QED FSR corrections

Goals

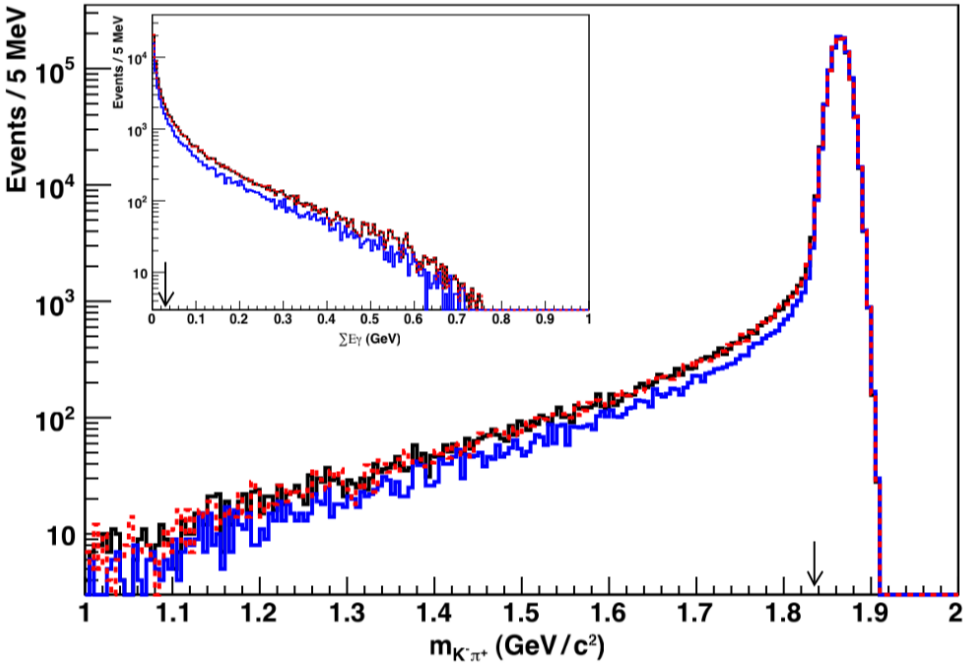
- Find alternatives to study systematic effects

⇒ Especially those associated with interference effects

- Find alternatives to exploit multithreaded processing



$D^0 \to K^+\pi^-$ **simulation with PHOTOS**



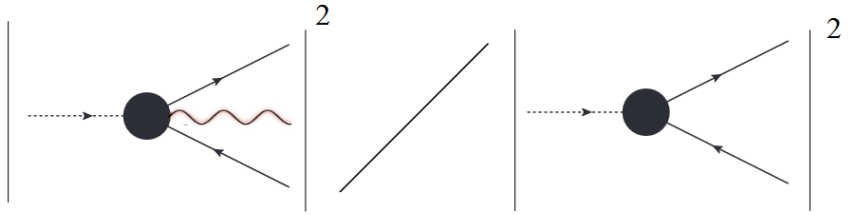
- Default generator is PHOTOS
- Recently included Sherpa's PHOTONS++ and
- Vincia (inside Pythia8) as alternative

Theory approaches in a nutshell

- Treat the effect of FSR as a correction to the Born-level decay rate (or cross section)

$$\mathrm{d}\Gamma^{\mathrm{radiative}} = \mathrm{d}\Gamma^{\mathrm{Born}}\, f(\Phi)\, \mathrm{d}\Phi \qquad \Phi\text{: Phase-space of photons}$$

- Example (oversimplified): neutral scalar $\rightarrow e^+e^-$ (single QED dipole)



$$\propto \underbrace{\frac{2s_{e^-e^+}}{s_{e^-\gamma}s_{\gamma e^+}}}_{\text{eikonal term}} + \frac{1}{M_0^2}\underbrace{\left(\frac{s_{\gamma e^+}}{s_{e^-\gamma}} + \frac{s_{e^-\gamma}}{s_{\gamma e^+}} + 2\right)}_{\text{collinear terms}}$$

LO QED

$s_{ij} = 2\, p_i \cdot p_j$
Dot product of 4-momenta

PYTHIA, PHOTOS

DGLAP

e^--collinear limit

e^+-collinear limit

$$\frac{P_{e^-\rightarrow e^-\gamma}(z_1)}{s_{1\gamma}} + \frac{P_{e^+\rightarrow e^+\gamma}(z_2)}{s_{2\gamma}}$$

VINCIA

Antenna

All Singular Terms

$$\frac{2s_{e^-e^+}}{s_{e^-\gamma}s_{\gamma e^+}} + \frac{1}{M_0^2}\left(\frac{s_{\gamma e^+}}{s_{e^-\gamma}} + \frac{s_{e^-\gamma}}{s_{\gamma e^+}}\right)$$

Soft limit

HERWIG, SHERPA

YFS

$$\frac{2s_{e^-e^+}}{s_{e^-\gamma}s_{\gamma e^+}}$$

Generators:
add photons
based on $f(\Phi)$

Final-state radiation generators

PHOTOS Barberio-Was 1991, Nanava-Was 2007, Davidson-Przedzinski-Was 2015

- Based on collinear approximation (LO), determines sets of dipoles (assuming spin-1/2)
- Soft (interference) effects and spin dependence through correction weight (NLO for scalar decays)

YFS Yennie-Frautschi-Suura 1961, Krauss-Schönherr 2008 **(basis for Herwig and Sherpa's PHOTONS++)**

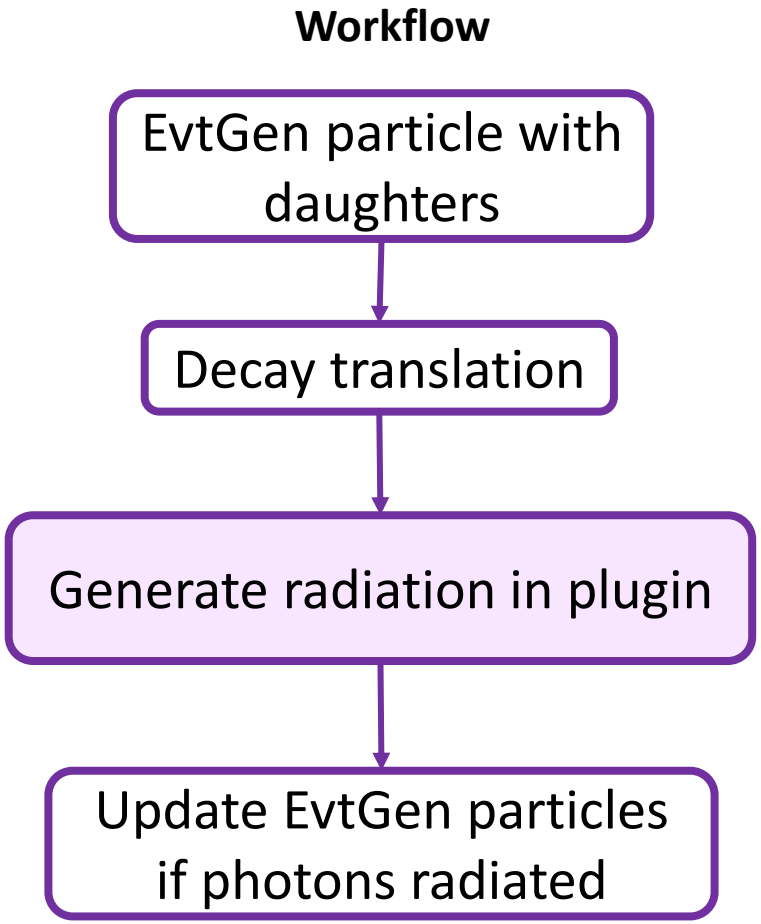
- Takes full (multipole) soft interference effects into account
- Scalar QED ⇒ spin dependence through Matrix-Element corrections to NLO

Vincia QED Kleiss-Verheyen 2017, Brooks-Verheyen-Skands 2020

- Parton shower evolution based on antenna approximation (can be interleaved)
- Takes full (multipole) soft interference effects into account
- Not limited to scalar QED (includes spin dependence)

Interfaces between EvtGen and Plugins

- Each decay-chain node translated
  - Into intermediate HepMC events (for PHOTOS)
  - Directly into Sherpa or Pythia objects (for Photons and Vincia)
- EvtGen random number propagated (full seed control)
- PHOTOS and Sherpa's PHOTONS++ not thread-safe yet ⇒ mutex
  - Need to mutex also HepMC translation (for PHOTOS)



Review (for Sherpa) by Marek Schönherr and Frank Krauss

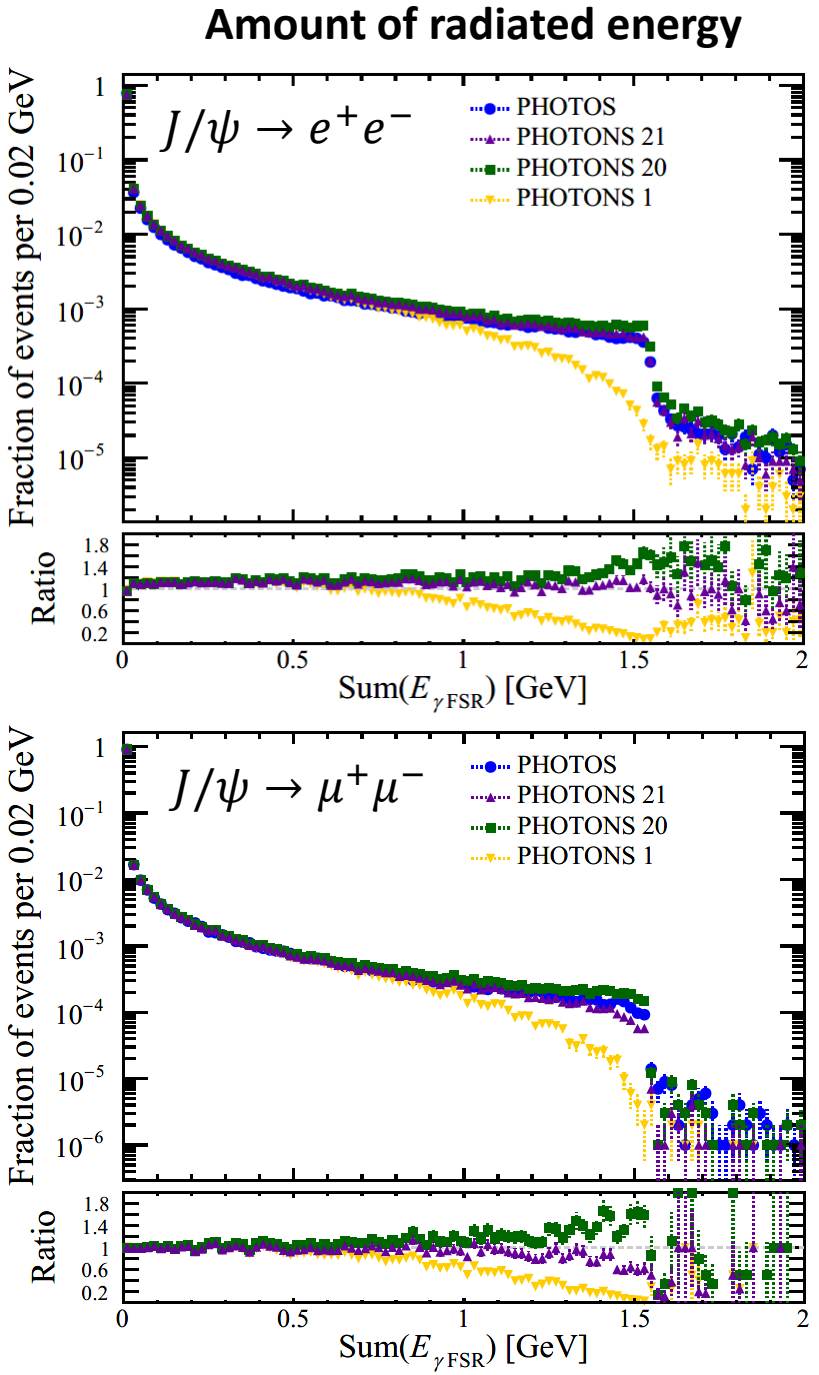
Supporting Sherpa 2.2.15 and **new 3.x.x releases**

Sherpa's PHOTONS++ for FSR

- PHOTONS++ in Sherpa can simulate emission of soft photons based on YFS approximation (mode 1)
- If switched on also hard photons based on collinear approximation (mode 2)
  - Approx. matrix-element corrections (mode 20) or
  - Exact matrix-element corrections (mode 21)
- With mode 1: fewer hard photons compared to PHOTOS (PHOTOS has matrix-element corrections implemented)
- With mode 2: generally good agreement with PHOTOS

⇒ Implemented switches for systematic studies

New in EvtGen R03-00-00-beta1!



Challenges for interfacing

- Need to create full Pythia/Sherpa objects
- Several initialised objects are not used

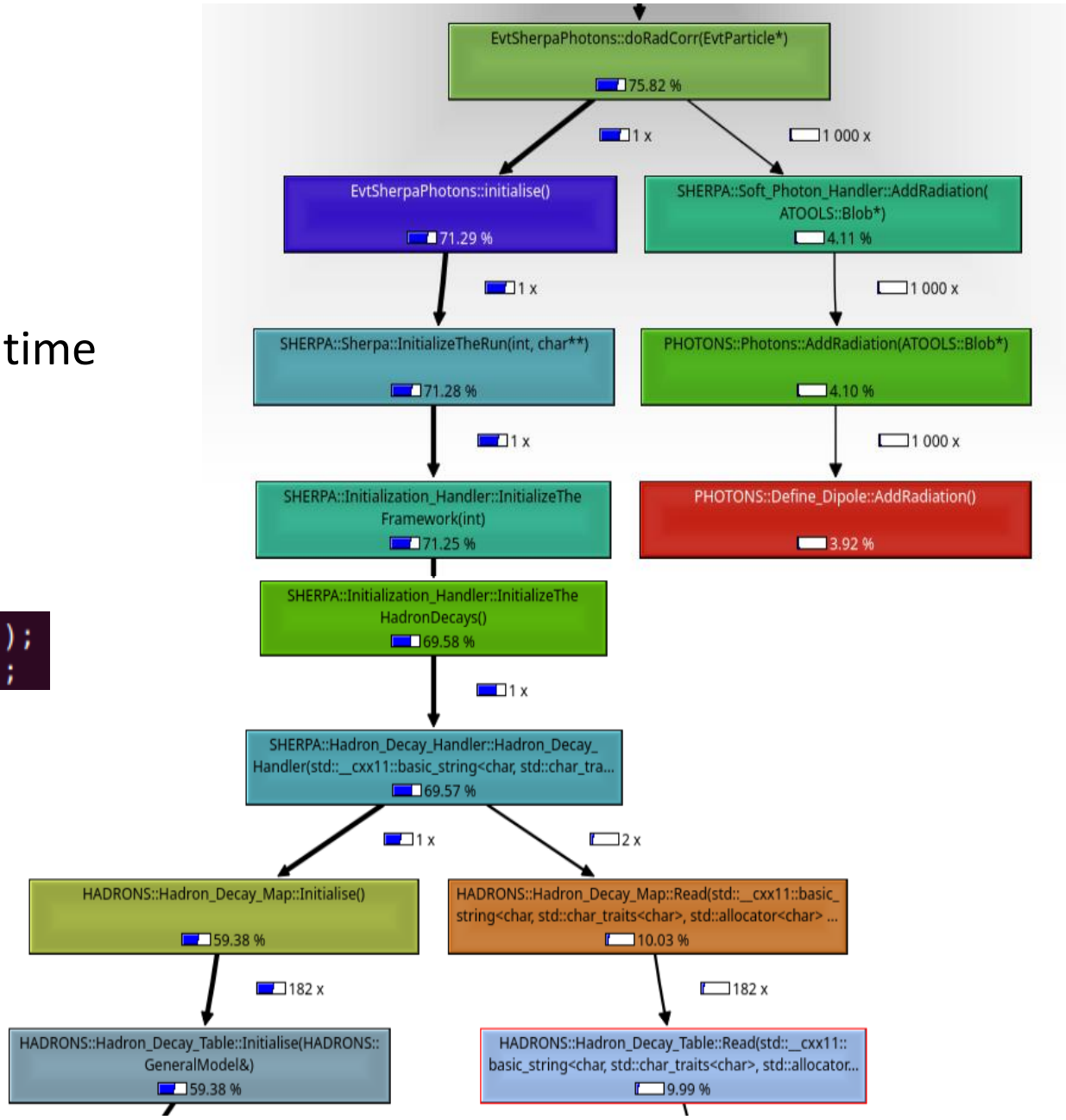
  Example: Sherpa's initialisation takes as much time as $\sim 10^4$ decay events
- Need to control destruction of static instances

Users must call this function

```
EvtSherpaPhotons::finalise();
```

```
m_photonsGen.reset();
m_sherpaGen.reset();
```

Currently discussing improvements with Sherpa authors



Intermezzo: Work on VINCIA

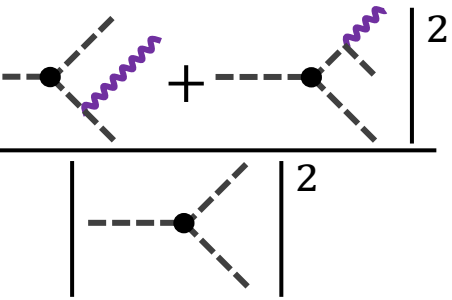
Vincia QED shower for FSR

- Recently adapted to radiate off hadrons (previously supporting only leptons)
- Matrix-element corrections (MECs) in progress
  - Use FeynRules to model hadron decays (produces universal FeynRules output file)
  - Generate tree-level NLO ME using Madgraph (produces plugin)

  ⇒ Use it for Antenna function

Giacomo Morgante

$$a_{\text{emit}} = \frac{|M_{n+\gamma}|^2}{|M_n|^2} =$$



Technical aspects

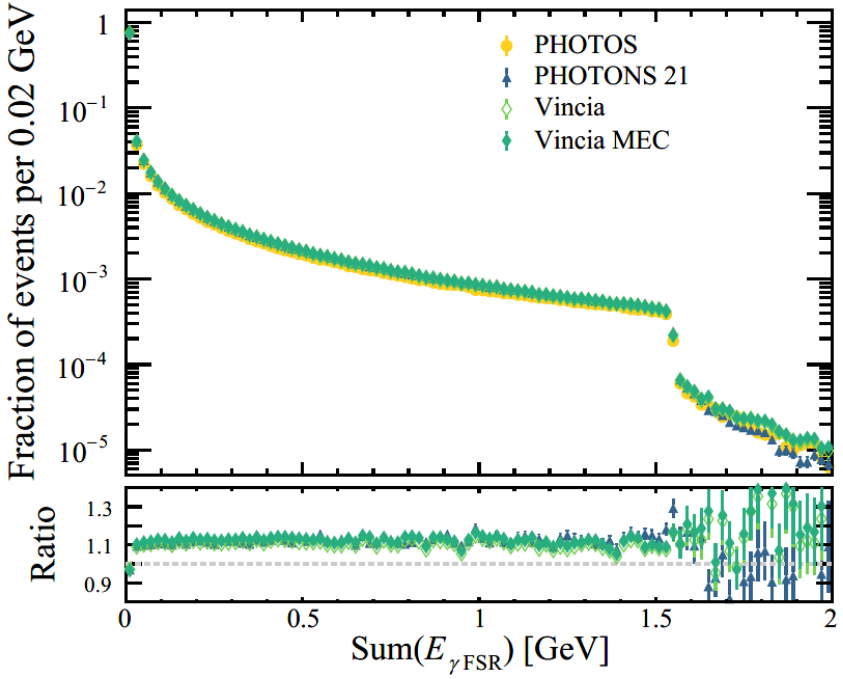
Vincia embedded in Pythia8 → Implementation enables thread safety → EvtGen ↔ Vincia interface based on existing dependency with Pythia8 → Will be part of EvtGen release 3

Comparisons between generators

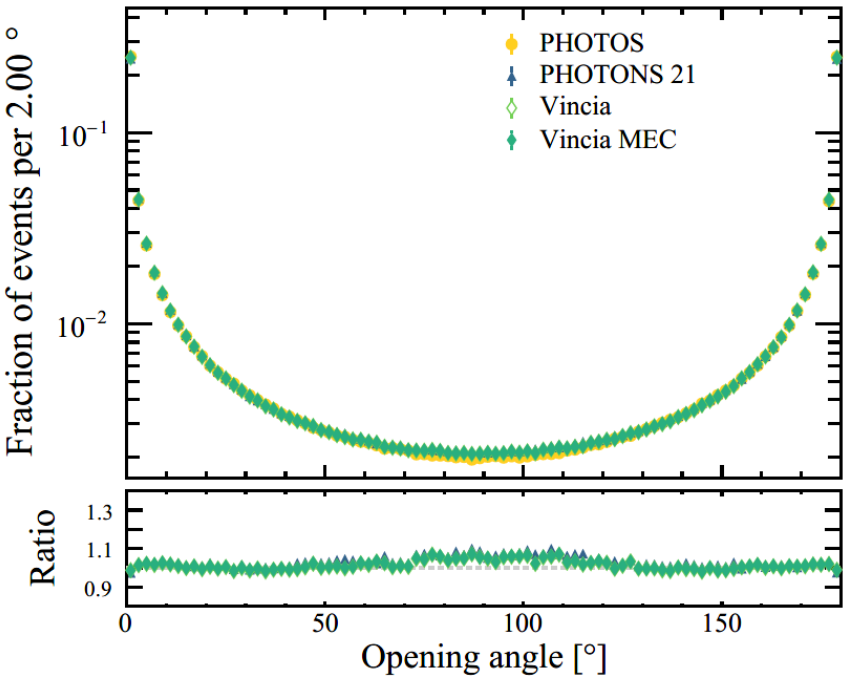
- Equalise photon p_{T}(Energy) cutoff value (0.1 keV)
- Consider photons only if energy above 0.1 MeV

$$J/\psi \rightarrow e^{+}e^{-}$$

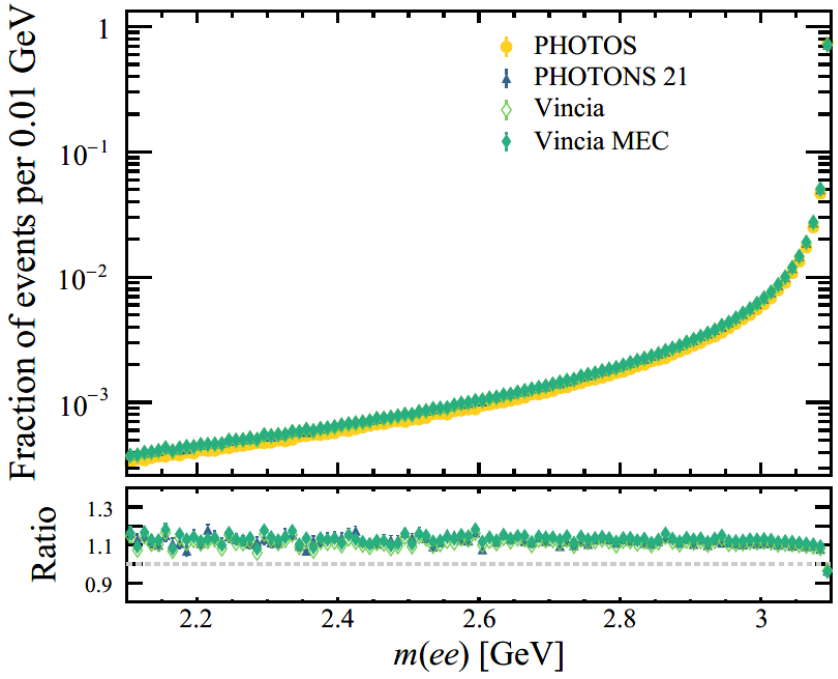
Amount of radiated energy



Angular distribution of photons



Dielectron mass

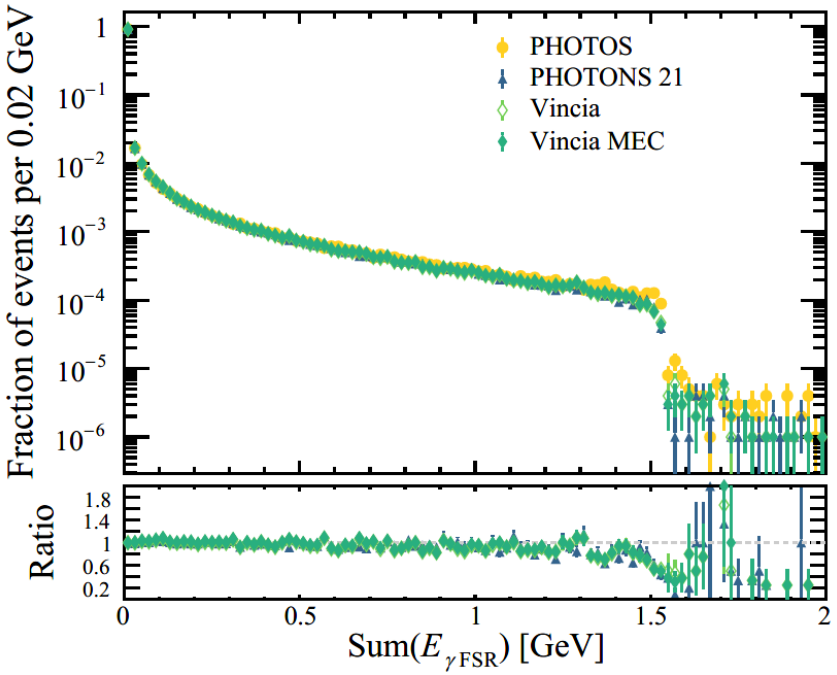


- Good agreement (within ~10%) for energy and angular distributions
- All generators radiate more photons that PHOTOS

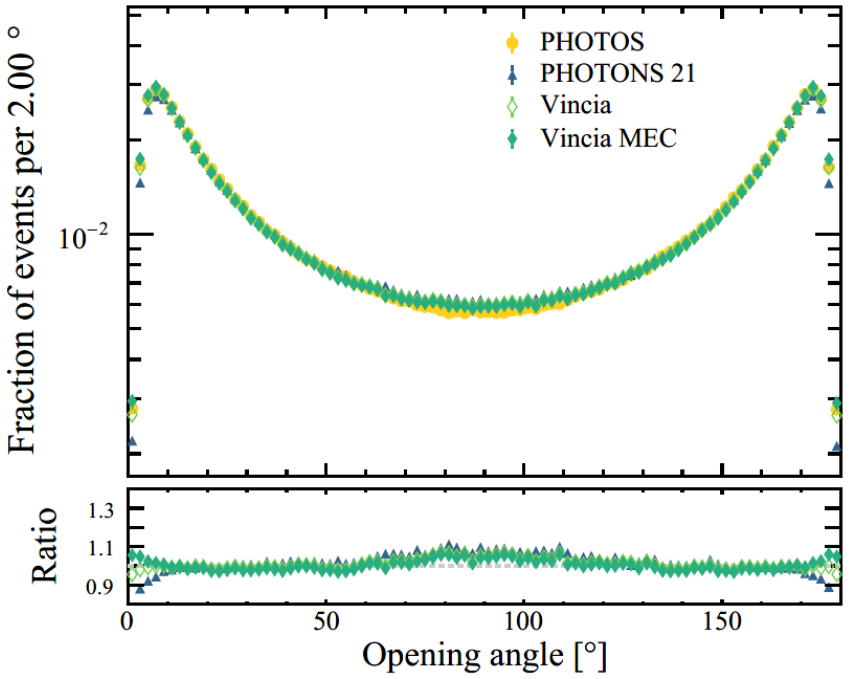
Comparisons between generators

$J/\psi \to \mu^+\mu^-$

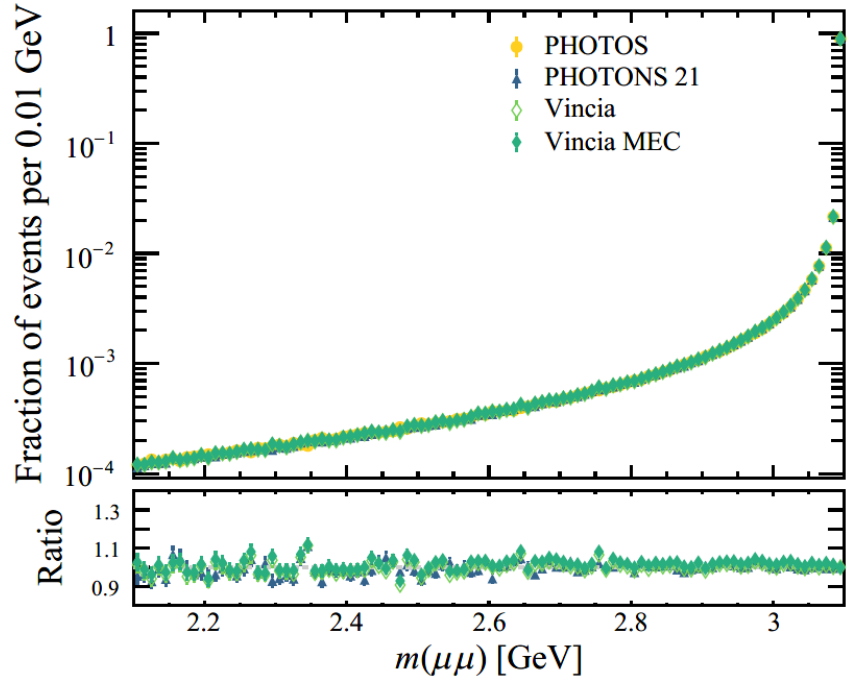
Amount of radiated energy



Angular distribution of photons



Dimuon mass

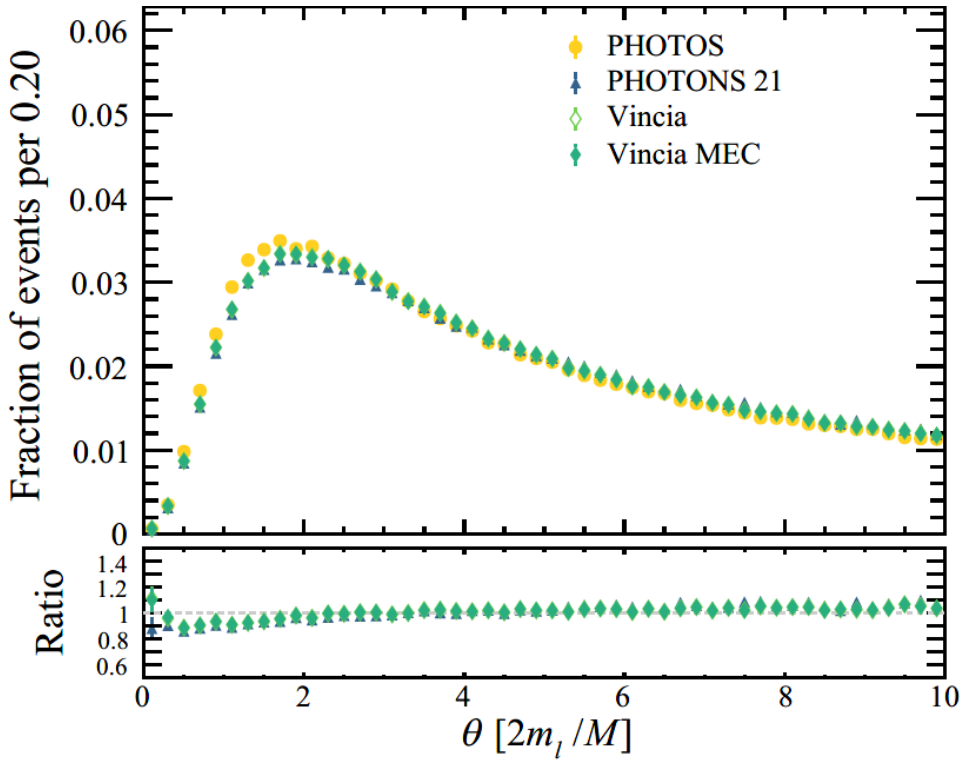


- Good agreement among generators

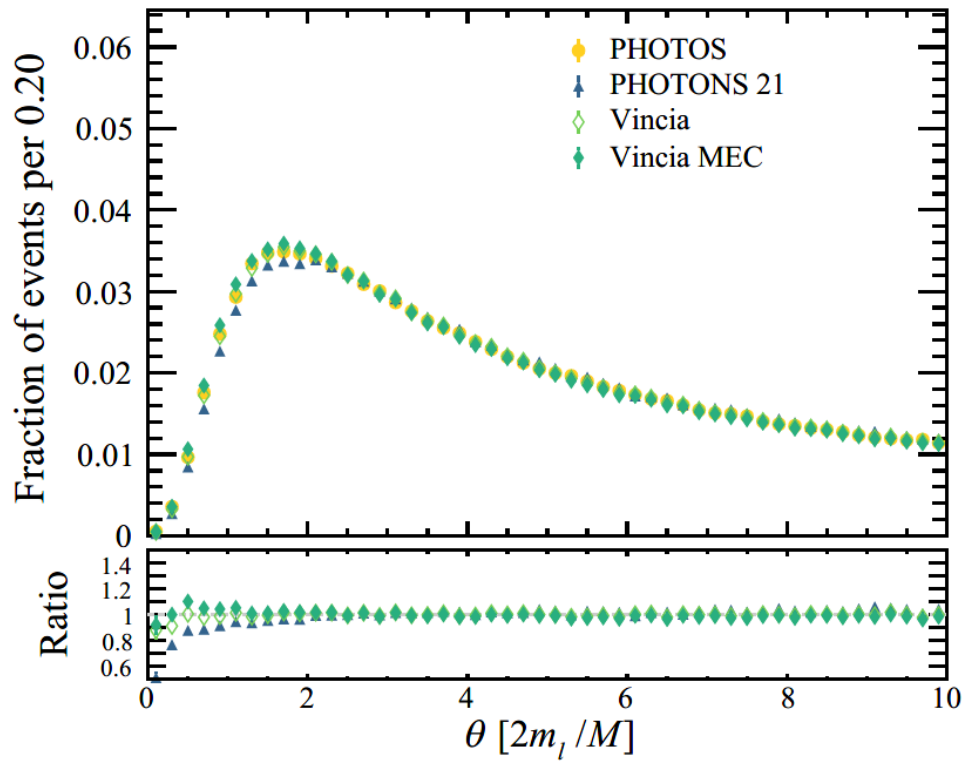
Dead cone angle

Combination of phase space and mass corrections ⇒ dead cone for $\theta \lesssim \frac{m}{E}$

$J/\psi \to e^+e^-$



$J/\psi \to \mu^+\mu^-$

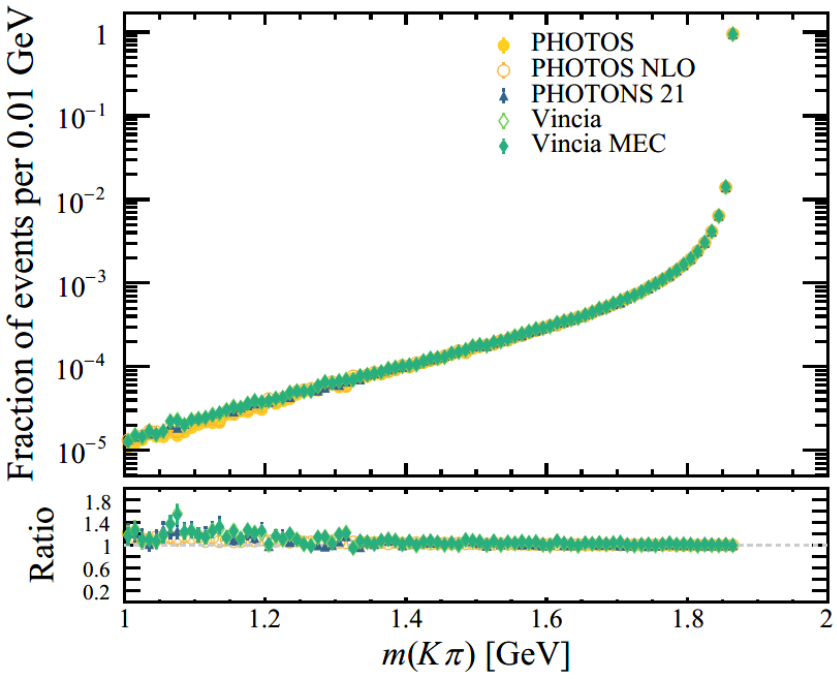


- Electrons: dead cone slightly narrower with PHOTOS
- Muons: dead cone slightly wider with PHOTONS++

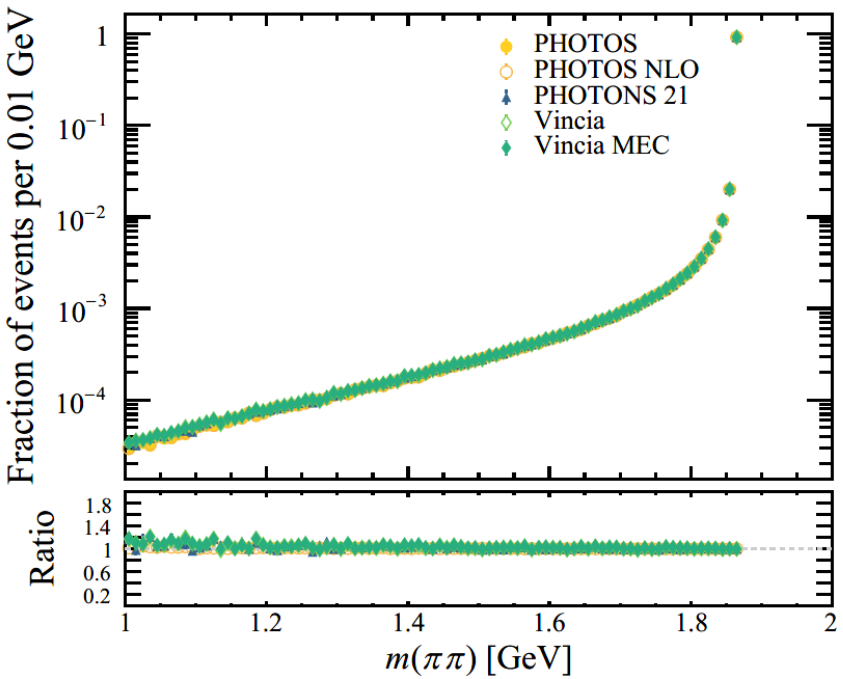
Comparisons between generators

Dihadron invariant mass

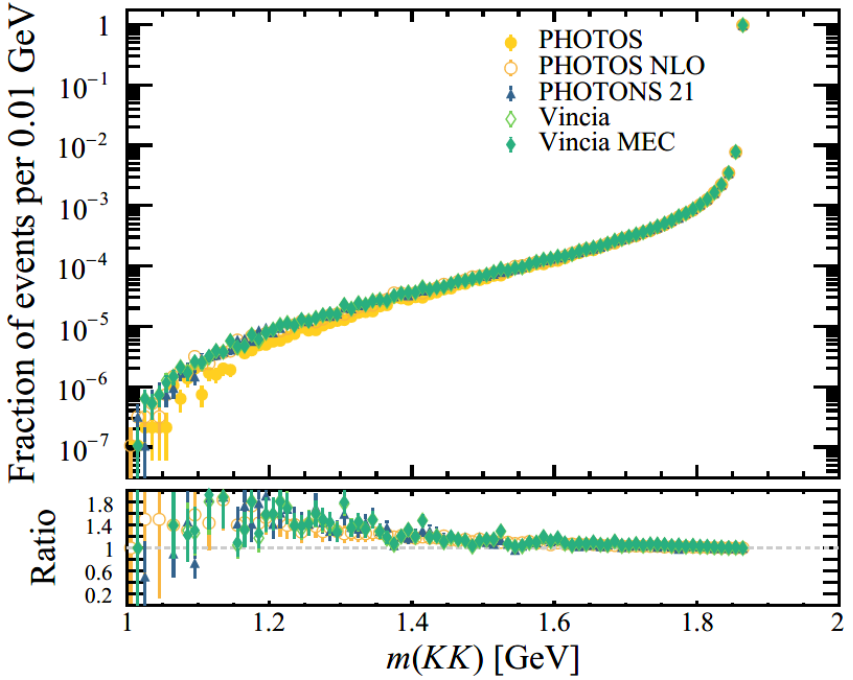
$D^0 \to K^-\pi^+$



$D^0 \to \pi^+\pi^-$



$D^0 \to K^+K^-$

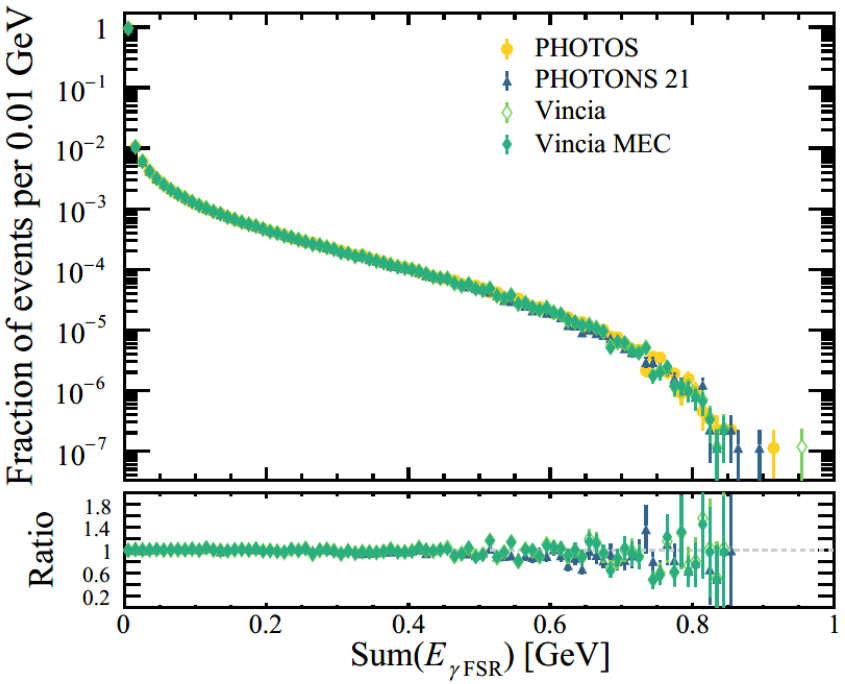


- Excellent agreement among generators, especially with NLO ME corrections

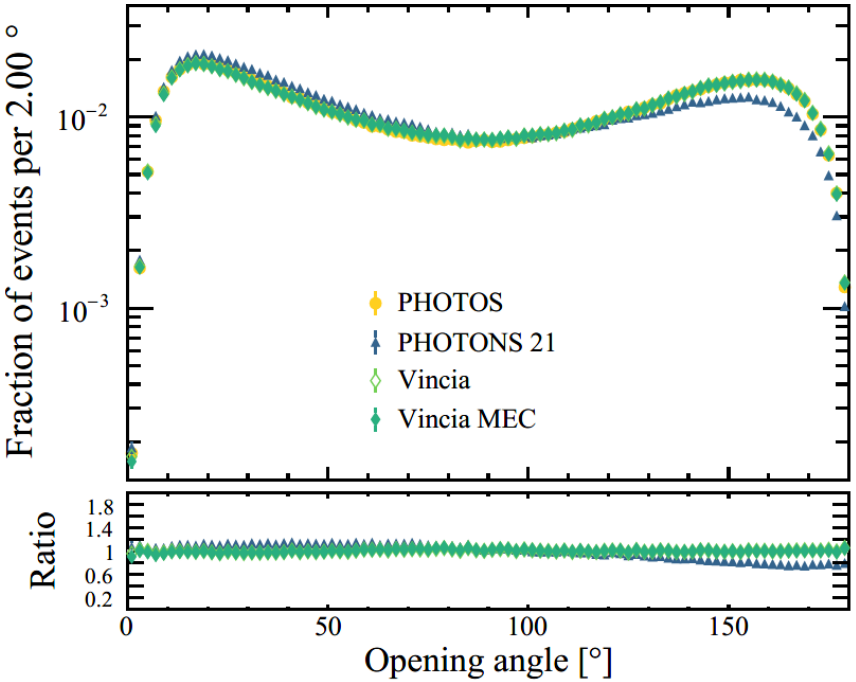
Three-body study

$$D^+ \to \pi^+\pi^-\pi^+$$

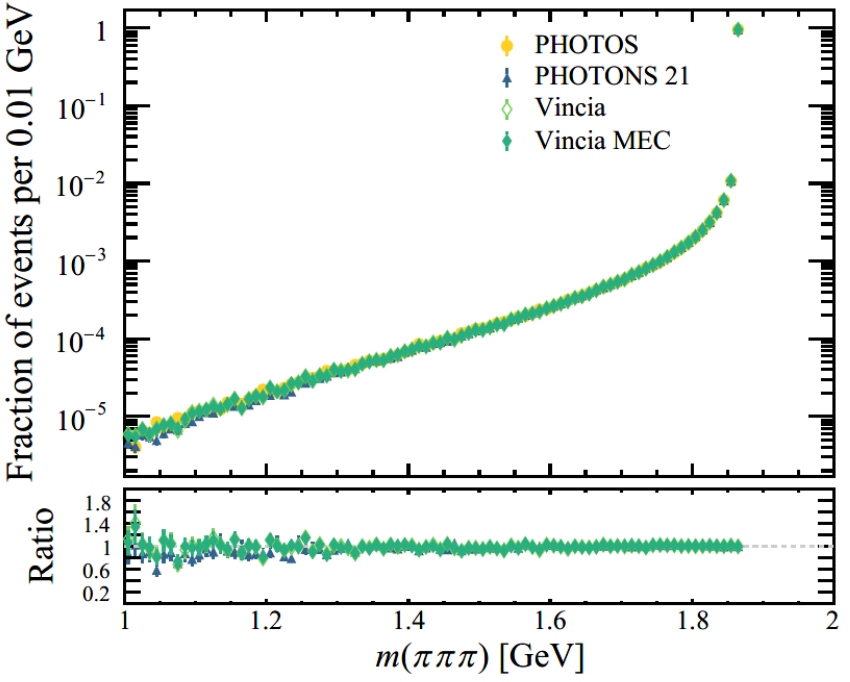
Amount of radiated energy



Angular distribution of photons



Tripion mass



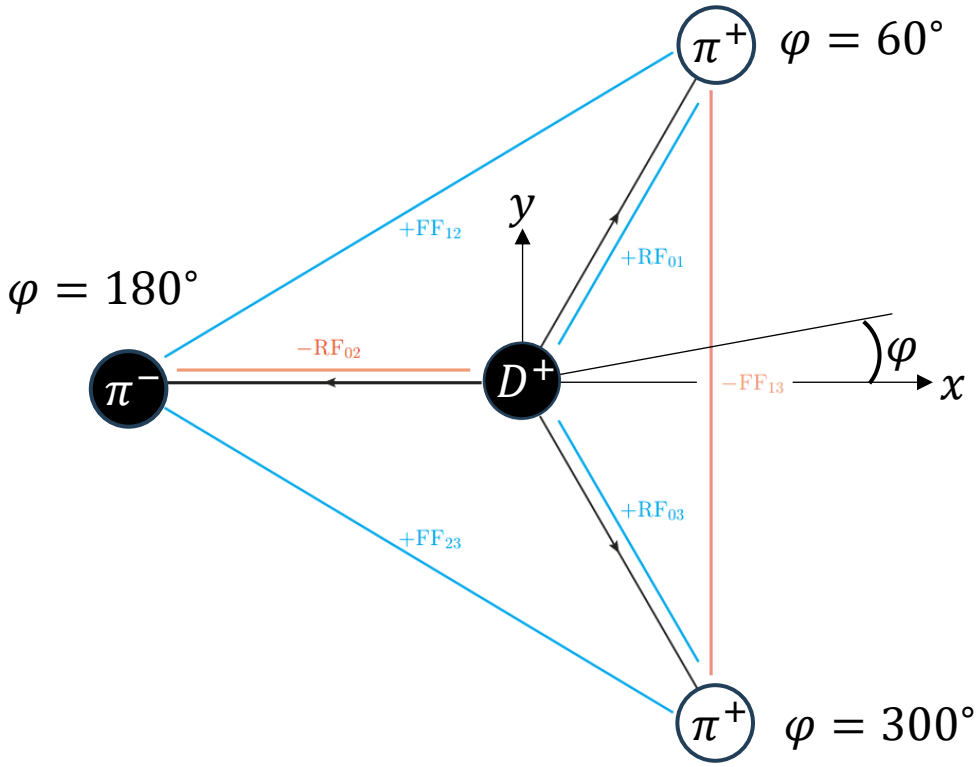
Angle with respect to hardest particle

- Good agreement between generators
- PHOTONS++ tends to radiate fewer photons in backwards direction (to be checked)

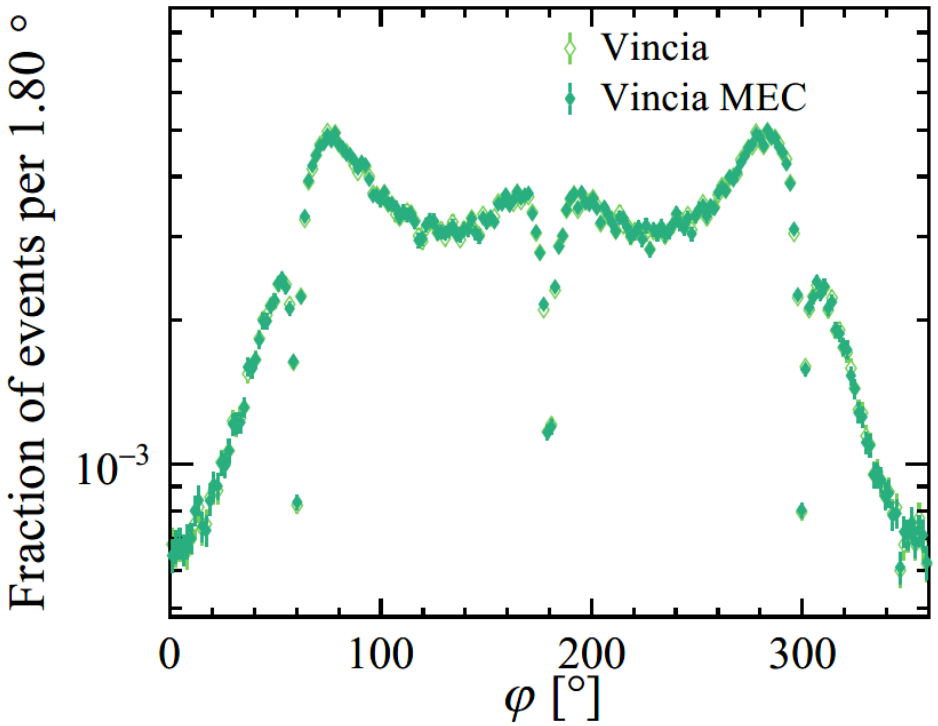
Three-body study

Enforce a specific geometric configuration

Generated geometric configuration



Angular distribution of radiated energy

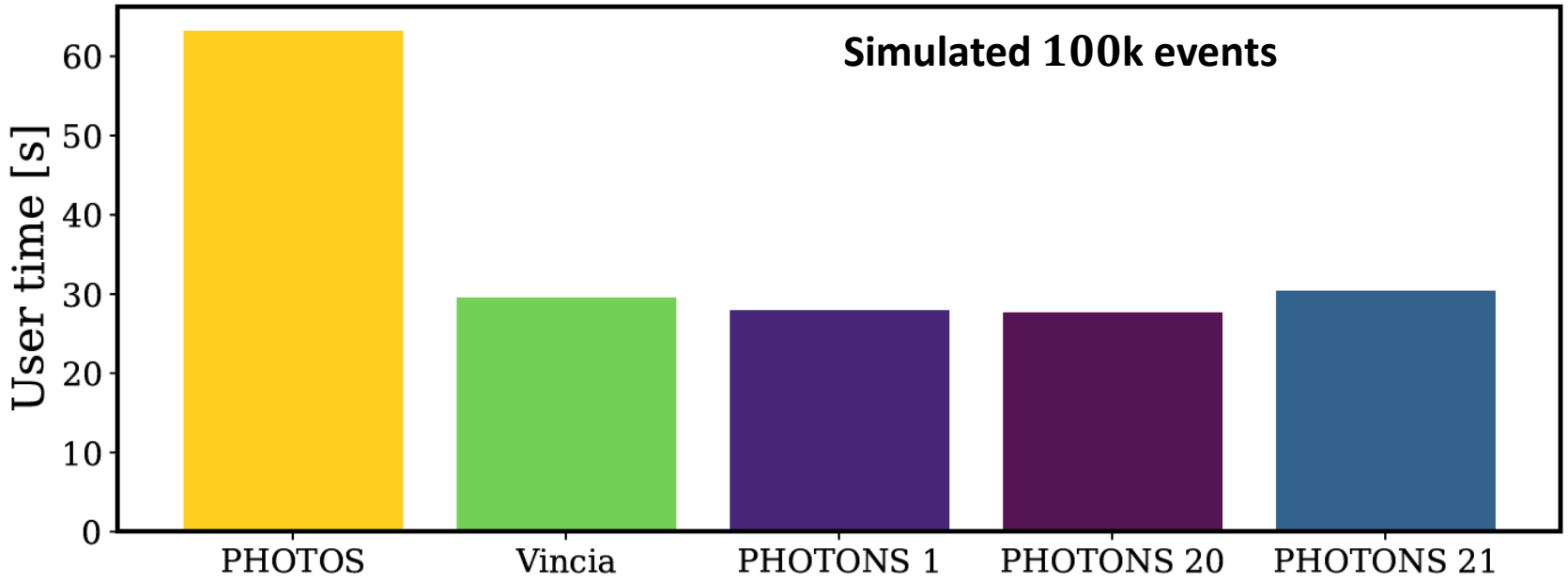


$\Rightarrow$ Expected decrease in radiation collinear with decay products in parent's frame

A word on timing

- Compare simulation time when simulating generic $\Upsilon(4S) \rightarrow B\bar{B}$

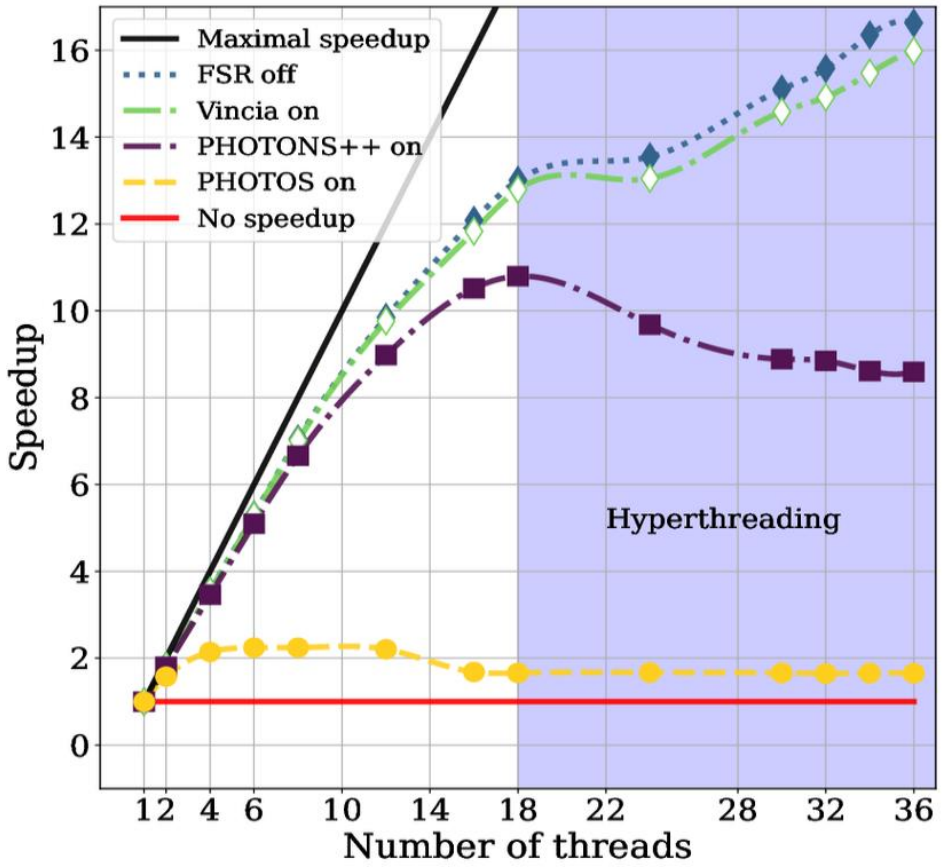
$\Rightarrow$ Benchmark for general use

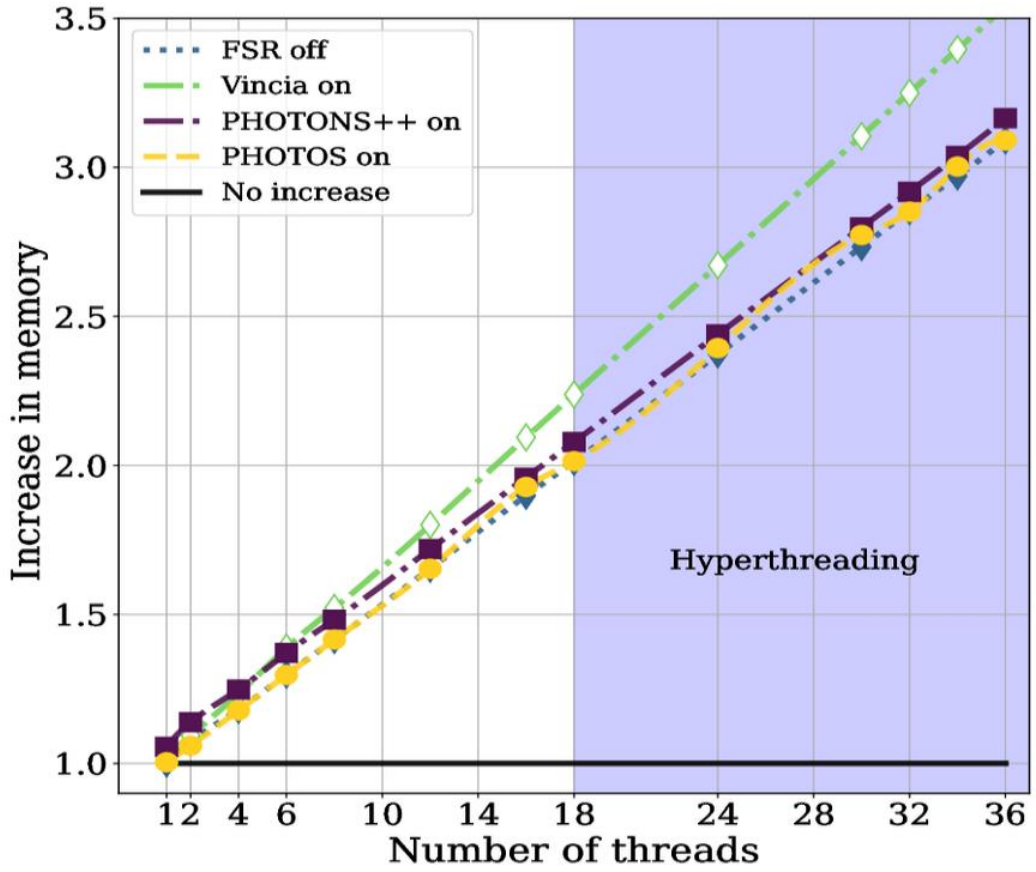


$\Rightarrow$ No large difference between PHOTONS options in generic case

$\Rightarrow$ Potential speedup using Vincia or PHOTONS by about factor 2

Multithreaded processing





⇒ Best performance with Vincia

⇒ SHERPA limited by initialisation overhead and mutex for static configs

Back to EvtGen

Integration into LHCb Gauss framework

Necessary modifications

- Make EvtGen object thread local
- Implement initialisation per thread
- Make particle data list (PDL) and Decay Table thread local

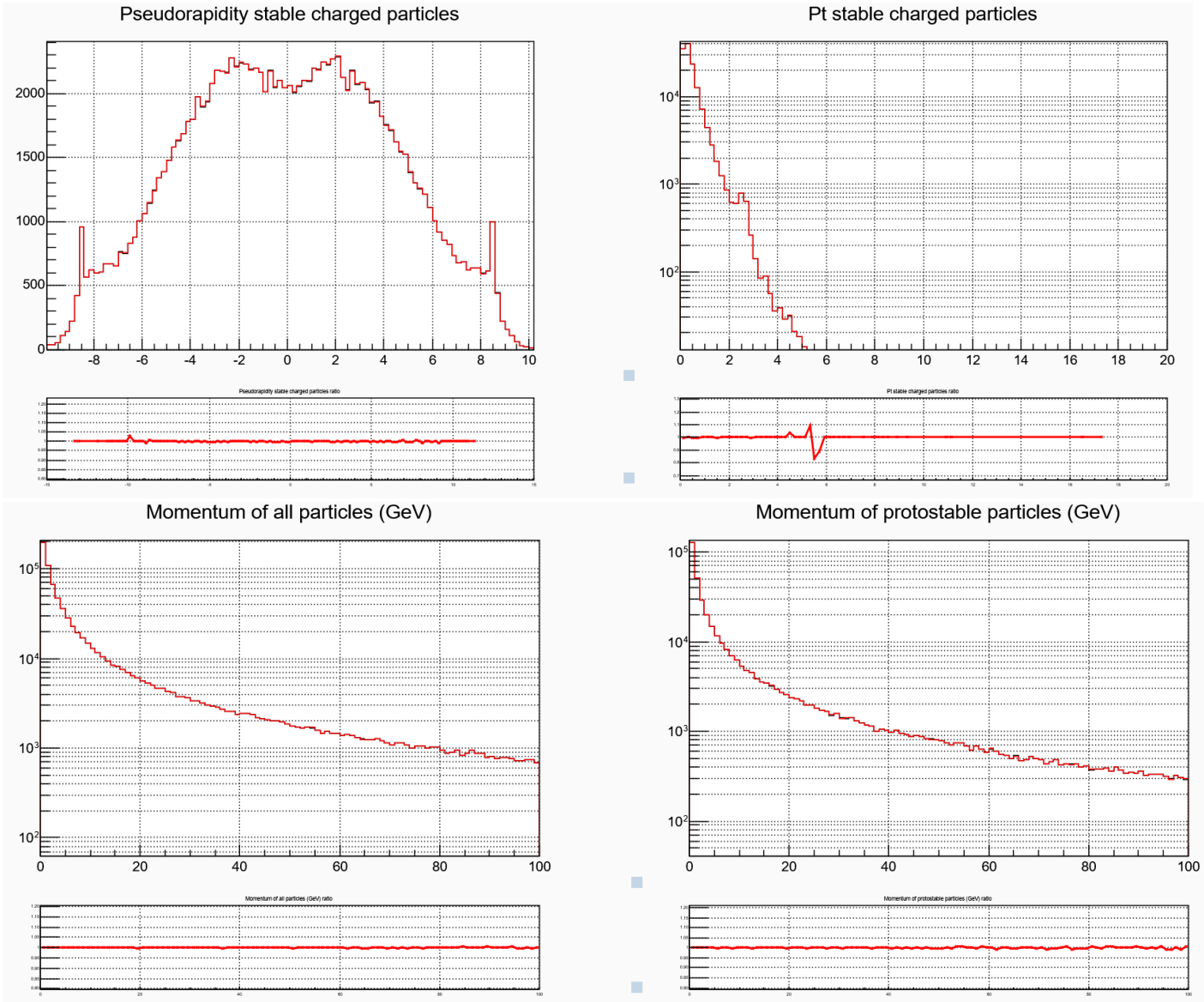
⇒ But read from same input files

- Propagate const correctness within private decay modules
- Remove mutex

Passed LHCb Sim11 tests

lhcb/Gauss!1107

LHCB Sim11 validation



Decay Table variations

Example from general decay table

```
Decay anti-B0

#                    b -> c semileptonic
#  BFs                                                  Models and configs
0.0493    D*+     e-      anti-nu_e      FSR  HQET2 1.207 0.920 1.406 0.853;
0.0219    D+      e-      anti-nu_e      FSR  HQET2 1.185 1.081;
0.0042    D_1+    e-    anti-nu_e        FSR  ISGW2;
0.0045    D_0*+   e-    anti-nu_e        FSR  ISGW2;
0.0046    D'_1+   e-    anti-nu_e        FSR  ISGW2;
0.0033    D_2*+   e-    anti-nu_e        FSR  ISGW2;
0.00045   D*+  pi0   e-    anti-nu_e      FSR  GOITY_ROBERTS;
0.00490   D*0   pi+  e-    anti-nu_e      FSR  GOITY_ROBERTS;
0.0015    D+   pi0   e-    anti-nu_e      FSR  GOITY_ROBERTS;
0.0043    D0   pi+   e-    anti-nu_e      FSR  GOITY_ROBERTS;
```

General decay table

- Ignores **uncertainties on branching fractions**
- Users can have **custom versions**
- Currently no support for **variations**

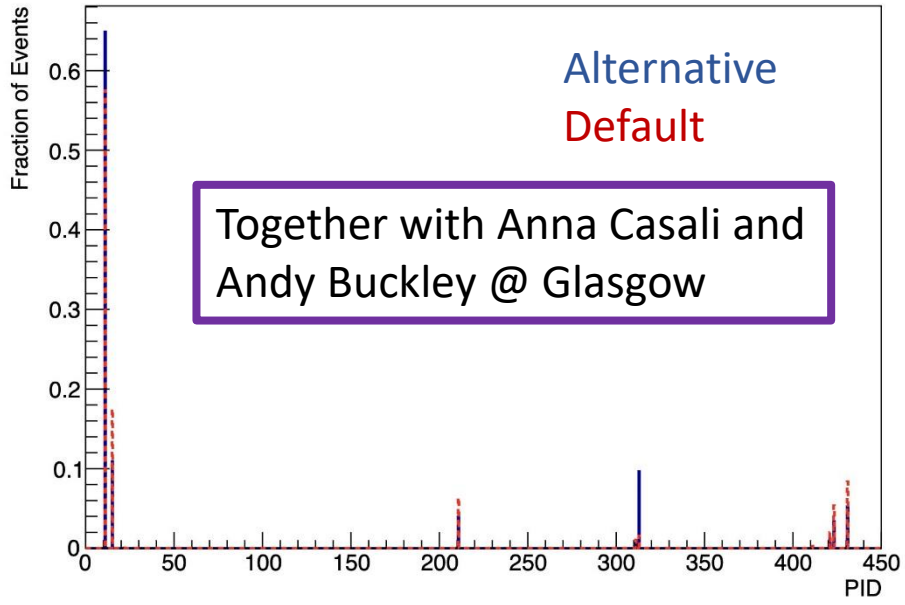
Idea: provide decay weights for systematic variations

See talk by Anna!

Challenges

- Inconsistencies in lists, PDG BFs not adding up to 1, theory uncertainties (form factors, etc)
- Flexibility to apply custom weighting (e.g. beauty, charm, …)
- Calculate alternative amplitude for generated kinematic config
- Propagate alternative spin-density matrix through decay chain

Simulation with two decay tables



Together with Anna Casali and Andy Buckley @ Glasgow

evtgen!47

Belle II decay table

Introduced Belle II Decay Table within EtvGen.

Necessary additions

New form factors for semileptonic B decays

- LLSW
- HQET3

} Added to existing BTOXELNU wrapper

- New models for three-body $\eta^{(\prime)}$ decays
  - ETA_PI0DALITZ
  - ETAPRIME_DALITZ
  - ETA_FULLDALITZ

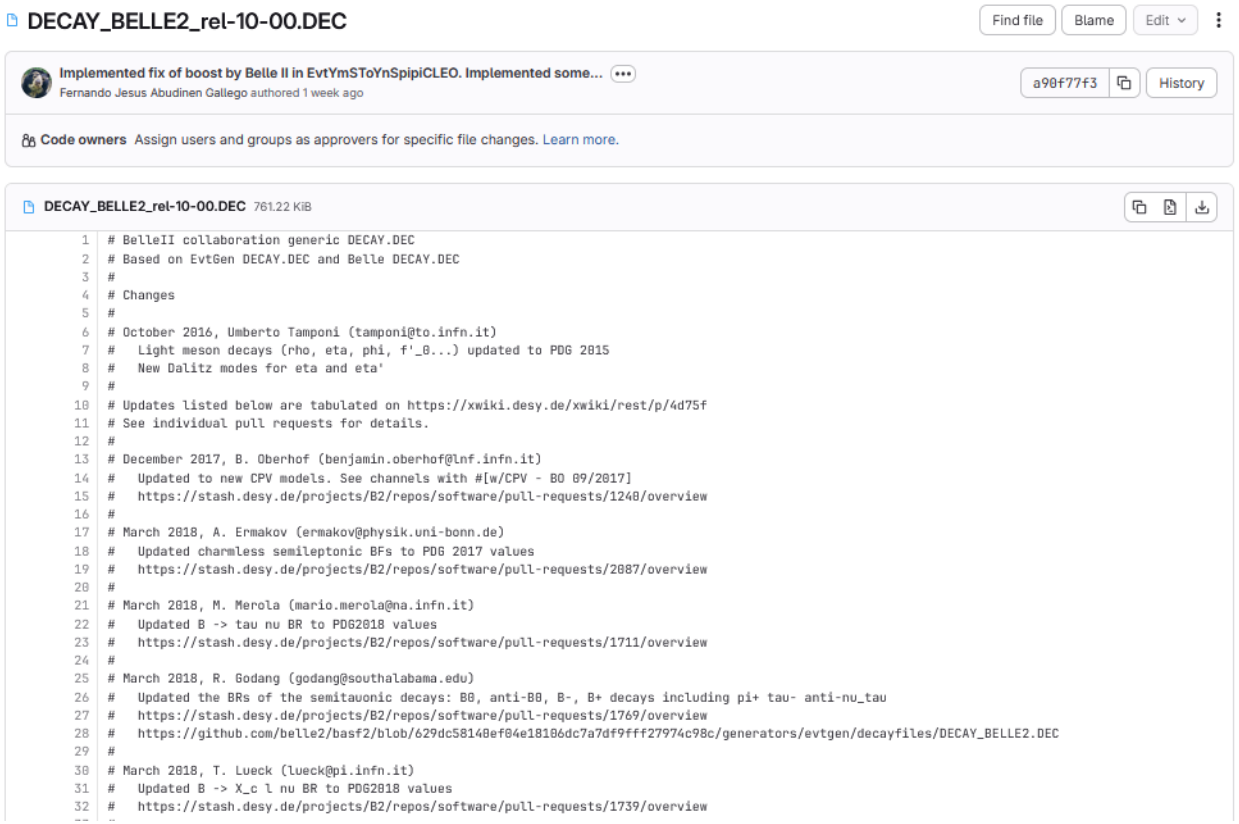
Fixed bug in existing $\Upsilon(mS) \to \Upsilon(nS)\pi\pi$ model

- YMSTOYNSPIPICLEOBOOST

evtgen!53

⇒New tests were added for all new models and from factor cases (scalar, vector, tensor).

⇒Implemented const correctness and improved Doxygen documentation



New $\eta^{(\prime)}$ decay models

Turn “Boilerplate” into models deriving from base class.

$\eta \to \pi\pi\pi^0$

```
p->initializePhaseSpace(getNDaug(), getDaugs());

EvtVector4R mompi0_0 = p->getDaug(0)->getP4();
EvtVector4R mompi0_1 = p->getDaug(1)->getP4();
EvtVector4R mompi0_2 = p->getDaug(2)->getP4();

double m_eta = p->mass();
double m_pi0 = p->getDaug(0)->mass();
double deltaM = m_eta - 3 * m_pi0;

//The decay amplitude comes from KLOE collab., Phys.Lett. B694 (2011) 16-21
double z0 = (3 * mompi0_0.get(0) - m_eta) / deltaM;
double z1 = (3 * mompi0_1.get(0) - m_eta) / deltaM;
double z2 = (3 * mompi0_2.get(0) - m_eta) / deltaM;

double z = (2. / 3.) * (z0 * z0 + z1 * z1 + z2 * z2) ;

double Amp2 = 1.0 + alpha * 2.0 * z;
if (Amp2 < 0)
  Amp2 = 0;

EvtComplex amp(sqrt(Amp2), 0.);

vertex(amp);
```

ETA_PI0DALITZ

$\eta' \to \pi\pi\pi^0$

```
p->initializePhaseSpace(getNDaug(), getDaugs());

EvtVector4R mompip = p->getDaug(0)->getP4();
EvtVector4R mompim = p->getDaug(1)->getP4();
EvtVector4R mometa = p->getDaug(2)->getP4();

double m_etaprime = p->mass();
double m_eta = p->getDaug(2)->mass();
double m_pi = p->getDaug(1)->mass();

//Q value
double deltaM = m_etaprime - 2 * m_pi - m_eta;

//The decay amplitude comes from BESIII collab, Phys.Rev. D92 (2015) 012014

//Kinetic energies T
double Tpip = (mompip.get(0) - m_pi);
double Tpim = (mompim.get(0) - m_pi);
double Teta = (mometa.get(0) - m_eta);

//Dalitz variables
double X = sqrt(3.) * (Tpip - Tpim) / deltaM;
double Y = ((m_eta + 2 * m_pi) / m_pi) * Teta / deltaM - 1. ;

double amp2 = 1. + a * Y + b * Y * Y + c * X + d * X * X;

EvtComplex amp(sqrt(amp2), 0.0);

vertex(amp);
```

ETAPRIME_DALITZ

$\eta^{(\prime)} \to \pi^+\pi^-\pi^0$

```
p->initializePhaseSpace(getNDaug(), getDaugs());

EvtVector4R mompip = p->getDaug(0)->getP4();
EvtVector4R mompim = p->getDaug(1)->getP4();
EvtVector4R mompi0 = p->getDaug(2)->getP4();

double m_eta = p->mass();
double m_pip = p->getDaug(0)->mass();
double m_pi0 = p->getDaug(2)->mass();

//Q value
double deltaM = m_eta - 2 * m_pip - m_pi0;

//The decay amplitude comes from BESIII collab, Phys.Rev. D92 (2015) 012014

//Kinetic energies T
double Tpip = (mompip.get(0) - m_pip);
double Tpim = (mompim.get(0) - m_pip);
double Tpi0 = (mompi0.get(0) - m_pip);

//Dalitz variables
double X = sqrt(3.) * (Tpip - Tpim) / deltaM;
double Y = 3.*Tpi0 / deltaM - 1. ;

double amp2 = 1. + a * Y + b * Y * Y + c * X + d * X * X + e * X * Y + f * Y * Y * Y;

EvtComplex amp(sqrt(amp2), 0.0);

vertex(amp);
```

ETA_FULLDALITZ

Numerically identical results before and after consolidation

⇒ Testing framework in action

PI0 case

```
if ( m_mode == Mode::PI0 ) {
    //The decay amplitude comes from KLOE collab., Phys.Lett. B694 (2011) 16-21.
    const double z0 = ( 3 * p0.get( 0 ) - M ) / Q;
    const double z1 = ( 3 * p1.get( 0 ) - M ) / Q;
    const double z2 = ( 3 * p2.get( 0 ) - M ) / Q;

    const double z = ( 2. / 3. ) * ( z0 * z0 + z1 * z1 + z2 * z2 );

    amp2 = 1.0 + m_params[0] * 2.0 * z;
    if ( amp2 < 0 ) {
        amp2 = 0;
    }
} else {
    const double Yfactor = m_mode == Mode::ETAPRIME ? ( m0 + m1 + m2 ) / m0
                                                    : 3;

    // Calculate Kinetic Energies in the Parent Rest Frame
    // T = E - m (Masses cached in init())
    const double t0 = p0.get( 0 ) - m0;
    const double t1 = p1.get( 0 ) - m1;
    const double t2 = p2.get( 0 ) - m2;

    // Calculate Dalitz variable Y
    const double Y = ( Yfactor * t2 / Q ) - 1.0;

    if ( m_mode == Mode::STANDARD ) {
        //The decay amplitude comes from Layter et al PRD 7 2565 (1973).
        amp2 = 1.0 - 1.07 * Y;
```

Standard

```
    } else {
        // Calculate Dalitz variable X
        const double X = sqrt( 3 ) * ( t0 - t1 ) / Q;

        if ( m_mode == Mode::ETAPRIME ) {
            //The decay amplitude comes from BESIII collab, Phys.Rev. D92 (2015) 012014.
            amp2 = 1. + m_params[0] * Y + m_params[1] * Y * Y +
                   m_params[2] * X + m_params[3] * X * X;
```

ETAPRIME

```
        } else if ( m_mode == Mode::FULL ) {
            //The decay amplitude comes from BESIII collab, Phys.Rev. D92 (2015) 012014.
            amp2 = 1. + m_params[0] * Y + m_params[1] * Y * Y +
                   m_params[2] * X + m_params[3] * X * X +
                   m_params[4] * X * Y + m_params[5] * Y * Y * Y;
        }
```

FULL

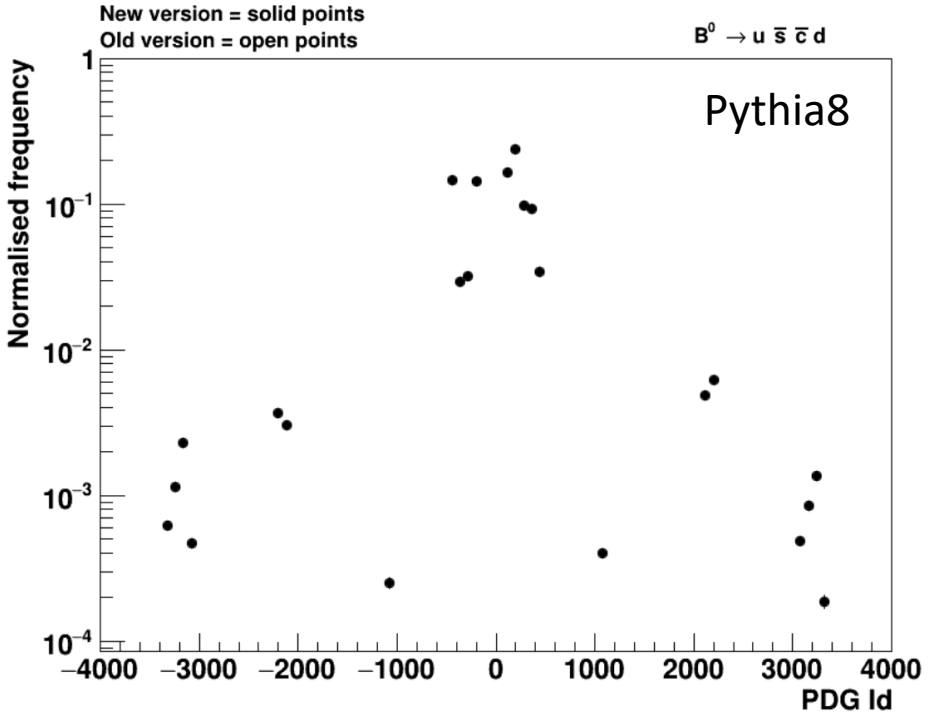
EvtGen Validation tests

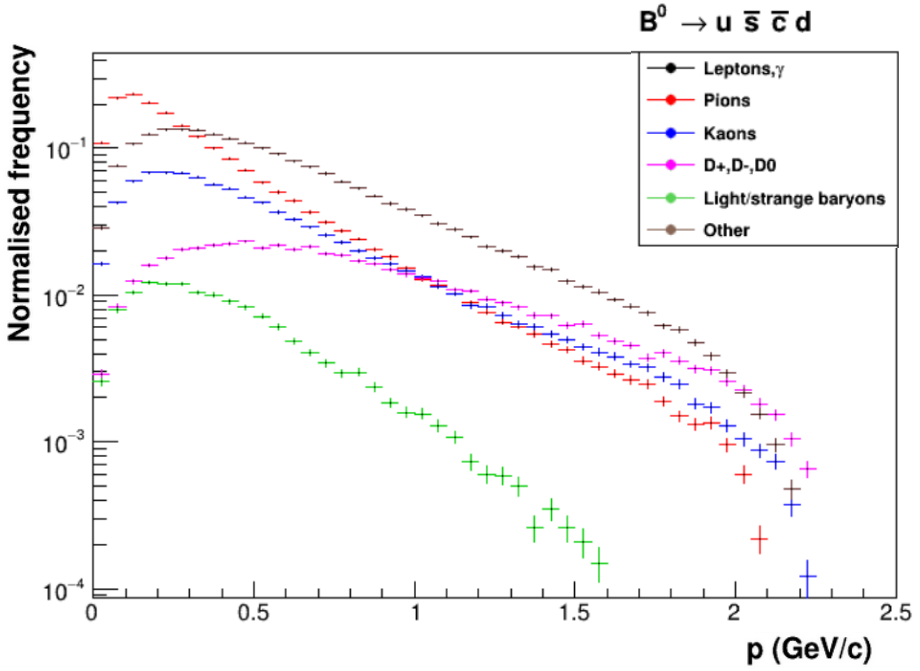
Simulation needs testing and validation after changes to ensure invariance of the physics models

- Implemented **testing framework** with common testing module and JSON configuration files
- Migrated all previous tests (covered only 40% of models) and added new ones to framework
- Tests cover all models and **external generators**
- Used for **continuous integration** in GitLab
- Implemented **multi-threaded testing**

  ⇒ With possibility to use std or tbb thread models

Available also for users to facilitate testing/development





Refactoring the testing framework

- Starting point: monolithic module that reads in JSONs

- Goals: improve maintainability

  increase friendliness for users

- Current work on splitting into
  - Test configuration classes
  - Variable utils (helper functions)
  - Variable factory (registry for variables)

  ⇒ Reduced code duplications
  - Testing module (orchestrator running the test)

⇒ Improving Doxygen, working on docu

Example JSON

```
{
   "parent" : "D+",
   "daughters" : ["K-", "pi+", "pi+"],
   "models" : ["D_DALITZ"],
   "parameters" : [[]],
   "do_conjugate_decay" : [true],
   "extras" : ["yesFSR"],
   "events" : 10000,
   "histograms" : [
       {"variable" : "prob", "title" : "Probability", "d1" : 0, "d2" : 0, "nbins" : 100, "xmin" : 0.0, "xmax" : 1.0},
       {"variable" : "nFSRPhotons", "title" : "nFSRPhotons", "d1" : 0, "d2" : 0, "nbins" : 10, "xmin" : 0, "xmax" : 10.0 },
       {"variable" : "totalFSREnergy", "title" : "totalFSREnergy", "d1" : 0, "d2" : 0, "nbins" : 100, "xmin" : 0, "xmax" : 0.
       {"variable" : "E_FSRPhotons", "title" : "FSR E(#gamma)", "d1" : 0, "d2" : 0, "nbins" : 100, "xmin" : 0.0, "xmax" : 0.8
       {"variable" : "E_FSRPhotons", "title" : "FSR E(#gamma) cut off", "d1" : 1, "d2" : 0, "nbins" : 50, "xmin" : 0.0, "xmax
       {"variable" : "cosTheta_FSRPhotons", "title" : "FSR cosTheta/#gamma)", "d1" : 0, "d2" : 0, "nbins" : 100, "xmin" : -1.
       {"variable" : "openingAngle_FSRPhotons", "title" : "FSR opening angle", "d1" : 0, "d2" : 0, "nbins" : 90, "xmin" : 0,
       {"variable" : "cosTheta_EnergyWeight_FSRPhotons", "title" : "FSR cosTheta(#gamma)", "d1" : 0, "d2" : 0, "nbins" : 100,
       {"variable" : "openingAngle_EnergyWeight_FSRPhotons", "title" : "FSR opening angle", "d1" : 0, "d2" : 0, "nbins" : 90,
       {"variable" : "mass", "title" : "m(K^{-} #pi^{+})", "d1" : 1, "d2" : 2, "nbins" : 100, "xmin" : 0.5, "xmax" : 1.8},
       {"variable" : "mass", "title" : "m(K^{-} #pi^{+})", "d1" : 1, "d2" : 3, "nbins" : 100, "xmin" : 0.5, "xmax" : 1.8},
       {"variable" : "mass", "title" : "m(#pi^{+} #pi^{+})", "d1" : 2, "d2" : 3, "nbins" : 100, "xmin" : 0.2, "xmax" : 1.5},
       {"variable" : "massSq", "title" : "Dalitz m^{2}(K^{-} #pi^{+}) vs m^{2}(K^{-} #pi^{+})", "d1" : 1, "d2" : 2, "nbins" :
       {"variable" : "massSq", "title" : "Dalitz m^{2}(#pi^{+} #pi^{+}) vs m^{2}(K^{-} #pi^{+})", "d1" : 2, "d2" : 3, "nbins"
       {"variable" : "mPrime", "title" : "sqDalitz", "d1" : 1, "d2" : 2, "nbins" : 25, "xmin" : 0.0, "xmax" : 1.00, "variable
       {"variable" : "cosHel", "title" : "cosHel12", "d1" : 1, "d2" : 2, "nbins" : 50, "xmin" : -1.0, "xmax" : 1.0},
       {"variable" : "cosHel", "title" : "cosHel23", "d1" : 2, "d2" : 3, "nbins" : 50, "xmin" : -1.0, "xmax" : 1.0},
       {"variable" : "cosHel", "title" : "cosHel13", "d1" : 1, "d2" : 3, "nbins" : 50, "xmin" : -1.0, "xmax" : 1.0}
   ],
   "outfile" : "D_DALITZ__D+_Kpipi+_yesFSR.root"
}
```

```
/**
 * @class VariableFactory
 * @brief Decouples variable string names from physics calculation logic.
 * * This factory maintains a registry of string-to-calculator mappings.
 * Calculations are performed via the public getValue() function.
 * * @note **Automatic Absolute Values**: Prefixing any name with "abs" and
 *   capitalizing the first letter (e.g., "cosTheta" -> "absCosTheta")
 *   automatically returns the absolute value.
 * * @note **Indices (d1, d2)**: References to @p d1 and @p d2 below refer to
 *   the 1-based indices of daughters as defined in the provided JSON configuration.
 * * @section var_list Supported Variables
```

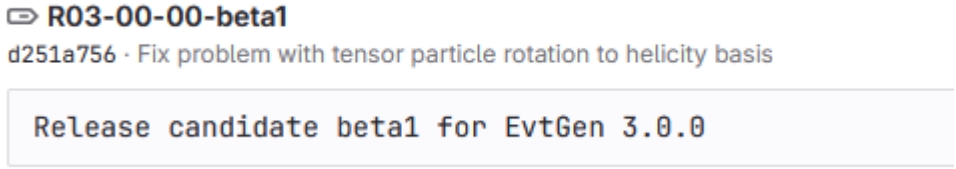
EvtGen releases

Release with latest patch release R02-02-03 (Sep 2024)

- Fixed RNG interface issue with Pythia8 310
- Fixed bug with tensor particle rotation to helicity basis

After the long campaign released R03-00-00-beta1 (Oct 2024) for users to test

- Implemented **thread safety**
- Implemented new **testing framework**
- Added Sherpa's PHOTONS++ as **FSR alternative**
- **Fixed** various **decay models** (removed obsolete ones)



Physics validation passed for all models with/without FSR when multithreading.

Passed LHCb Sim11 tests

Check it out!

Plans for the future

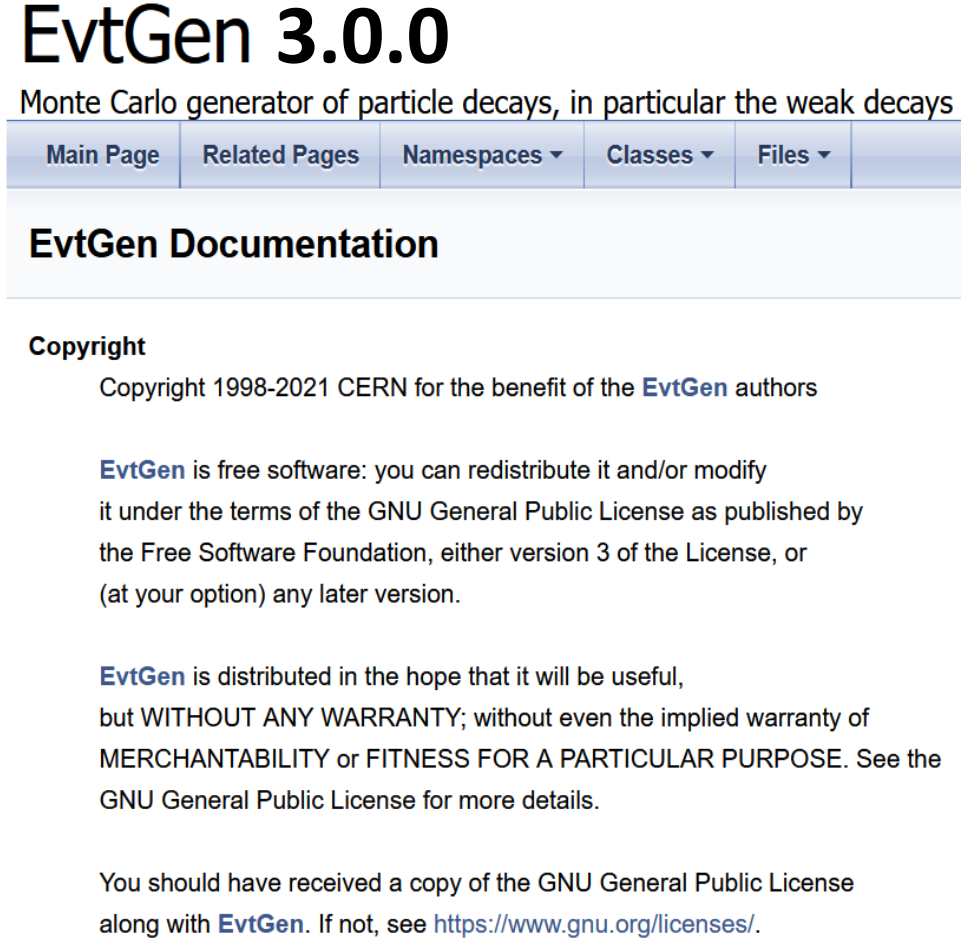
https://evtgen.hepforge.org/doc/doxygen/test

Immediate

- Improve Doxygen documentation
- Release version 3
- Prepare main Journal article

Long-term

- Add further models as requested by community
- Implement structural changes to fully exploit multi-threadir
- Implement alternatives for τ particle simulation (fix spin propagation using PYTHIA8)
- Interleave Vincia FSR with entire decay chain
- Refine event weights to support alternative model configs



EvtGen **3.0.0**

Monte Carlo generator of particle decays, in particular the weak decays

Main Page | Related Pages | Namespaces | Classes | Files

EvtGen Documentation

Copyright

Copyright 1998-2021 CERN for the benefit of the **EvtGen** authors

EvtGen is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

EvtGen is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with **EvtGen**. If not, see https://www.gnu.org/licenses/.

Summary and outlook

- EvtGen — essential tool to study heavy-flavour particle decays in multiple experiments
- After long modernisation campaign ⇒ R03-00-00-beta1

⇒ New general **testing framework**

⇒ Implemented **thread safety**

(full exploitation of multi-threading will require further structural changes)

⇒ Performance limited by external dependencies

⇒ Implemented Sherpa's PHOTONS++ and Vincia as **alternatives for FSR**

- Various projects for future developments (τ-spin propagation, structural modifications, …)

⇒ Started decay weight implementation

- Full release 3.0.0 expected by summer 2025 (and paper by year's end)

Stay tuned!

https://evtgen.hepforge.org

evtgen@projects.hepforge.org