# Interfacing to specialized tools

Fernando Abudinén, John Back, Michal Kreps, Thomas Latham

MC support tools workshop
IPPP Durham
April 03, 2025

# Outline

- Introduction
- Discussion from EvtGen’s perspective on
  - Decay weighting to study effect of alternative configurations
  - Plugins for final-state radiation
  - Propagating spin-information

# Introduction

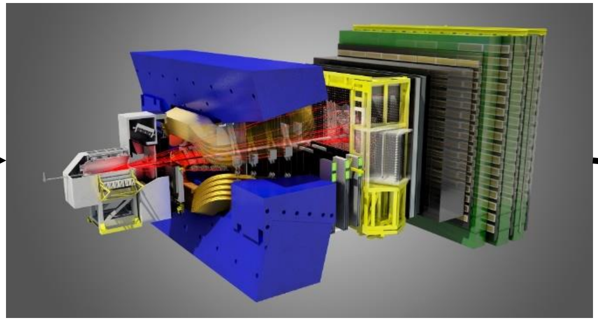
# Simulation in high-energy physics

Essential since we interpret measurements by comparing simulation with collision data
$\Rightarrow$ Ideally, simulation should mirror data differing only by the knowledge of the “truth”

**The world**

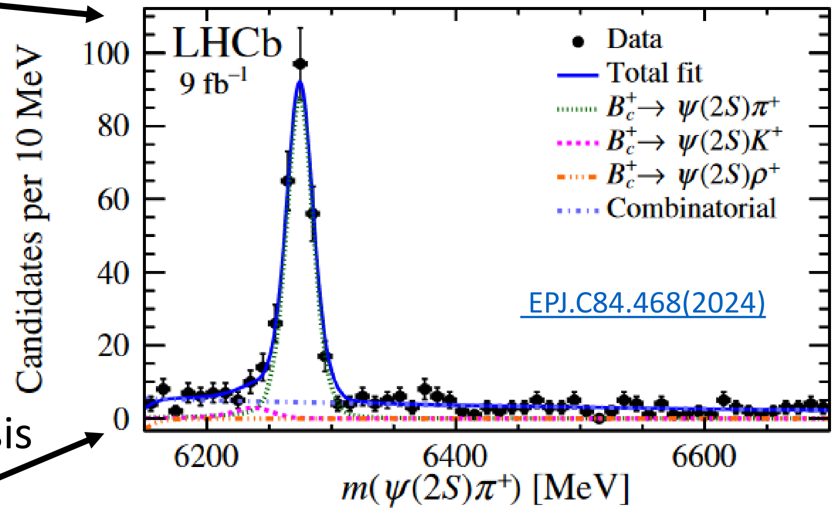
Collisions

**Experiment**

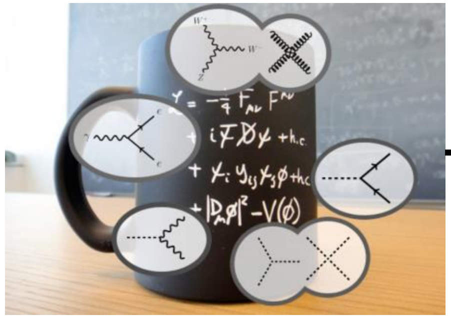


Analysis

**Measurement**

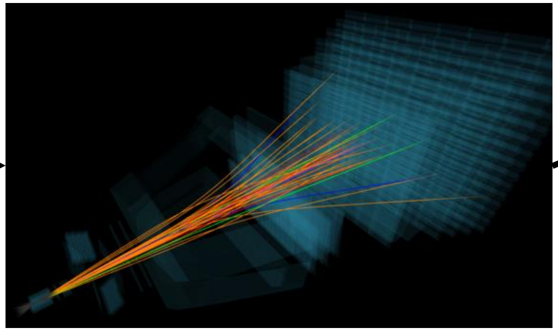



**Theory**



Event generators

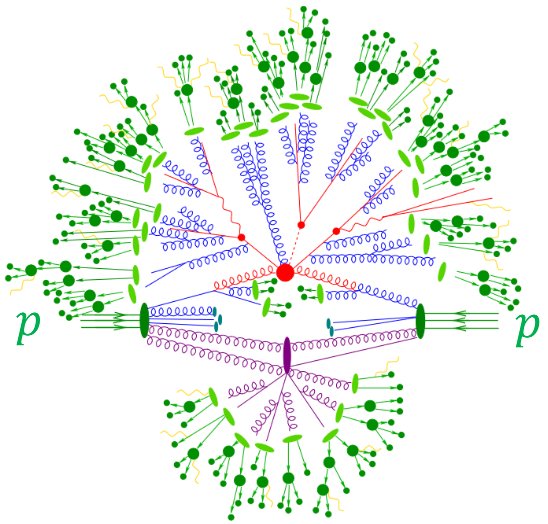
**Simulation**



Analysis

# The EvtGen generator

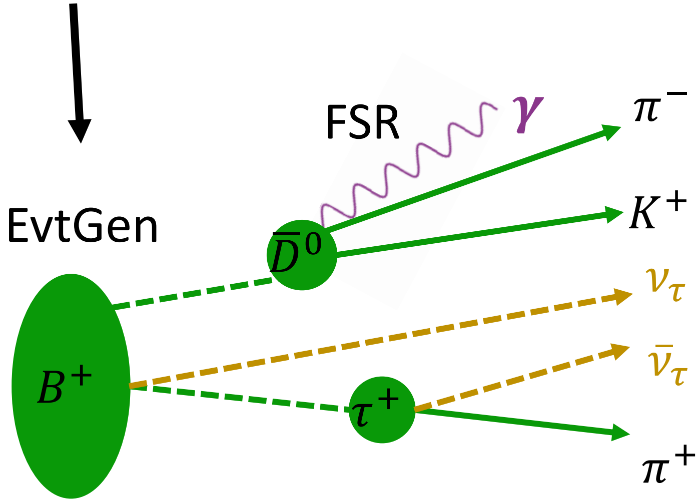
Example collision simulated by PYTHIA8




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

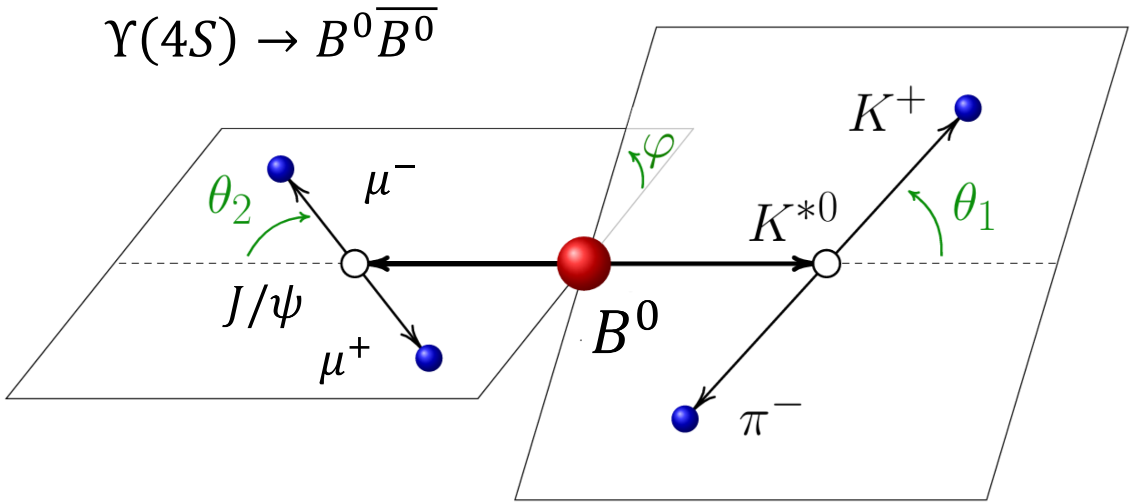
- Implements detailed decay dynamics based on theoretical models
- Originally developed for BaBar and CLEO by Anders Ryd and David Lange
- Used in multiple high-energy physics experiments:

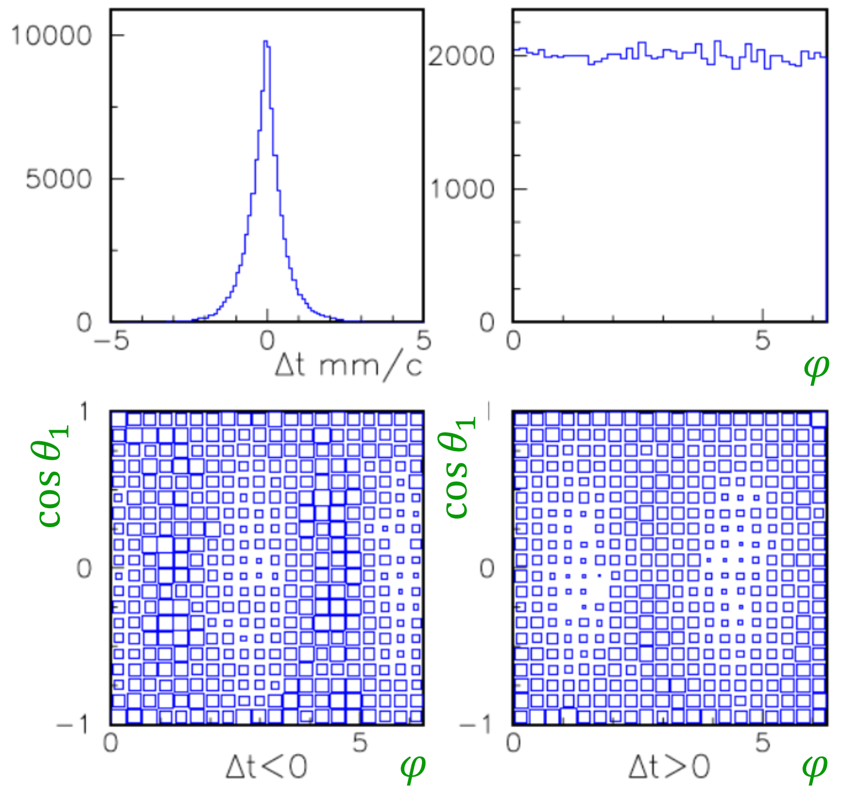
  ATLAS, Belle II, BES III, CMS, LHCb, …



# Physics motivation

- Designed to handle complicated decay chains
- Account for dependencies between different observables

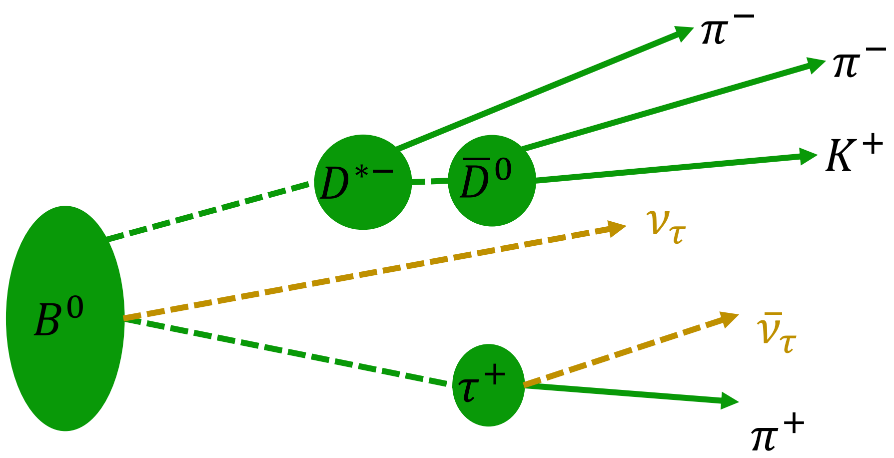



Example decay with $CP$ violation and dependencies between decay time and angular observables.

# Decay Chains

Decays of heavy-flavour particles often involve many sequential decays




**Concept for a reusable generic tool**

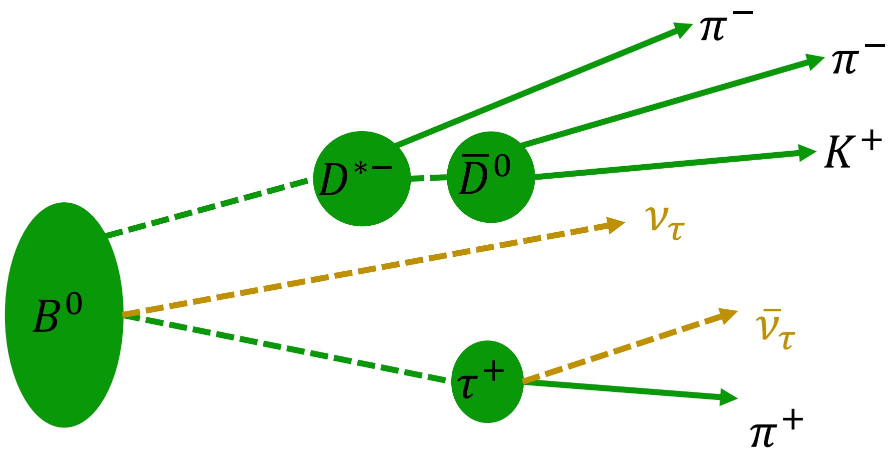
- Simulate correctly the **full decay chain** implementing only **individual nodes**

$\Rightarrow$ Use **decay amplitudes** instead of probabilities

$\Rightarrow$ Make use of **modular design** in C++ implementation

# Decay amplitudes

- Use decay amplitudes to simulate **sequences** of decays
- Each node in the chain generated separately
  $\Rightarrow$ Must be associated with a **decay model**
- Decay models provide the amplitudes
- Framework handles bookkeeping to generate full decay chain




**Workflow**

Generate kinematics of $B^0$ according to phase space → Perform accept/reject based on $P_B = \sum_{\lambda_{D^*}\lambda_\tau} \left| A^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau} \right|^2$ → Propagate the spin-state information to subsequent decays through spin-density matrix

# Decay models

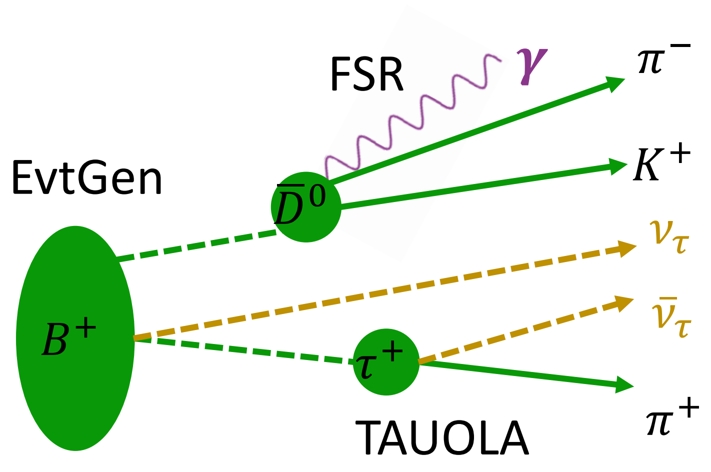
EvtGen contains about **130 decay models**

- **General purpose** models
  - Based on particle spin properties
  - Or specified helicity/partial wave amplitudes
- **Semileptonic** models with **form-factors**
- **Dalitz plot** decays (generic and specific Dalitz models)
- **Specific models** for electroweak penguins / radiative decays
  - For example $b \to s\ell\ell$, $b \to s\gamma$
- Many models have versions including $\boldsymbol{CP}$ **violation**

# External dependencies

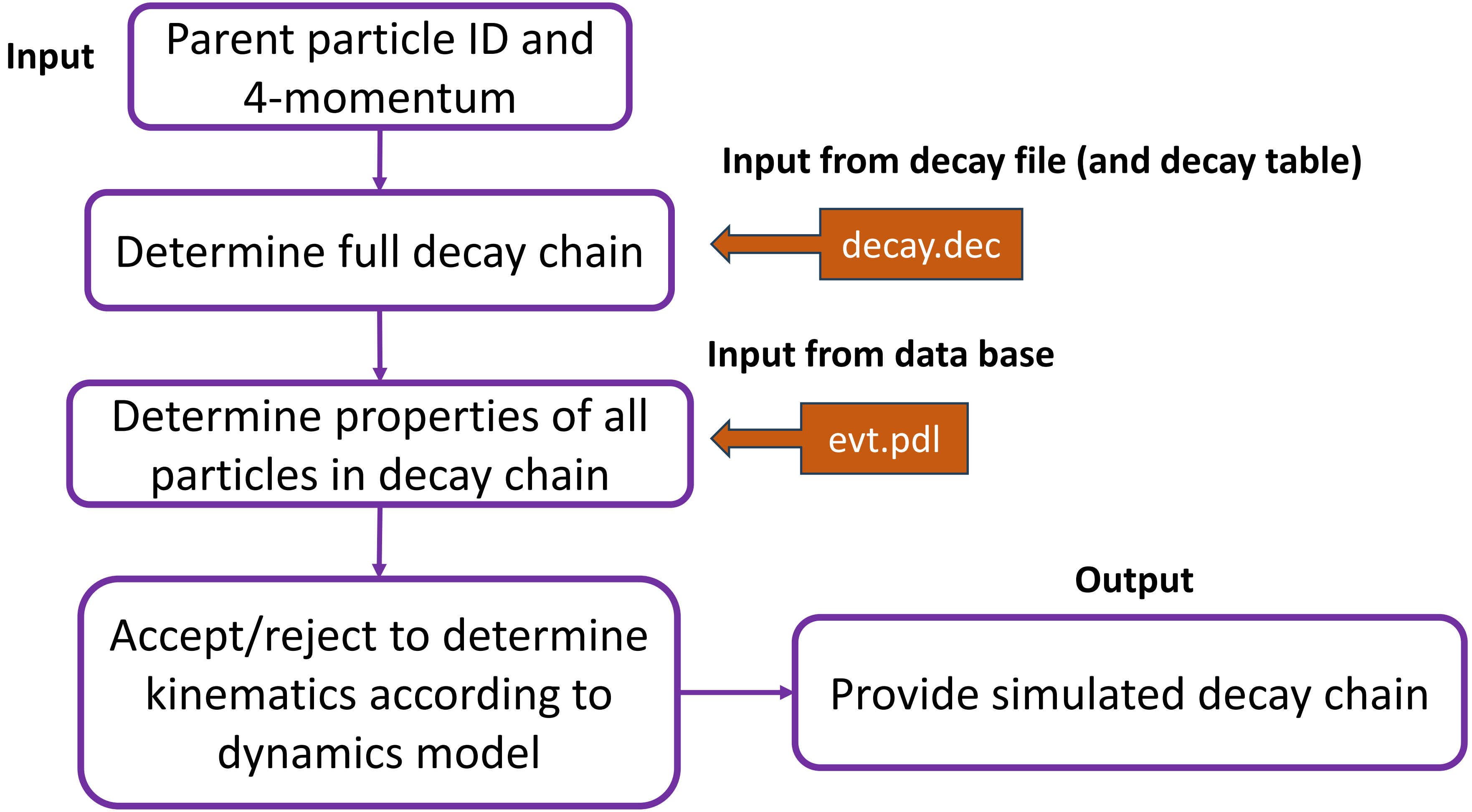
Interface with external packages for **additional features**

- HepMC for writing events in HepMC format (mandatory)
- Pythia8 for decays of generic quark configurations (optional)
- TAUOLA for decays of $\tau$ particles (optional)
- PHOTOS for final-state photon radiation (FSR) (optional)
- PHOTONS++ for final-state photon radiation (FSR) (optional)




Inconsistencies can result from mixing other (typically Pythia) hadron models into another generator's decay chains (and unresponsive to tunes)

# EvtGen decay algorithm



# Decay files and decay table

**Decay files** determine the decay chain

- Specify **decay modes** and their **branching fractions**
- Specify **decay models** and their **input parameters**
- Can be provided as text (.dec) or XML file

```
Decay B0sig
  0.90 J/psi K*0    SVV_HELAMP Hp pHp Hz pHz Hm pHm;
  0.10 Jpsi  K+ pi- PHSP;
Enddecay
```

EvtGen maintains a **generic decay table** (DECAY.dec) with properties of $\sim 10^4$ explicit decays

- Updated from PDG at intervals (nontrivial effort)
- When known branching fractions do not add up to 100%

  $\Rightarrow$ Fill up remainder with generic quark configurations and pass to Pythia8

  $\Rightarrow$ $b$-baryons rely more on Pythia8 than other particles

# Decay table and custom IDs

- General Decay table does not consider uncertainties in branching fractions
- Each user can have custom general Decay table
- Currently no possibility to provide uncertainty or weighting variation for decays
- **Challenges:** Inconsistencies, PDG BFs not adding up to 1, theory uncertainties (form factors, etc)
- **Particle ID standards:** some particle IDs used in EvtGen are outside the standard range
- First step would be to support alternative BFs

**Example from general Decay table**

```
Decay B0
#
# b -> s gamma
#
0.0003118   Xsd    gamma     BTOXSGAMMA 2 ;
#
Decay Xsd
1.00000 d anti-s PYTHIA 42;
Enddecay
```

# Interface between EvtGen and Pythia

- EvtGen calls Pythia to decay and hadronise quark configurations in some cases
- Nothing to be put into the event record before Pythia simulation
- Direct translation of EvtGen objects into Pythia event (and back)
- Needs interplay between generators (making sure to avoid double counting)
- Needs matching of particle properties and decay table.

**Example from general Decay table**

```
Decay Omega_b-
#                   SemiLeptonic Decays
  0.05460    Omega_c0     e-     anti-nu_e                  PHSP;
  0.05460    Omega_c0     mu-    anti-nu_mu                 PHSP;
  0.02000    Omega_c0     tau-   anti-nu_tau                PHSP;
#                   Hadronic Decays with Xi_c+
  0.00600    Omega_c0     pi-                               PHSP;
  0.02200    Omega_c0     pi-     pi+       pi-             PHSP;
  0.00055    Omega_c0     K-                                PHSP;
  0.02200    Omega_c0     D_s-                              PHSP;
  0.0011     D0           Xi-                               PHSP;
#
  0.00047    Omega-       J/psi                             PHSP;
  0.00038    Omega-       psi(2S)                           PHSP;
#                   filling out with inclusives
0.68192 d anti-u s cs_0 PYTHIA 43;
0.10910 d anti-u d cs_0 PYTHIA 43;
0.02728 d anti-u s su_0 PYTHIA 43;
Enddecay
```

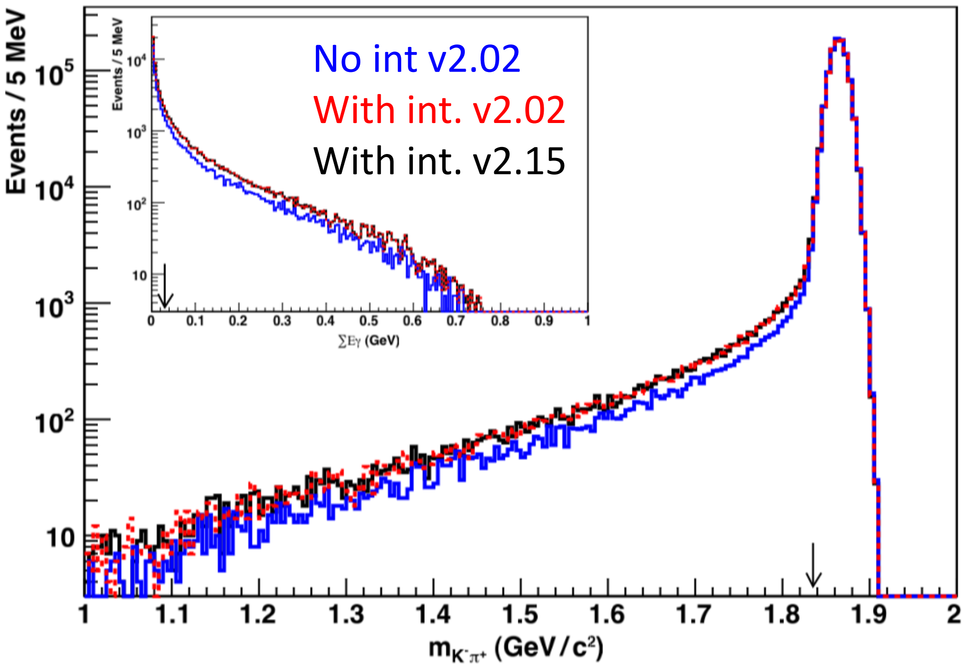
# Studies of final-state radiation

# Alternatives for final-state radiation

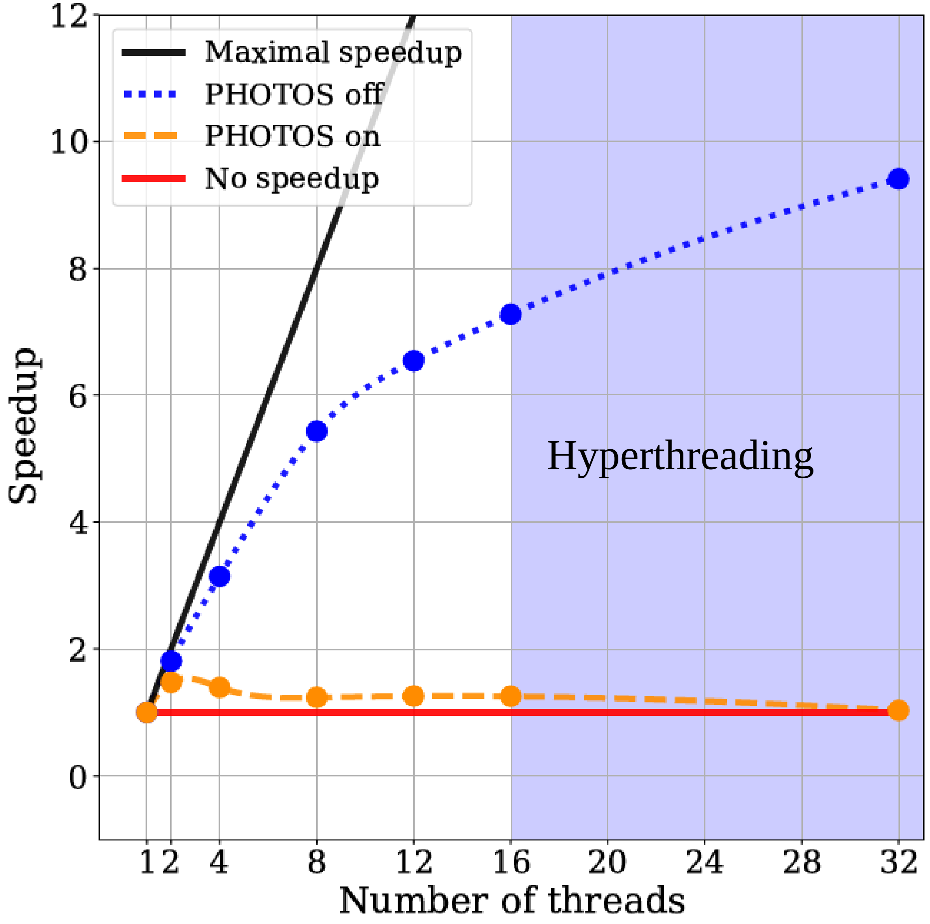
- FSR is main limitation to exploit multi-threading
- Find alternatives to study systematic effects

$\Rightarrow$ Especially those associated with interference effects

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




**EvtGen simulation**

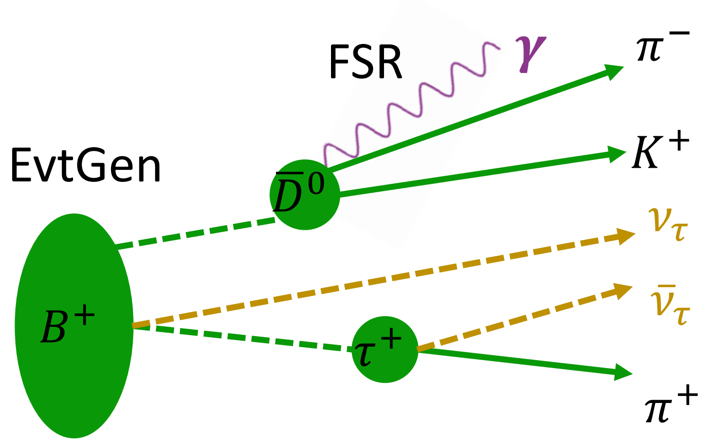



See HFLAV Sec 11.3

| | | stat. | syst. | FSR |
|---|---|---|---|---|
| $\mathcal{B}(D^0 \to K^-\pi^+)$ | $= (3.999$ | $\pm 0.006$ | $\pm 0.031$ | $\pm 0.032)\,\%,$ |
| $\mathcal{B}(D^0 \to \pi^+\pi^-)$ | $= (0.1490$ | $\pm 0.0012$ | $\pm 0.0015$ | $\pm 0.0019)\,\%,$ |
| $\mathcal{B}(D^0 \to K^+K^-)$ | $= (0.4113$ | $\pm 0.0017$ | $\pm 0.0041$ | $\pm 0.0025)\,\%\,.$ |

# Final-state radiation in EvtGen

- EvtGen relies on external specialised generators to add QED FSR corrections
- Generators generally treat the effect of FSR as a multiplicative correction to the decay rate




$$d\Gamma^{\text{radiative}} = d\Gamma^{\text{Born}}\, f(\Phi)\, d\Phi$$

$\Phi$: Phase-space of photons

```
Decay D0sig
  0.0390    K-   pi+    PHOTOS PHSP;
Enddecay
CDecay anti-D0sig
```

- Generators add photons (accept/reject) based on $f(\Phi)$
- Default generator is PHOTOS
- Recently included Sherpa's PHOTONS++ as alternative
- Currently developing Vincia (inside Pythia8) as alternative

PHOTOS flag deprecated with FSR flag in EvtGen r3.X.X

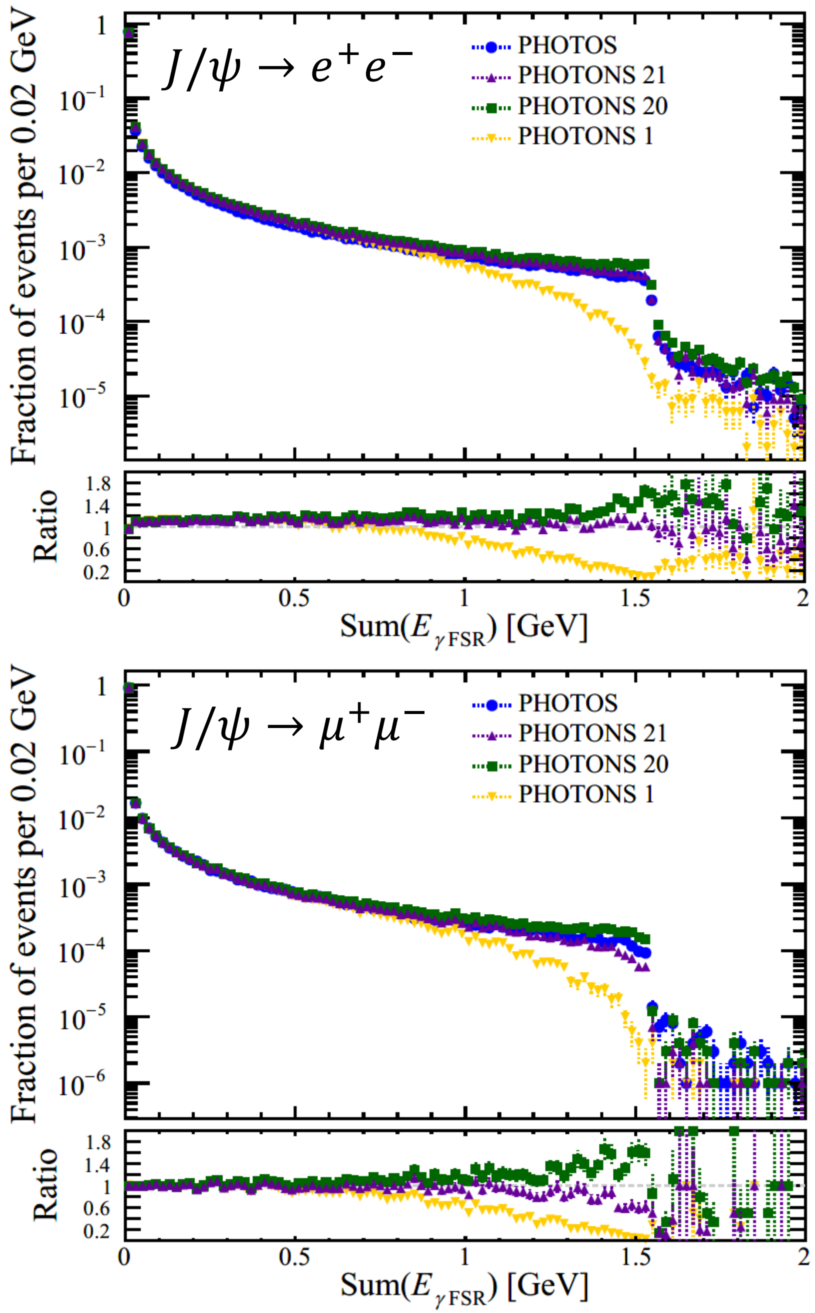
# Sherpa’s PHOTONS++ for FSR

- 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

$\Rightarrow$ Implemented switches for systematic studies

New in EvtGen R03-00-00-beta1!

**Amount of radiated energy**



# Vincia QED shower for FSR

With Giacomo Morgante and Peter Skands @ Monash

- Vincia parton shower evolution based on Antenna approximation (can be interleaved)
- Recently adapted to radiate off hadrons (previously supporting only leptons)
- Matrix-element corrections (MECs) in progress

$\Rightarrow$ Currently implementing and validating

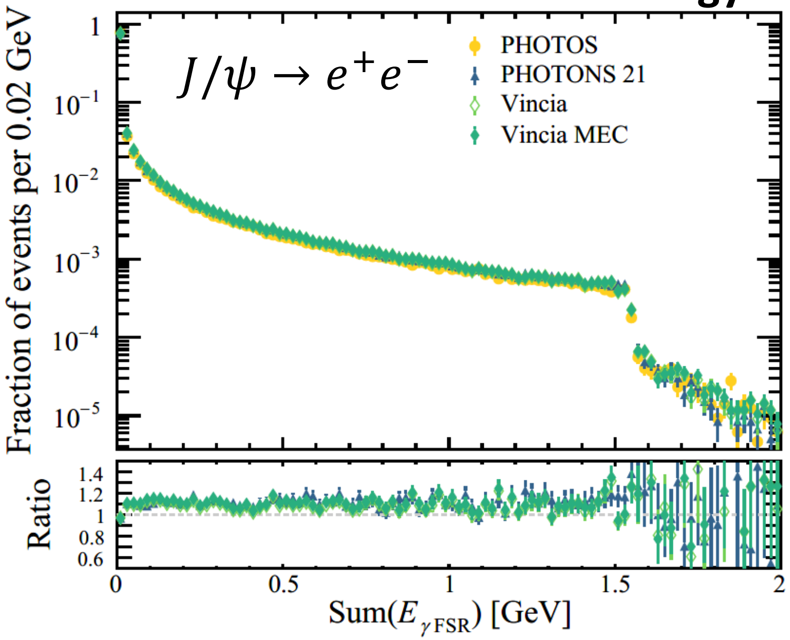
$\Rightarrow$ Preliminary results look promising

## Technical aspects

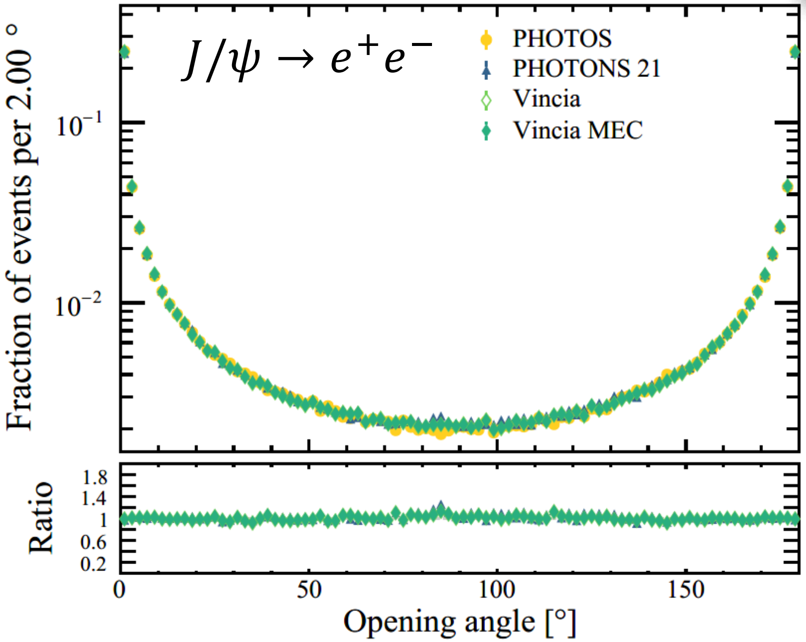
- Vincia is embedded in Pythia8
- Algorithm implementation enables thread safety
- Developed EvtGen $\leftrightarrow$ Vincia interface based on existing dependency with Pythia8

$\Rightarrow$ To be added to release (once in Pythia8 release)

**Amount of radiated energy**




**Angular distribution of photons**

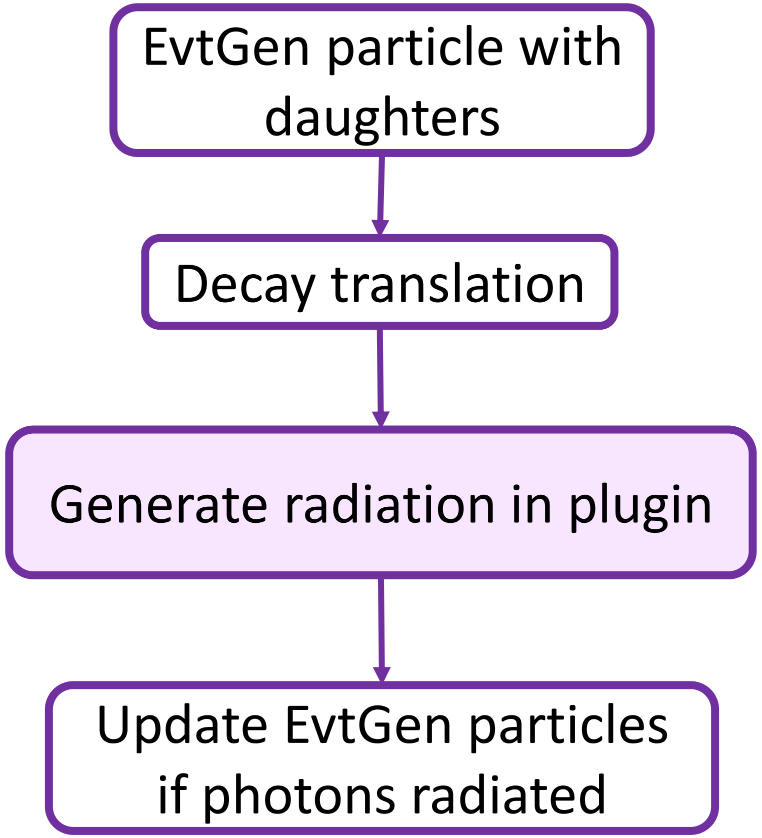


# Interfaces between EvtGen and Plugins

- 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

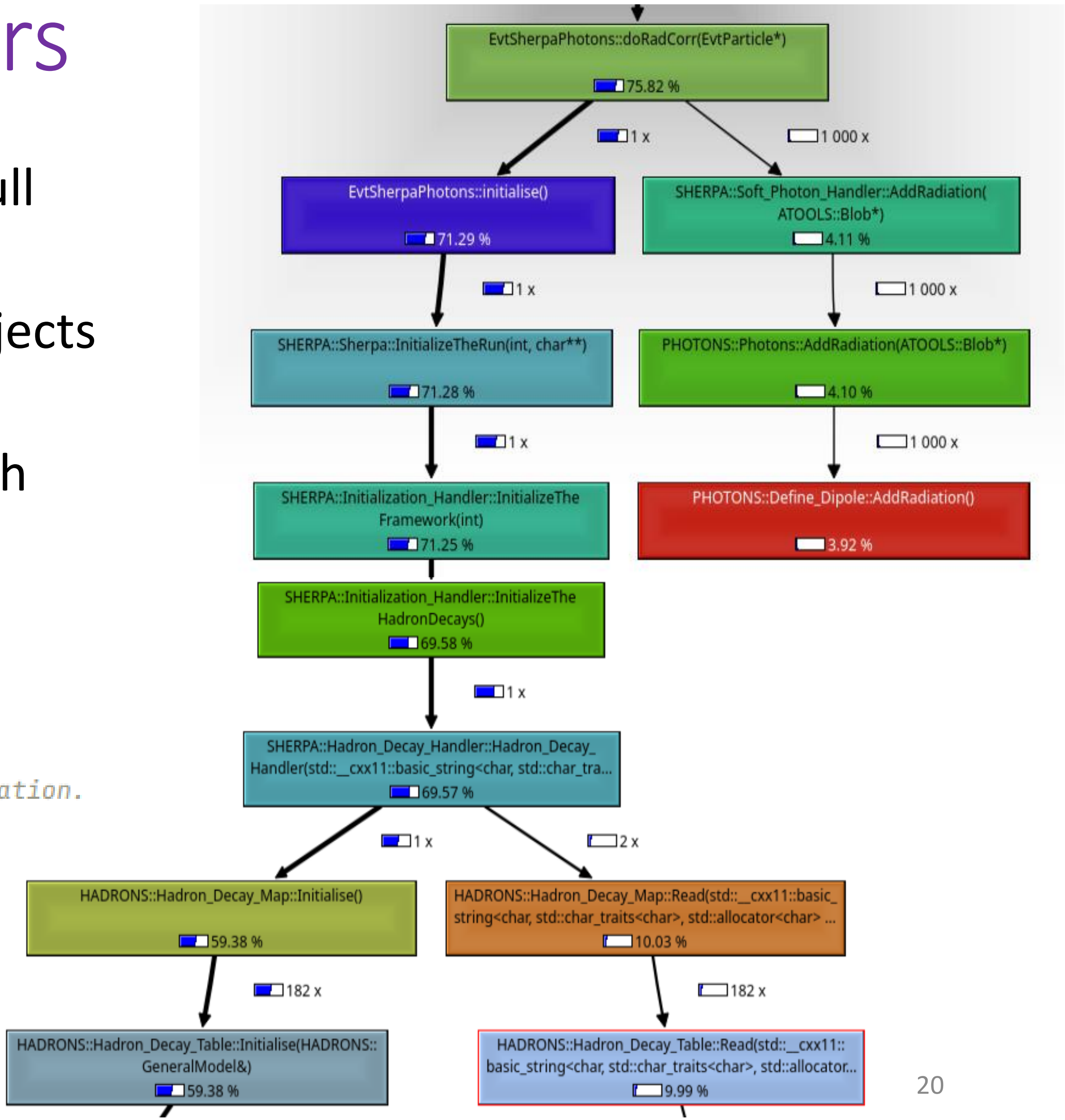
**Workflow**



# Initialisation of afterburners

- For external dependencies EvtGen creates full Pythia/Sherpa objects
- Would be useful to initialise only needed objects (shower/soft-photon handler)
- Example: Sherpa's initialisation takes as much time as $\sim 10^4$ decay events
- Several initialised objects are not used

```
// Vector containing the configuration strings for Sherpa
// INIT_ONLY=6 intialises the Sherpa objects without launching simulation.
std::vector<std::string> m_configs{ "Sherpa", "INIT_ONLY=6" };

// Create instance and initialise Sherpa.
m_sherpaGen = std::make_unique<SHERPA::Sherpa>();
m_sherpaGen->InitializeTheRun( argv.size(), &argv[0] );
m_sherpaGen->InitializeTheEventHandler();
```



# Static destructor fiasco

When generators have static instances the destruction order is undefined (C++ issue)

**Our current solution**

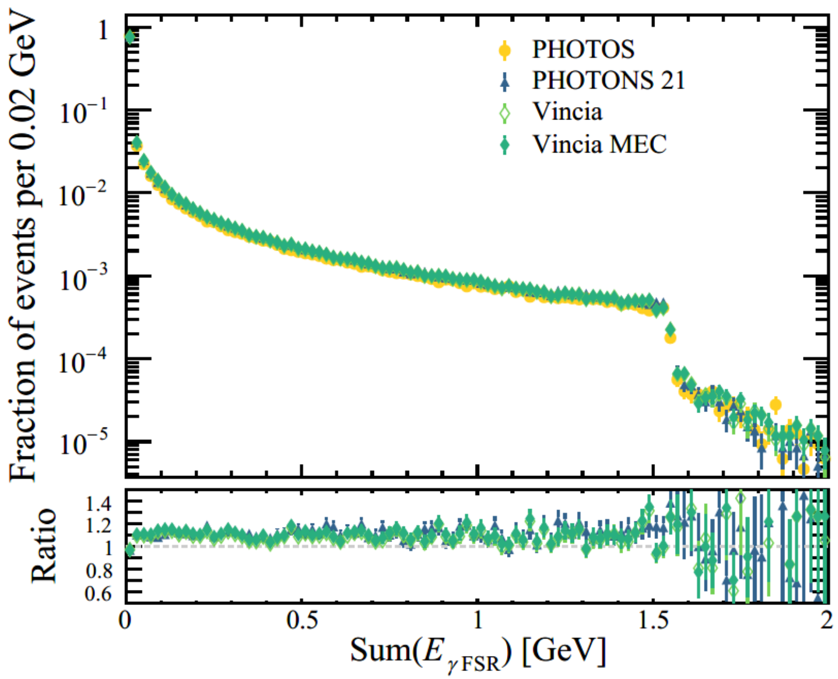
```
#ifdef EVTGEN_SHERPA
    // The Sherpa and PHOTONS++ instances are static global instances inside EvtGen
    // as they are not thread safe (they are mutexed). From Sherpa 3.0.0 on,
    // the internal Sherpa settings are also a static global instance.
    // Because of the static destruction order fiasco,
    // we need to control the order in which the static instances are destroyed.
    // This is done by hand inside the function below. If the function below is not called,
    // a SegFault occurs at the end of the program execution when using Sherpa for FSR.
    EvtSherpaPhotons::finalise();
#endif
```

```
void EvtSherpaPhotons::finalise()
{
    // Mutex this since Sherpa instances are static
    const std::lock_guard<std::mutex> lock( m_sherpa_mutex );

    if ( !m_initialised ) {
        return;
    }

    // We explicitly destroy the static instances to make sure that this occurs in the right order.
    // This is needed due to the static destruction order fiasco.
    m_photonsGen.reset();
    m_sherpaGen.reset();
    m_initialised = false;
}
```

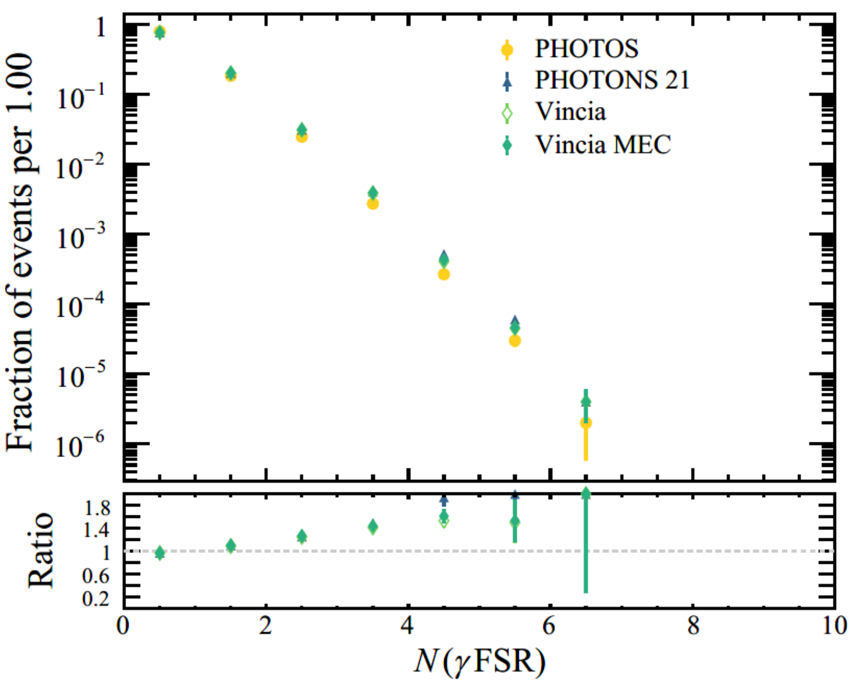
# Comparisons between generators

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

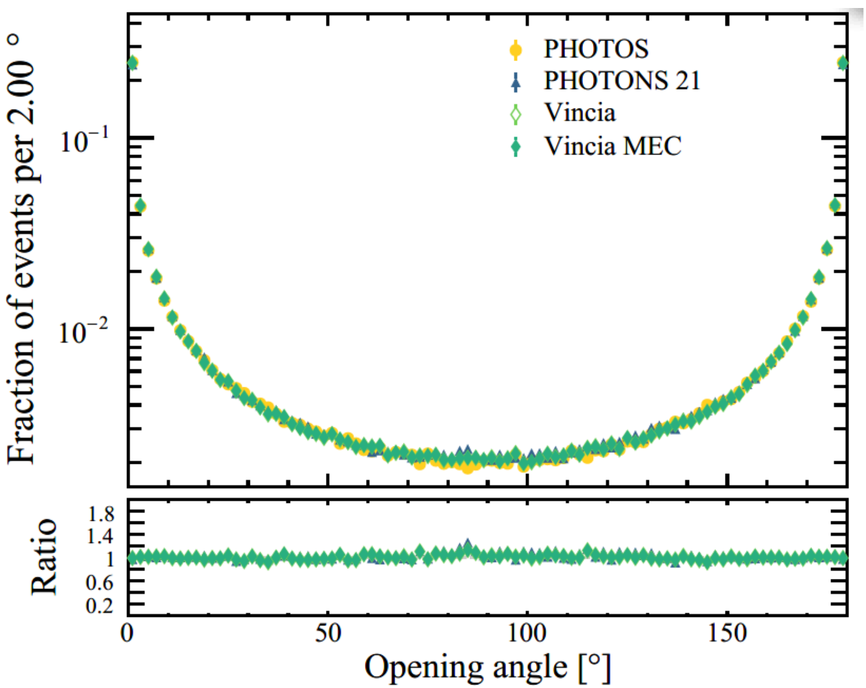
**Amount of radiated energy**




**Number of photons**




**Angular distribution of photons**

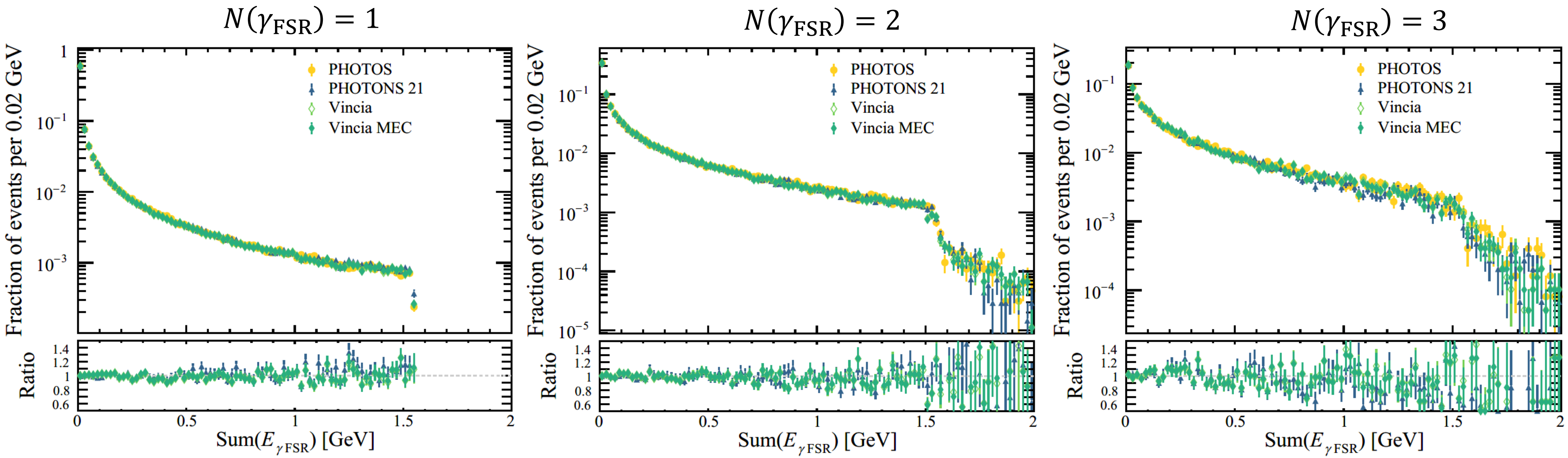



- Good agreement (within ~10%) for energy and angular distributions
- All generators radiate more photons that PHOTOS

# Comparisons between generators

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

**Amount of radiated energy**



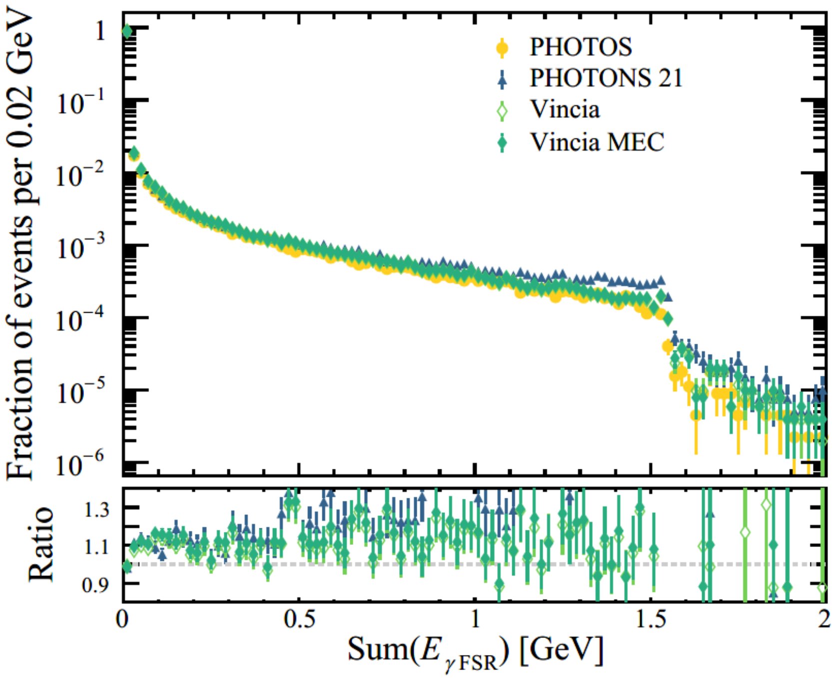

- Energy range above $M_{J\psi}/2$ kinematically accessible for events with more than one photon

# Comparisons between generators

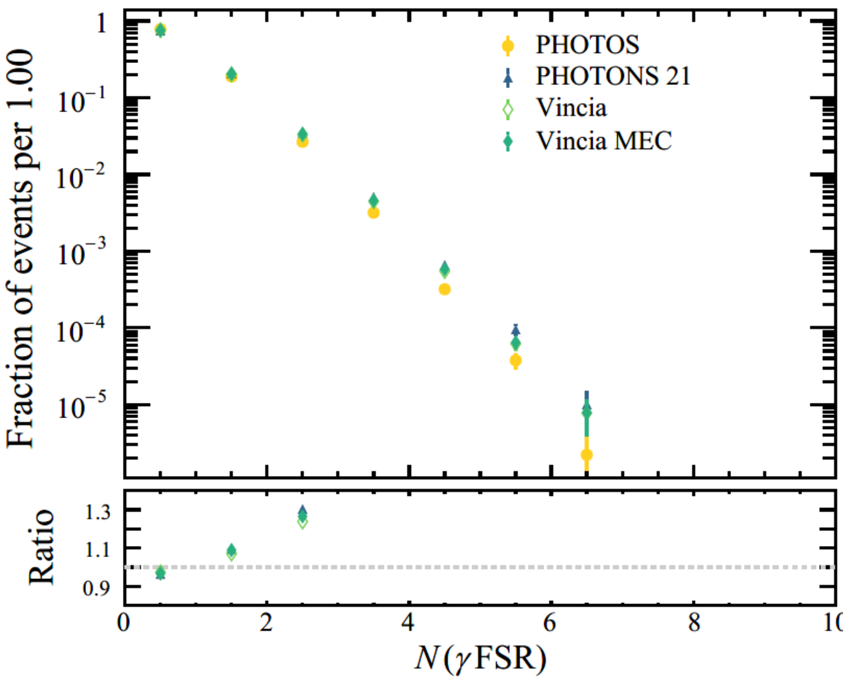
Reject photons with $\Delta R = \sqrt{\Delta\eta^2 + \Delta\varphi^2} < 0.1$ and $E_\gamma < 0.1$ MeV

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

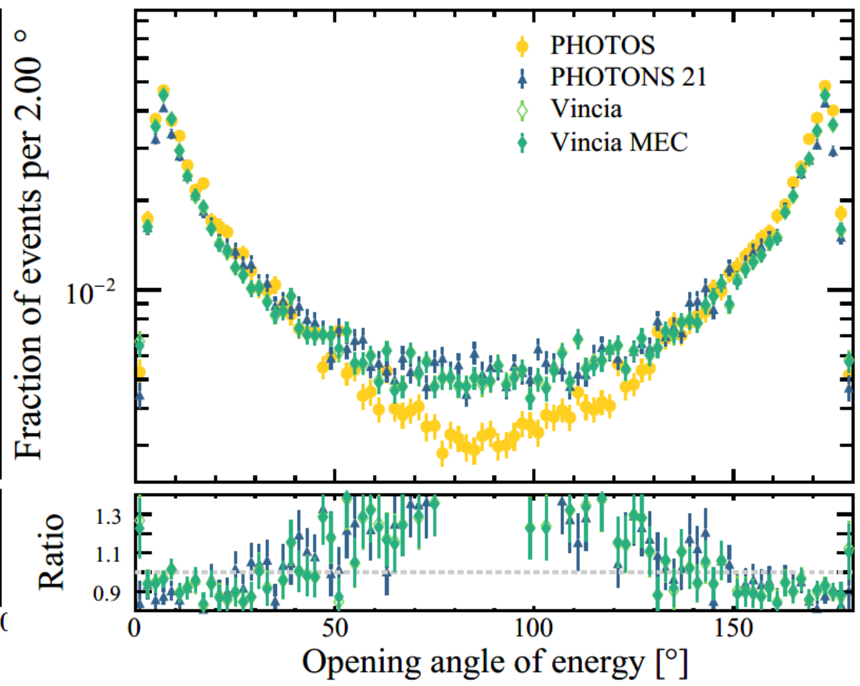
**Amount of radiated energy**




**Number of photons**




**Angular distribution of photons**

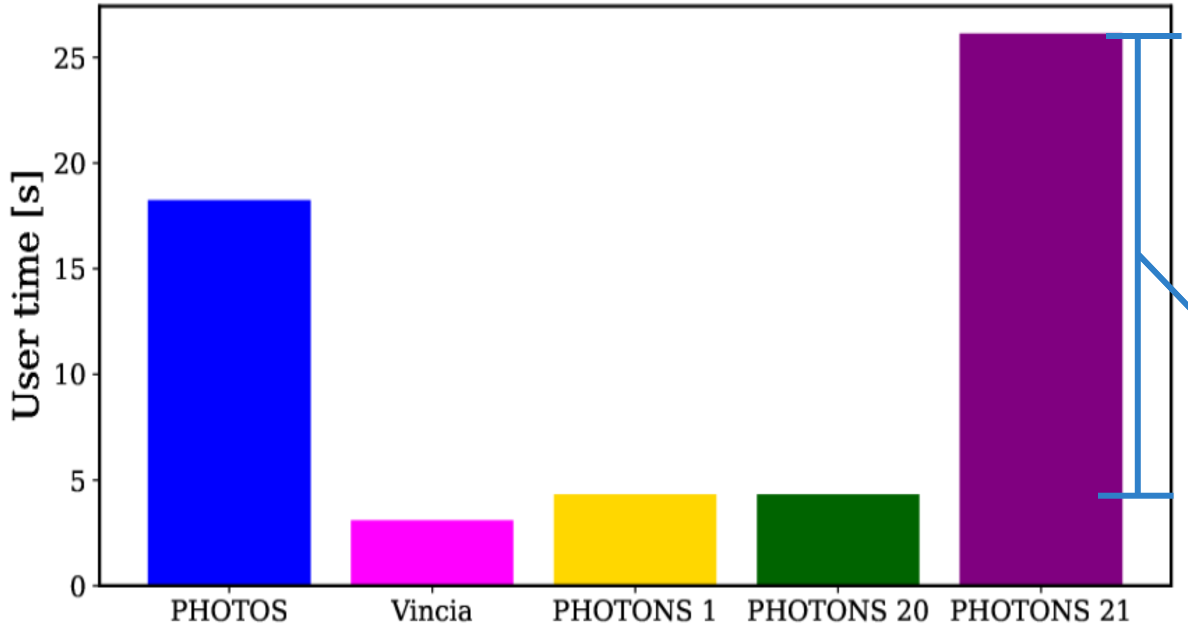


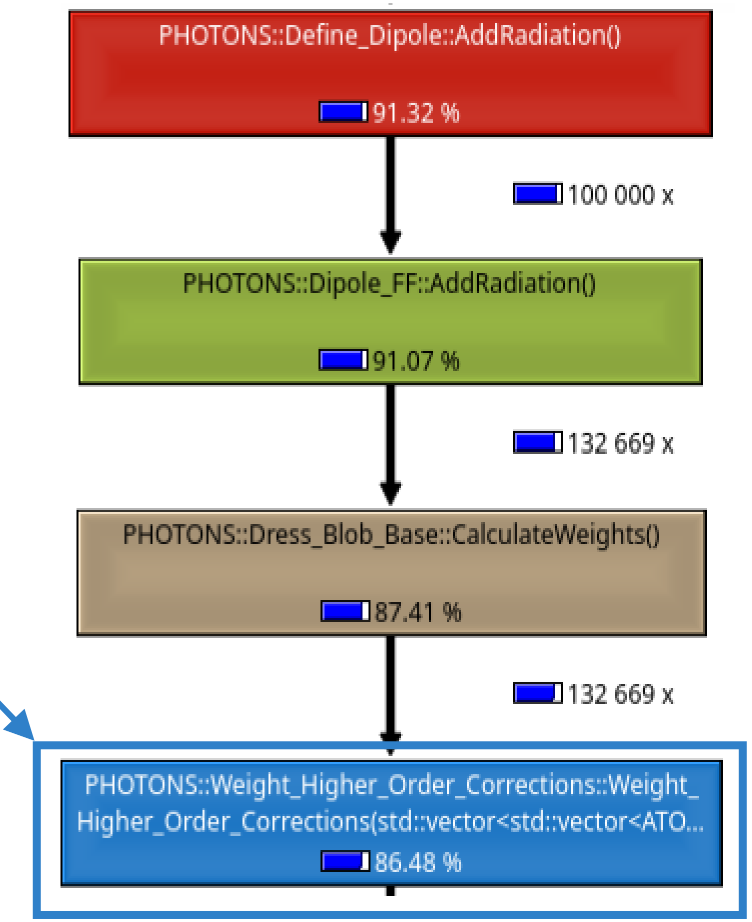

- Good agreement (within ~10%) for energy and angular distributions
- All generators radiate more photons that PHOTOS

# A word on timing

- Compare simulation time using $J/\psi \rightarrow e^+e^-$ decay as benchmark

$\Rightarrow$ Collinear singularities enhanced due to small electron mass




$\Rightarrow$ Largest consumption by exact matrix-element calculation

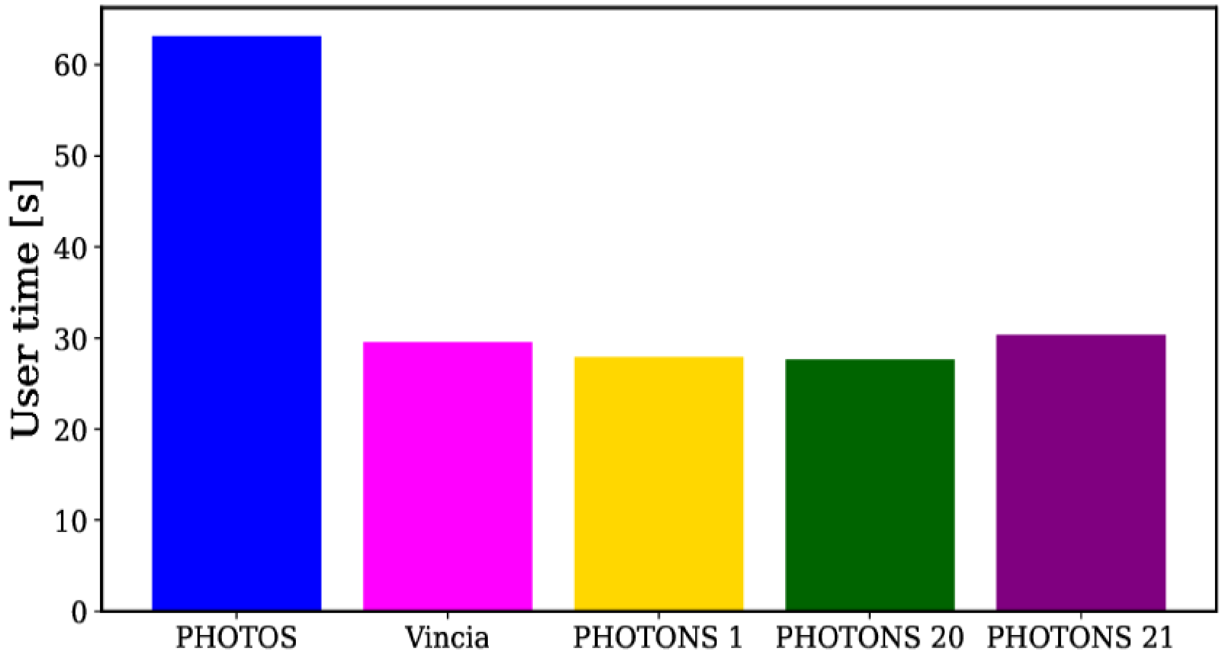
$\Rightarrow$ Good precision/time trade-off for option 20 (will use as default)

$\Rightarrow$ Potential speedup using Vincia or PHOTONS by about factor 4

# Another word on timing

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

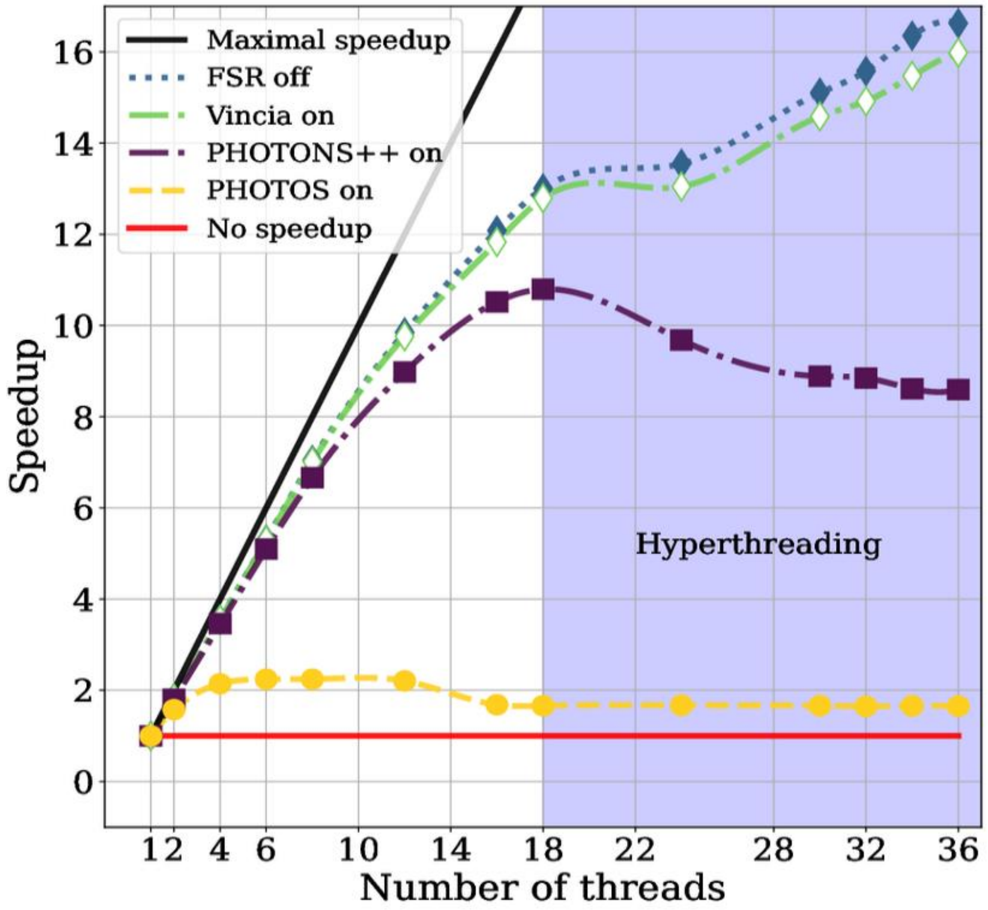
$\Rightarrow$ Benchmark for general use

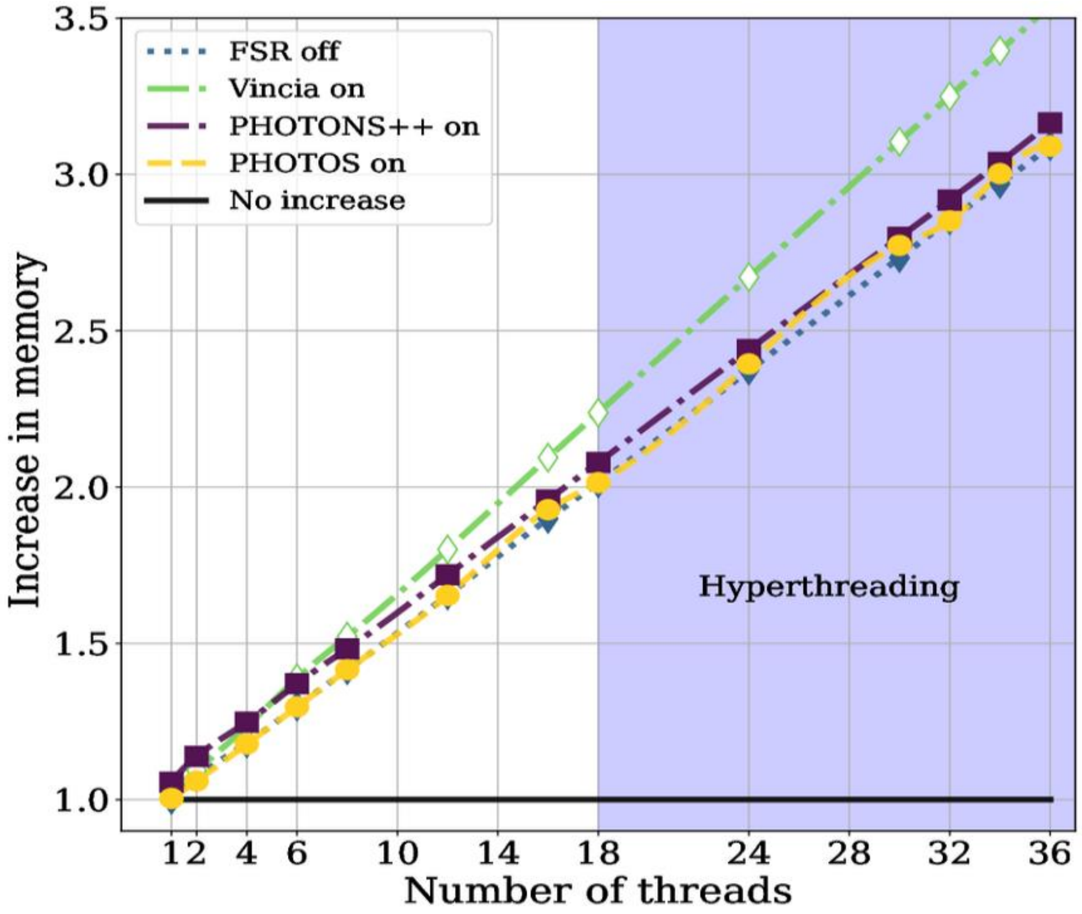



$\Rightarrow$ No large difference between PHOTONS options in generic case

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

# Performance with multithreading



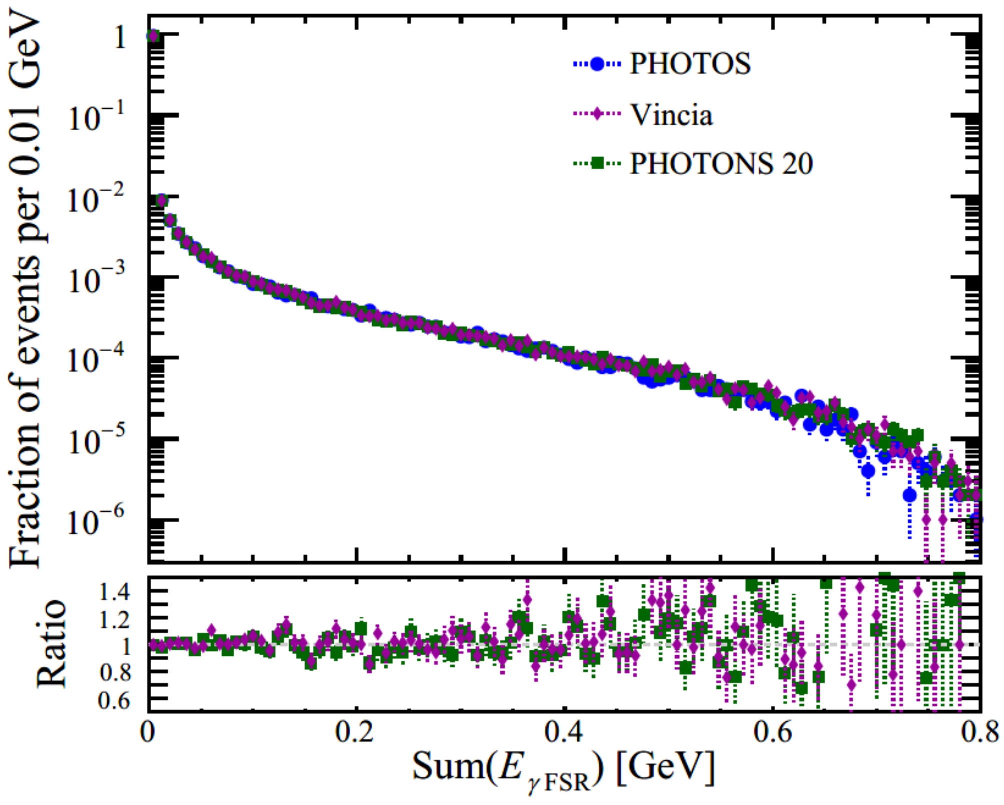

⇒ Better performance with new FSR alternatives

⇒ Deeper structural changes needed to fully exploit multithreading with increased memory sharing

# Ideas for future FSR generation

- FSR added by passing decay tree step-by-step (node-by-node) to the FSR generators
- Prototyped EvtGen ↔ Vincia interface using existing Pythia dependency (step-by-step)
- Aim to propagate full decay chain to Vincia to simulate radiation standalone
- Could simulate interference and resonance interleaving effects

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



# Spin-info propagation for $\tau$ decays

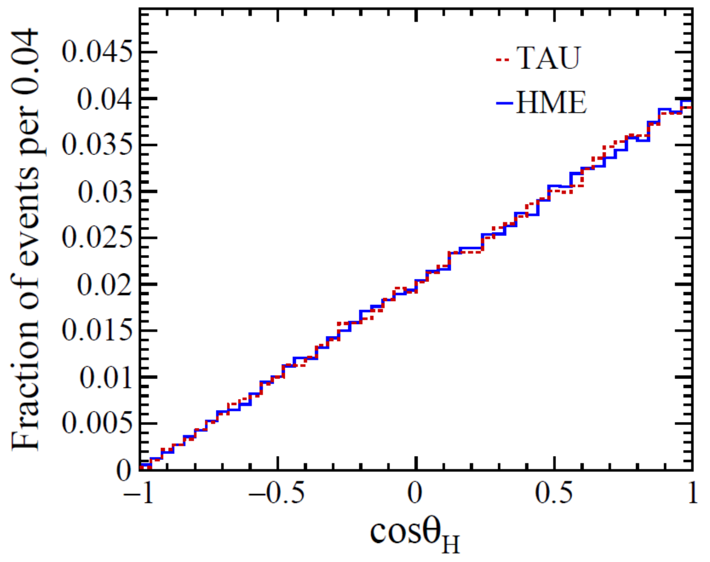
# Plugins for $\tau$ decays

- EvtGen $\leftrightarrow$ TAUOLA interface based on HEPMC
- Spin-state information of $\tau$ not propagated
  - TAUOLA reconstructs spin info from ancestors
  - Needed for analyses sensitive to $\tau$ polarization
- Simulation of $\tau$ decays with spin-state propagation possible with PYTHIA8 using HME (helicity-matrix element) model

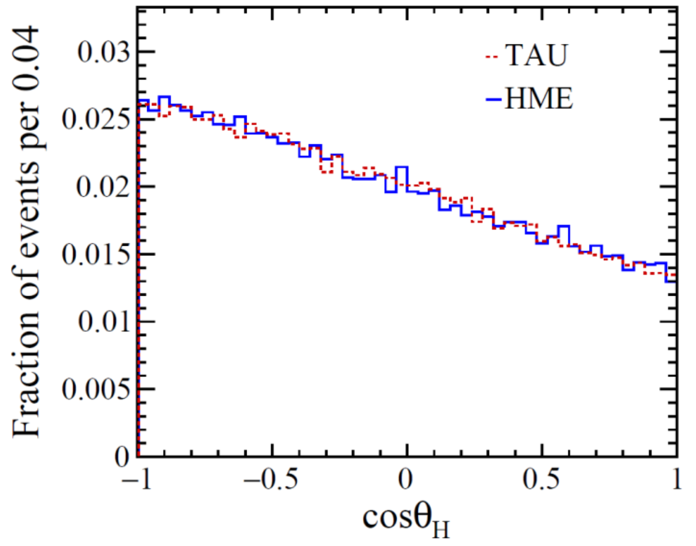
  $\Rightarrow$ Prototyped EvtGen $\leftrightarrow$ Pythia interface with spin-density matrix propagation

- Generalisation of helicity/spin basis conversion has turned out challenging (but wish to continue work)

$B^+ \to \tau^+(\to \pi^+\bar{\nu}_\tau)\,\nu_\tau$




$B^+ \to \tau^+(\to \mu^+\nu_\mu\bar{\nu}_\tau)\,\nu_\tau$



# Snippet from Herwig

Custom interfaces are efficient but need to align details about frame/conventions

[EvtGenInterface.cc]

```
else if ( thisSpin == EvtSpinType::DIRAC ) {
  EvtDiracParticle *myPart =new EvtDiracParticle;
  // get the spin info
  tcFermionSpinPtr sp(dynamic_ptr_cast<tcFermionSpinPtr>(spin));
  // has spin info transfer
  if(sp) {
    vector<EvtDiracSpinor> prod,decay;
    // transfer spinors
    for(unsigned int ix=0;ix<2;++ix) {
      prod .push_back(EvtGenSpinor(sp->getProductionBasisState(ix)));
      decay.push_back(EvtGenSpinor(sp->getDecayBasisState     (ix)));
    }
    myPart->init(id, p4,prod[0],prod[1],decay[0],decay[1]);
    // set the density matrix
    myPart->setSpinDensityForward(EvtGenSpinDensity(sp->rhoMatrix()));
  }
```

# Snippet from Herwig

```
/**
 * Convert our tensor to the EvtGen one.
 * @param ten Our tensor
 */
EvtTensor4C EvtGenTensor(const LorentzTensor<double> & ten) const {
  EvtTensor4C output;
  unsigned int ix,iy;
  for(ix=0;ix<4;++ix){
    for(iy=0;iy<4;++iy) output.set(ix,iy,EvtGenComplex(ten(ix,iy)));
  }
  return output;
}

/**
 * Convert a spin density matrix to an EvtGen spin density matrix.
 * @param rho The spin density matrix to be converted.
 */
EvtSpinDensity EvtGenSpinDensity(const RhoDMatrix & rho) const {
  EvtSpinDensity rhoout;
  unsigned int ix,iy,ispin(rho.iSpin());
  rhoout.setDim(ispin);
  for(ix=0;ix<ispin;++ix) {
    for(iy=0;iy<ispin;++iy)
      rhoout.set(ix,iy,EvtGenComplex(rho(ix,iy)));
  }
  return rhoout;
}
```

# Snippet from Herwig

[EvtGenInterface.h]

```
/**
 * Convert a LorentzSpinor to an EvtGen one. The spinor is converted to the
 * EvtGen Dirac representation/
 * @param sp The LorentzSpinor
 */
EvtDiracSpinor EvtGenSpinor(const LorentzSpinor<SqrtEnergy> & sp) const {
  InvSqrtEnergy norm(sqrt(0.5)/sqrt(GeV));
  EvtDiracSpinor output;
  output.set(EvtGenComplex(-norm*( sp.s1()+sp.s3())),
             EvtGenComplex(-norm*( sp.s2()+sp.s4())),
             EvtGenComplex(-norm*(-sp.s1()+sp.s3())),
             EvtGenComplex(-norm*(-sp.s2()+sp.s4())));
  return output;
}
```

# Summary and discussion

## EvtGen

- Decay weighting: feasible based on alternative BFs, different models/params challenging
- FSR: introduced new alternative using Sherpa's PHOTONS++
- FSR: Vincia under development (envisaged plans for full decay-chain propagation)
- $\tau$ decays: check interface between HERWIG EvtGen

## General

- We can revisit particle ID definitions, but must coordinate with experiments
- Possibility of a "bidirectional" API between main generator and specialist afterburners?