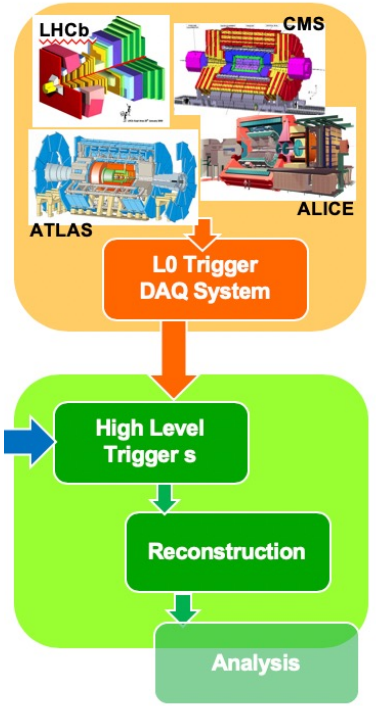
LHCb Simulation - Event Generators

Patrick Robbe, IJCLab Orsay, 13/05/2026

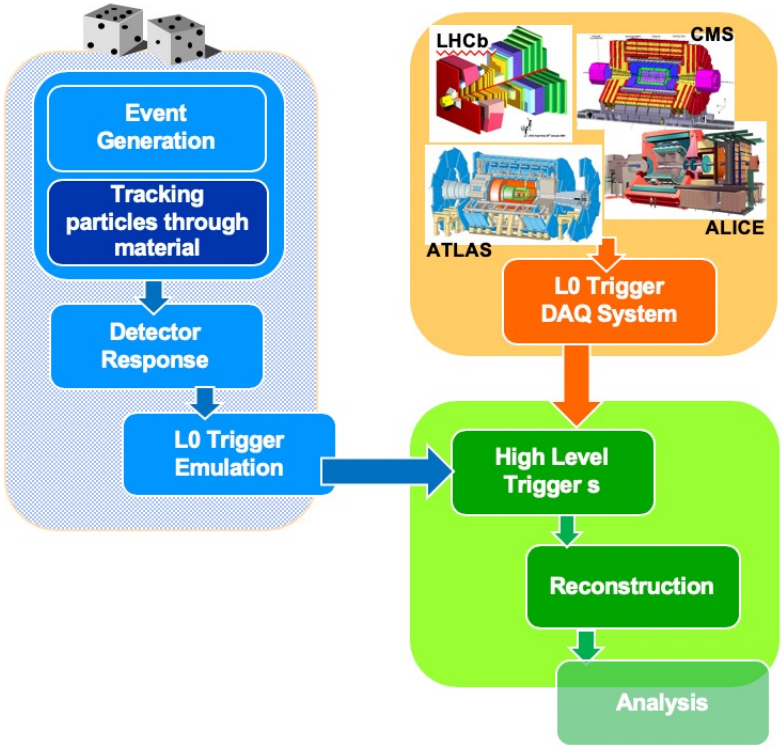
Introduction

- In High Energy Physics (HEP) experiments, when *particle beams* collide in accelerators, a lot of phenomena happen in a single “event”
    - For example, new unstable particles are produced and decay quickly to other particles
- To understand what happens, it is necessary to reconstruct an “image” of the event through measurements by complex detectors and complex software algorithms.

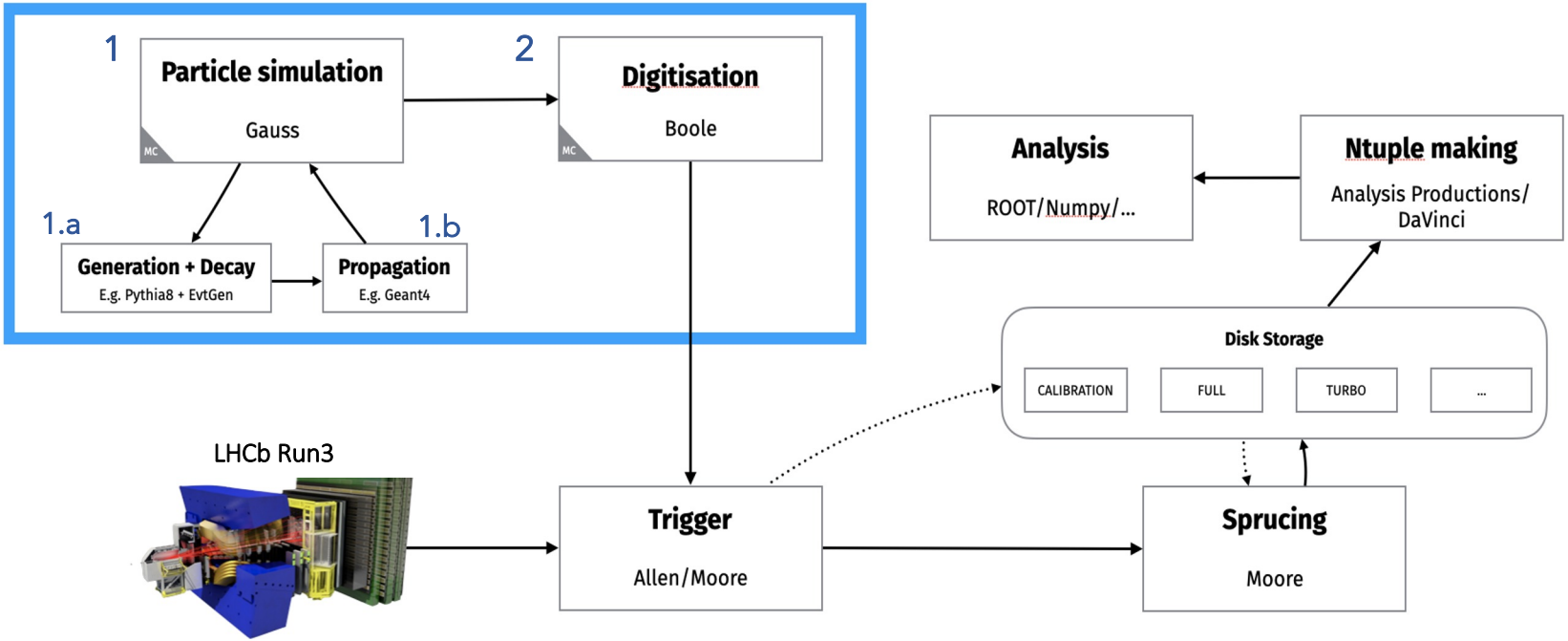


Introduction

- In HEP the role of Monte Carlo simulations is to mimic what happens in the experiments
  - Monte Carlo data are processed as real data to reconstruct the image produced by the detectors… BUT we know the “truth” = what was generated!
  - Comparing the simulation with what is measured allows to understand the experimental conditions and performance and is a key ingredient in interpreting the results.



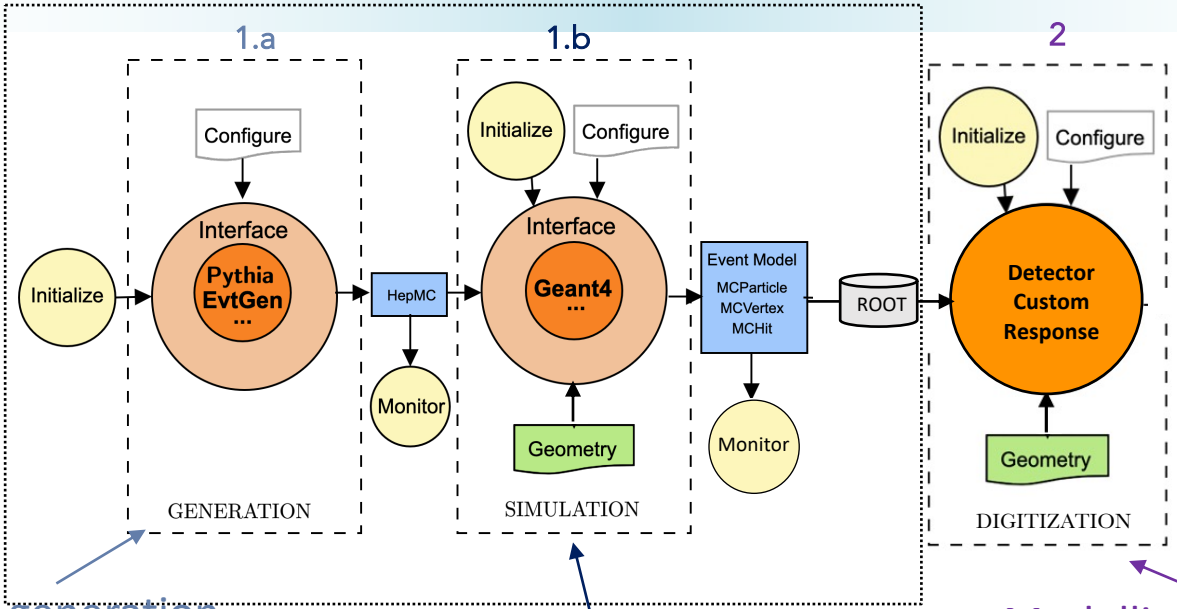
LHCb Data processing of simulated events



Experiments simulation softwares

- HEP experiments have their own software frameworks, providing the infrastructure
  - Athena (ATLAS), CMSSW (CMS), VMC (ALICE, CBM@GSI, Minos), **Gauss** in LHCb, etc.
- and use **external packages** developed in the physics community for generation of events and transport in the detectors
- Response of the detectors is often in-”house” and requires detectors experts
  - tuned first with test beam data, then with measurements in the experiment
  - often in a separate application, **Boole** in LHCb

LHCb Simulation framework



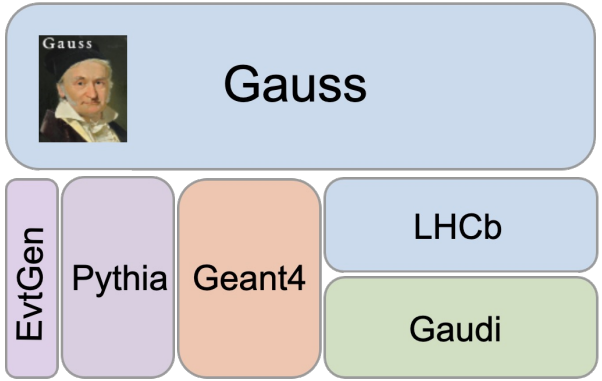
Primary event generation and decay of particles, multiple events per bunch crossings

Transport of particles through detector layout and modelling of physics processes occurring with material

Modelling of detector and its electronics response including imperfections

Gauss – generation and particle transport

- Gauss is the LHCb simulation framework
- It mimics what happens when collisions (or other events) occur in the LHCb spectrometer.
- It provides:
  - generation of proton-proton collisions
    - and other type of events (beam-gas, cosmic rays, calibration, …)
  - decay of particles with special attention to those of b-hadrons
  - handling of beam parameters as luminous regions, pileup (multiple collisions in a beam crossing), spillover (collisions from adjacent beam crossings)
  - tracking of particles in the detector and interactions with the material
  - production of “hits" when particles cross sensitive detectors

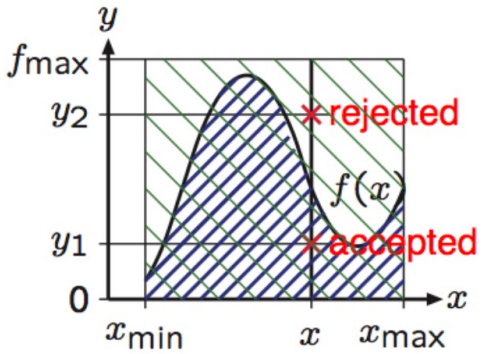


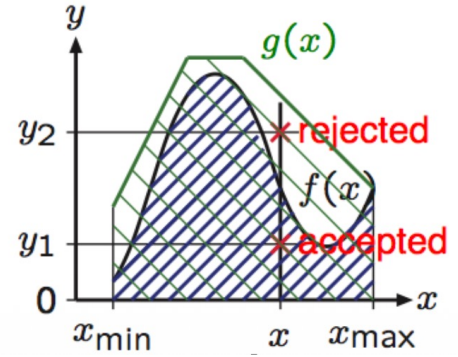
Monte Carlo techniques

- To generate x distributed randomly according to the function $f(x)$ between $x_{\min}$ and $x_{\max}$
- Analytical solution: $x = F^{<-1>}\left(F(x_{\min}) + R\left(F(x_{\max} - x_{\min})\right)\right)$, with R generated uniformly randomly between 0 and 1, and $F(x)$ is a primitive of f, $F(x) = \int_a^x f(y)\mathrm{d}y$ and $F^{<-1>}$ the inverse of F
- For example, to generate exponential decay time τ:
  - $\tau = -\log R, R \in (0,1)$

Monte Carlo techniques

- If f cannot be integrated or inverted:
  - Hit-and-miss method:
    1. Generate randomly R_1 and R_2 uniformly between 0 and 1
    2. Keep $x = x_{min} + R_1(x_{max} - x_{min})$ if $R_2 f_{max} \leq f(x)$
  - Importance sampling method (to improve efficiency):
    1. Find a function $g(x)$ where the analytical method can be applied such that $f(x) < g(x)$ for $x \in (x_{min}, x_{max})$
    2. Generate x following $g(x)$ with the analytical method
    3. Generate R uniformly between 0 and 1
    4. Keep x if $Rg(x) \leq f(x)$





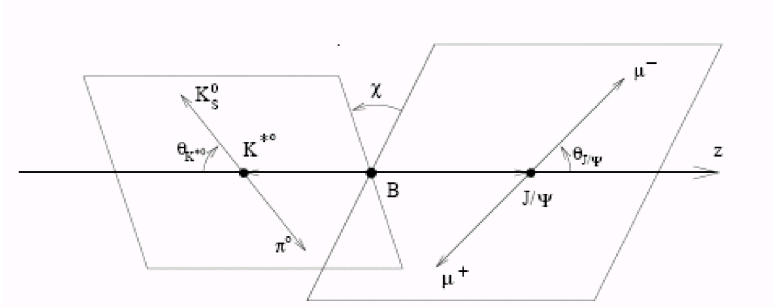
[figures by T. Sjöstrand]

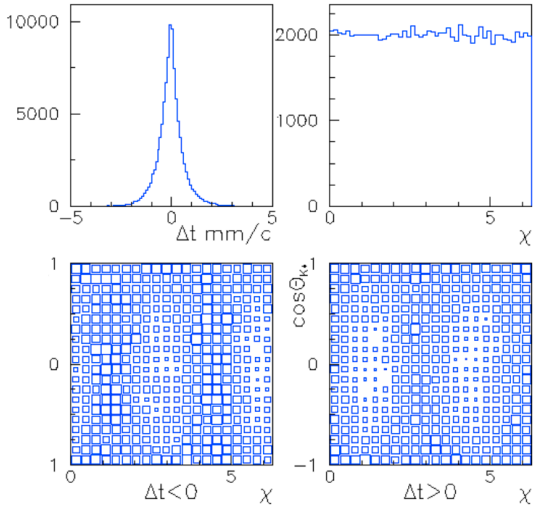
EvtGen

- Important for LHCb to generate the proper characteristics of hadron decays for the main physics program, in particular weak decays of *B*, *D*, τ, etc... Need to model properly for example:
  - CP violation
  - B mixing
  - Decay kinematics, …
- EvtGen is used in LHCb for that:
  - decay generator written originally for the BaBar experiment.
  - BaBar = B factory experiment running at the SLAC e^+e^- collider running at a center of mass energy equal to the Y(4S) mass
  - Adapted in LHCb for pp collisions; used also now Belle II, ATLAS and CMS
  - Implements decays in detail according to theoretical models
  - It is written in C++
  - Can be extended to include new modes easily

Correlations in decays: CP violating decays

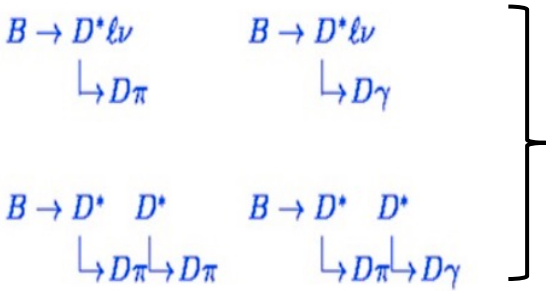
- EvtGen is written to handle complex decays where there can be correlations between the different observables.
- For example in the decay $B^0 \to J/\psi K^{*0},\ J/\psi \to \mu^+\mu^-,\ K^{*0} \to K^0_S\pi^0$ there are correlations between the time and angular distributions





EvtGen

- Many of the interesting decays are sequential decay chains that may need to be considered together (correlations, …)



The D decays will be different depending on the B decay mode*

- But EvtGen needs to generate correctly these decays while only implementing one decay at a time:

$$B \to D^* \ell\nu \qquad B \to D^* D^*$$
$$D^* \to D\pi \qquad D^* \to D\gamma$$

Particle decays

- EvtGen computes for a particle at rest *P*, for example $P \rightarrow A\ B\ C$
  - The proper time τ of *P*
  - The 4-momenta of *A*, *B* and *C*
- The final configuration depends on:
  - Kinematics: phase space
  - Dynamics: physics models
- Algorithm in EvtGen:
  1. Determine the decay tree kinematics according to the masses of A, B, C
  2. Determine the properties of each particle in the tree according to particle types of A, B, C
  3. Accept/reject the kinematics found in 1. to generate the correct dynamics
- For step 1., the probability that A, B, C have 4-momenta of
  $P_A(p_A^0, p_A^1, p_A^2, p_A^3)$, $P_B(p_B^0, p_B^1, p_B^2, p_B^3)$ and $P_C(p_C^0, p_C^1, p_C^2, p_C^3)$
- Is
  $$d\Gamma = \frac{A(2\pi)^4}{2\hbar m_P} \prod_{j=1} 2\pi \frac{d^4 p_j}{(2\pi)^4} \delta(p_j^2 - m_j^2 c^2) \theta(p_j^0) \delta^4(p_P - p_A - p_B - p_C - \dots)$$
- I.e. all configurations respecting momentum conservation and with on-shell particles are equi-probable = phase space

Particle decays

- Dynamic effects (conservation of helicity, decay diagrams, conservation of parity, etc…) are included via helicity amplitudes (and not probabilities): $\mathcal{M}_{\lambda_P,\lambda_A,\lambda_B,\lambda_C} = \langle A_{\lambda_A} B_{\lambda_B} C_{\lambda_C} | H | P_{\lambda_P} \rangle$
- Then the decay probability is:

$$\mathrm{d}\Gamma = \frac{A(2\pi)^4}{2\hbar m_P} |\mathcal{M}|^2 \prod_{j=A,B,C} 2\pi \frac{\mathrm{d}^4 p_j}{(2\pi)^4} \delta\left(p_j^2 - m_j^2 c^2\right) \theta(p_j^0) \delta^4(p_P - p_A - p_B - p_C)$$

- Helicity states are used as a basis $h = \vec{J} \cdot \frac{\vec{P}}{||\vec{P}||} = \vec{s} \cdot \frac{\vec{P}}{||\vec{P}||}$ where $\vec{J} = \vec{s} + \vec{L}$ is the total angular momentum ($\vec{s}$ is the intrinsic spin and $\vec{L}$ the orbital momentum)

Particle decay chains

- In case of decay chains, the use of amplitudes helps to define the total amplitude
- For example, in the case of the decay: $B \to D^* \quad \tau\nu$, $\hookrightarrow D\pi \quad \hookrightarrow \pi\nu$ the total amplitude is

$$\mathcal{M} = \sum_{\lambda_{D^*}\lambda_\tau} \mathcal{M}^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau} \mathcal{M}^{D^*\to D\pi}_{\lambda_{D^*}} \mathcal{M}^{\tau\to\pi\nu}_{\lambda_\tau}$$

EvtGen decay algorithm (1)

- The decays are generated sequentially:
- First the $B \to D^* \tau \nu$:
  - Accept/reject the decay chain generated according to phase space kinematics using the probability

$$P = \sum_{\lambda_{D^*}\lambda_\tau} \left| \mathcal{M}^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau} \right|^2$$

  - Regenerate $B \to D^* \tau \nu$ decay until one is accepted
  - Compute the D^* production helicity density matrix ρ^{D^*} (=probability to produce the D^* in a given helicity state in the B decay), averaging over the τ helicities. The matrix elements are:

$$\rho^{D^*}_{\lambda_{D^*}\lambda'_{D^*}} = \sum_{\lambda_\tau} \left(\mathcal{M}^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau} \right) \left(\mathcal{M}^{B\to D^*\tau\nu}_{\lambda'_{D^*}\lambda_\tau} \right)^*$$

EvtGen decay algorithm (2)

- Then the D^* decay is generated:
  - Accept/reject according to the probability

$$P = \sum_{\lambda_{D^*}\lambda'_{D^*}} \rho^{D^*}_{\lambda_{D^*}\lambda'_{D^*}} \left(\mathcal{M}^{D^*\to D\pi}_{\lambda_{D^*}}\right)\left(\mathcal{M}^{D^*\to D\pi}_{\lambda'_{D^*}}\right)^*$$

  - If rejected, regenerate only the D^* decay, **not** the B decay
- Compute the τ production helicity density matrix ρ^{τ}, using the D^* decay helicity density matrix $\hat{\rho}^{D^*}$ (=proportions of the different D^* helicity states given the D^* decay)

$$\rho^{\tau}_{\lambda_\tau\lambda'_\tau} = \sum_{\lambda_{D^*}\lambda'_{D^*}} \hat{\rho}^{D^*}_{\lambda_{D^*}\lambda'_{D^*}} \left(\mathcal{M}^{B\to D^*\tau\nu}_{\lambda_{D^*}\lambda_\tau}\right)\left(\mathcal{M}^{B\to D^*\tau\nu}_{\lambda'_{D^*}\lambda_\tau}\right)^* \text{with } \hat{\rho}^{D^*}_{\lambda_{D^*}\lambda'_{D^*}} = \left(\mathcal{M}^{D^*\to D\pi}_{\lambda_{D^*}}\right)\left(\mathcal{M}^{D^*\to D\pi}_{\lambda'_{D^*}}\right)^*$$

EvtGen decay algorithm (3)

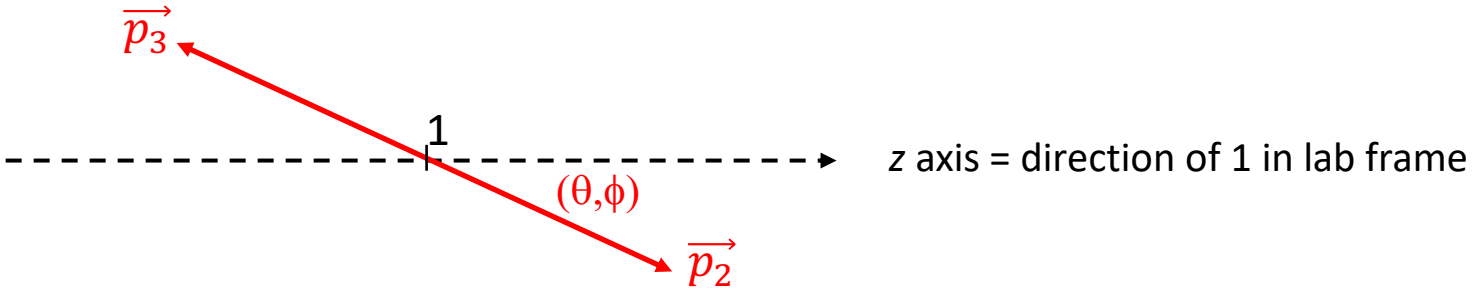
- Then the τ decay is generated:
  - Accept/reject according to the probability

$$P = \sum_{\lambda_\tau \lambda'_\tau} \rho^{\tau}_{\lambda_\tau \lambda'_\tau} \left(\mathcal{M}^{\tau\to\pi\nu}_{\lambda_\tau}\right) \left(\mathcal{M}^{\tau\to\pi\nu}_{\lambda'_\tau}\right)^*$$

  - When the τ decay is rejected, regenerate only the τ decay, **not** the *B* and D^* decays.
- The implementation based on amplitudes is simplified and more efficient than the one using only probabilities.
- The helicity density matrix allows to generate each decay independently
- It can be generalized to any number of subsequent decays
- EvtGen computes helicity density matrix automatically, the user should only provide the helicity amplitudes $\mathcal{M}$

Angular momentum conservation : HELAMP model

- A step further can be done in two-body decays applying total angular momenum/helicity conservation, which is done in the HELAMP model.
- Decay of 1 → 2 + 3, particles with spins s_1, s_2 and s_3.



- Frame = helicity frame = 1 rest frame, the projection of spin s_1 on z is $\boldsymbol{m_1}$
- $\boldsymbol{\lambda_2}$= helicity of 2 = project of spin s_2 on $\overrightarrow{p_2}$
- $\boldsymbol{\lambda_3}$= helicity of 3 = project of spin s_3 on $\overrightarrow{p_3}$

Angular momentum conservation : HELAMP model

- The orbital angular momentum is 0 because $\vec{L} = \vec{r} \times \vec{p}$, so its projection is also 0
- Because of the angular momentum conservation:
  - Total momentum of 2+3 is $J = s_1$
  - The projection of J on $\overrightarrow{p_2}$ is $\lambda_2 - \lambda_3$, with $|\lambda_2 - \lambda_3| \leq J$
- In the helicity frame, the helicity amplitude $\mathcal{M}$ can be written as

$$\mathcal{M}(\theta, \phi; m_1, \lambda_2, \lambda_3) = \sqrt{\frac{2s_1 + 1}{4\pi}} D^{s_1 *}_{m_1, \lambda_2 - \lambda_3}(\phi, \theta, -\phi)\, \mathcal{A}_{\lambda_2, \lambda_3}$$

  - D are the Wigner D-functions, $D^{j}_{m',m}(\alpha, \beta, \gamma) = e^{-i\alpha m'} d^{j}_{m'm}(\beta) e^{-i\gamma m}$
  - d are the Wigner d-functions (given in the PDG)

Angular momentum conservation : HELAMP model

- $\mathcal{A}_{\lambda_2,\lambda_3}$ are called (also) helicity amplitudes: they describe the probability of each helicity configuration. They contain the physics of the decay (they can depend on the momentum for example)
- If parity is conserved (P_1, P_2, P_3 = parities of 1, 2, 3):

$$\mathcal{A}_{-\lambda_2,-\lambda_3} = P_1 P_2 P_3 (-1)^{s_2+s_3-s_1} \mathcal{A}_{\lambda_2\lambda_3}$$

- For a given (λ_2,λ_3) configuration, the angular distribution is given by

$$\frac{\mathrm{d}\Gamma}{\mathrm{d}\Omega}_{m_1,\lambda_2,\lambda_3}(\theta,\phi) = \left|\mathcal{M}(\theta,\phi;m_1,\lambda_2,\lambda_3)\right|^2 = \frac{2s_1+1}{4\pi}\left|d^{s_1}_{m_1,\lambda_2-\lambda_3}(\theta)\right|^2 \left|\mathcal{A}_{\lambda_2\lambda_3}\right|^2$$

- NB: the ϕ dependency disappears as only θ, the angle between the quantization axis and the particle momentum, is physical

Angular momentum conservation : HELAMP model

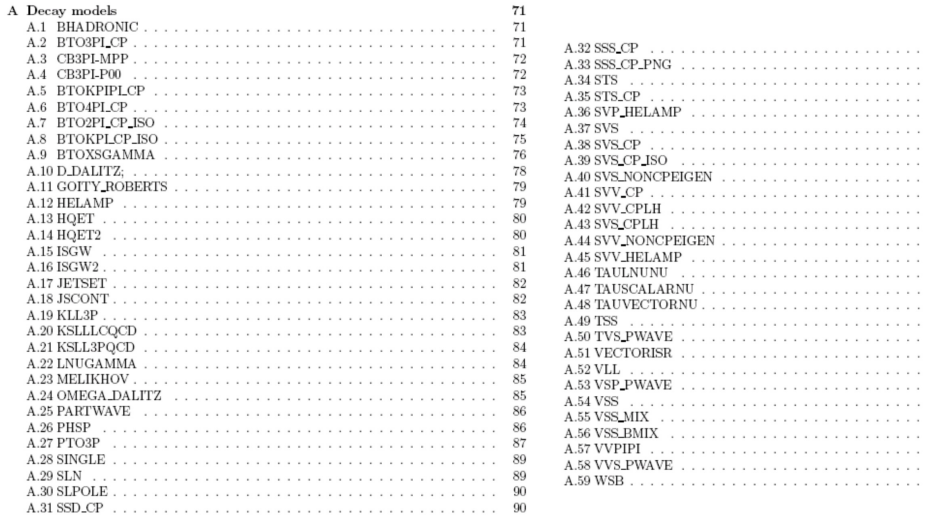
- For the total angular distribution, one must sum incoherently over the final state helicities:

$$\frac{\mathrm{d}\Gamma}{\mathrm{d}\Omega}_{m_1}(\theta,\phi) = \sum_{\lambda_2,\lambda_3} \frac{\mathrm{d}\Gamma}{\mathrm{d}\Omega}_{m_1,\lambda_2,\lambda_3}(\theta,\phi) = \frac{2s_1+1}{4\pi} \sum_{\lambda_2,\lambda_3} \left| d^{s_1}_{m_1,\lambda_2-\lambda_3}(\theta) \right|^2 |\mathcal{A}_{\lambda_2\lambda_3}|^2$$

- For a decay chain, this is done at each step of the chain. In that case, the sum over the helicities of the intermediate steps is done coherently as these are unmeasurable quantities: this is taken care by EvtGen automatically
- The final angular distribution depends on the polarisation of the particle 1, i.e. the repartition of the m_1 projection states in the initial state.

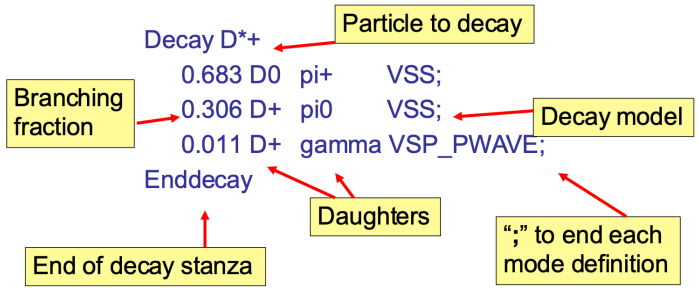
EvtGen decay models

- Many decay models are available already in EvtGen, to generate:
  - CP violation for various decay modes
  - Semileptonic decays with several form factor models
  - Three-body decay with resonance structure over Dalitz plane: *D* ($D^0 \to K^-\pi^+\pi^0$ for example), ω, η
  - $b \to sll$ or $b \to s\gamma$ decays
- List of all models:



Decay files

- EvtGen is configured at run time with decay files, using the following syntax:



- Some decay models have arguments that are passed also in the decay file. For example, for the HELAMP model:

Decay B+

1.000 anti-D*0 rho HELAMP 0.228 0.95 0.283 1.13 0.932 0;

Enddecay

$\mathcal{A}_{-1,-1} = 0.228e^{0.95i},\ \mathcal{A}_{0,0} = 0.282e^{1.13i},\ \mathcal{A}_{1,1} = 0.932e^{0i}$

$|A_{\lambda_2\lambda_3}|$ $Arg(A_{\lambda_2\lambda_3})$ ordered by λ_2, then λ_3

Decay files

- LHCb uses also a very large generic decay file, called DECAY.DEC that covers all known particles and decays.
- Improved with respect to the original file developed by BaBar and Belle to add B_s^0, Λ_b, B_c, … decays.
- Extract of the B_c section:

```
10130 Decay B_c-
10131 0.01600    tau-       anti-nu_tau              SLN;
10132 #
10133 #                SemiLeptonic Decays
10134 0.01900    J/psi      e-      anti-nu_e        PHOTOS  PHSP;
10135 0.00094    psi(2S)    e-      anti-nu_e        PHOTOS  PHSP;
10136 0.00750    eta_c      e-      anti-nu_e        PHOTOS  PHSP;
10137 0.00020    eta_c(2S)  e-      anti-nu_e        PHOTOS  PHSP;
10138 0.00004    anti-D0    e-      anti-nu_e        PHOTOS  PHSP;
10139 0.00018    anti-D*0   e-      anti-nu_e        PHOTOS  PHSP;
10140 0.04030    anti-B_s0  e-      anti-nu_e        PHOTOS  PHSP;
10141 0.05060    anti-B_s*0 e-      anti-nu_e        PHOTOS  PHSP;
10142 0.00340    anti-B0    e-      anti-nu_e        PHOTOS  PHSP;
10143 0.00580    anti-B*0   e-      anti-nu_e        PHOTOS  PHSP;
10144 #
10145 0.01900    J/psi      mu-     anti-nu_mu       PHOTOS  PHSP;
10146 0.00094    psi(2S)    mu-     anti-nu_mu       PHOTOS  PHSP;
10147 0.00750    eta_c      mu-     anti-nu_mu       PHOTOS  PHSP;
10148 0.00020    eta_c(2S)  mu-     anti-nu_mu       PHOTOS  PHSP;
10149 0.00004    anti-D0    mu-     anti-nu_mu       PHOTOS  PHSP;
10150 0.00018    anti-D*0   mu-     anti-nu_mu       PHOTOS  PHSP;
10151 0.04030    anti-B_s0  mu-     anti-nu_mu       PHOTOS  PHSP;
10152 0.05060    anti-B_s*0 mu-     anti-nu_mu       PHOTOS  PHSP;
10153 0.00340    anti-B0    mu-     anti-nu_mu       PHOTOS  PHSP;
10154 0.00580    anti-B*0   mu-     anti-nu_mu       PHOTOS  PHSP;
10155 #
```

User decay files: signal decay generation

- To simulate signal decays in the LHCb simulation (most useful for analyses in LHCb), one needs to write a decay file to specify the decay modes of the signal decay.
- This requires using ‘aliases’ so that a decay of a particle can be forced in a specific final state without changing the other particles decaying generically.
- For example, to force the decay $D^0 \to K^- \pi^+$:

```
Alias MyD0   D0
Alias Myanti-D0 anti-D0
ChargeConj MyD0  Myati-D0
Decay MyD0
  1.000 K- pi+  PHSP;
Enddecay
CDecay Myanti-D0
```

- Predefined aliases are mandatory for the top of the decay chain so that it is used correctly by the Gauss framework: <particle name>sig (ex B0sig, D0sig, …)

User decay files: signal decay generation

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

```
+ # EventType: 27165078
+ #
+ # Descriptor: [D*+ -> (D0 -> K- pi- pi+ pi+) pi+]cc
+ #
+ # NickName: Dst_D0pi,Kpipipi=TightCuts,AmpGen,PHOTOS
+ #
+ # Cuts: LoKi::GenCutTool/TightCut
+ #
+ # InsertPythonCode:
+ # #
+ # from Configurables import LoKi__GenCutTool
+ # from Gauss.Configuration import *
+ # gen = Generation()
+ # gen.SignalPlain.addTool ( LoKi__GenCutTool , 'TightCut' )
+ # minPTAndDaughtersInLHCb = gen.SignalPlain.TightCut
+ # minPTAndDaughtersInLHCb.Decay     = '^[ D*(2010)+ => ^( D0 => ^K- ^pi- ^pi+ ^pi+ ) ^pi+ ]CC'
+ # minPTAndDaughtersInLHCb.Preambulo += [
+ #     'from GaudiKernel.SystemOfUnits import MeV ',
+ #     'inAcc     = in_range ( 0.010 , GTHETA , 0.400 ) ',
+ #     'goodD       = ( GPT > 2000 * MeV ) & ( GP > 30000 * MeV )',
+ #     'goodDst     = ( GPT > 2070 * MeV ) & ( GP > 31400 * MeV )'
+ # ]
+ # minPTAndDaughtersInLHCb.Cuts      =    {
+ #     '[pi+]cc'         : 'inAcc',
+ #     '[K+]cc'          : 'inAcc',
+ #     '[D0]cc'          : 'goodD',
+ #     '[D*(2010)+]cc'   : 'goodDst',
+ #     }
+ #
+ # EndInsertPythonCode
+ #
+ #
+ # Documentation:
+ #   CF D decay. Intended to be a copy of 27165071 with PHOTOS enabled.
+ # EndDocumentation
+ #
```

Header (mandatory):

EventType: follows convention (see later)
Descriptor: description of decay mode
NickName: unique name to identify content
Cuts: generator level cut to apply
InsertPythonCode: implementation of the cut

Documentation: free text for documentation

User decay file: signal decay generation

```
37 + # CPUTime: < 1 min
38 + #
39 + # PhysicsWG: Charm
40 + # Tested: Yes
41 + # Responsible: Cheryl Pappenheimer
42 + # Email: c.pappenheimer@cern.ch
43 + # Date: 20260331
44 + #
45 + Alias MyD0         D0
46 + Alias Myanti-D0    anti-D0
47 + ChargeConj MyD0    Myanti-D0
48 +
49 + Decay D*+sig
50 +   1.000 MyD0 pi+  VSS;
51 + Enddecay
52 + CDecay D*-sig
53 +
54 + Decay MyD0
55 +   1.0 K- pi+ pi+ pi- LbAmpGen DtoKpipipi;
56 + Enddecay
57 + CDecay Myanti-D0
58 + #
59 + End
60 + #
```

CPUTime: time needed for generation (to adjust time limits on the grid)
PhysicsWG, Tested, Responsible, Email, Date: to identify person in charge

Aliases definitions

Decay of the predefined alias (D*+sig)

Decay of the other particles in the chain (with custom aliases, not the predefined ones)

Event type convention

- Described in the note LHCb-2005-034
- Number with 8 digits: **GSDCTNXU**
- **G:** General event type and production scheme: 1 = minimum bias, 2 = charm signal, 3 = beauty signal, 4 = exotics
- **S:** Signal decay, for example if G=2, S=1 for B^0, 2 for B^+, 3 for B_s, 4 for B_c^+, 5 for Λ_b, …
- **D:** Decay flag, D=0 if no decay is forced, 1 if forced to a unique final state, …
- **C:** Based on the number of charm hadrons and leptons present.
- **T:** Number of stable charged particles: p, π, K, e, μ.
- **N:** Number of neutral particles: K_S, Λ, K_L, γ, n, π^0, η.
- **X,U:** Extra flags to distinguish between different event types.

Adding new models

- New models can be added to EvtGen if the process is not yet included in the existing models.
- It consists in implementing in C++ the computation of the helicity amplitudes given the final state kinematics.
- For example, if one wants to implement a model for a vector meson (spin 0) to two scalar particles (spin 0). The amplitude for such decay is given by

$$\mathcal{M} = \prod_{\mu=0..2} \epsilon^{\mu} v_{\mu}$$

- Where ε is the polarisation vector of the initial meson and *v* the velocity of one scalar particle in the vector meson rest frame.

Adding new models

- Define class with mandatory functions:

```
class EvtVSS:public  EvtDecayAmp  {

public:
  EvtVSS() {}
  virtual ~EvtVSS();

  void getName(std::string& name);
  EvtDecayBase* clone();

  void decay(EvtParticle *p);
  void init();
  void initProbMax();

};
```

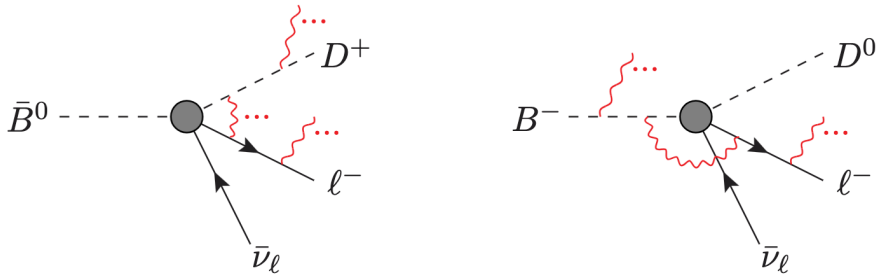
(unique name)

(compute amplitude)
(sanity checks)
(max proba for accept/reject method)

```
void EvtVSS::decay( EvtParticle *p){

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

 EvtVector4R pdaug = p->getDaug(0)->getP4();

 double norm=1.0/pdaug.d3mag();
 vertex(0,norm*pdaug*(p->eps(0)));
 vertex(1,norm*pdaug*(p->eps(1)));
 vertex(2,norm*pdaug*(p->eps(2)));

 return;
}

void EvtVSS::init(){
  // check that there are 0 arguments
  checkNArg(0);

  // check that there are 2 daughters
  checkNDaug(2);

  // check the parent and daughter spins
  checkSpinParent(EvtSpinType::VECTOR);
  checkSpinDaughter(0,EvtSpinType::SCALAR);
  checkSpinDaughter(1,EvtSpinType::SCALAR);
}

void EvtVSS::initProbMax() {
   setProbMax(1.0);
}
```

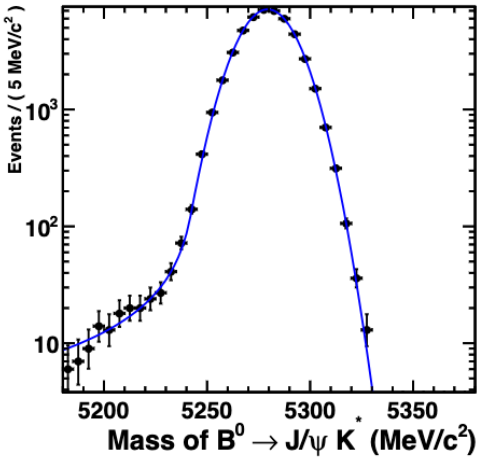
PHOTOS

- QED corrections affects decays with charged particles: photons are emitted modifying the momenta of the charged particles. Most of the photons are soft and escape detection by the detector.



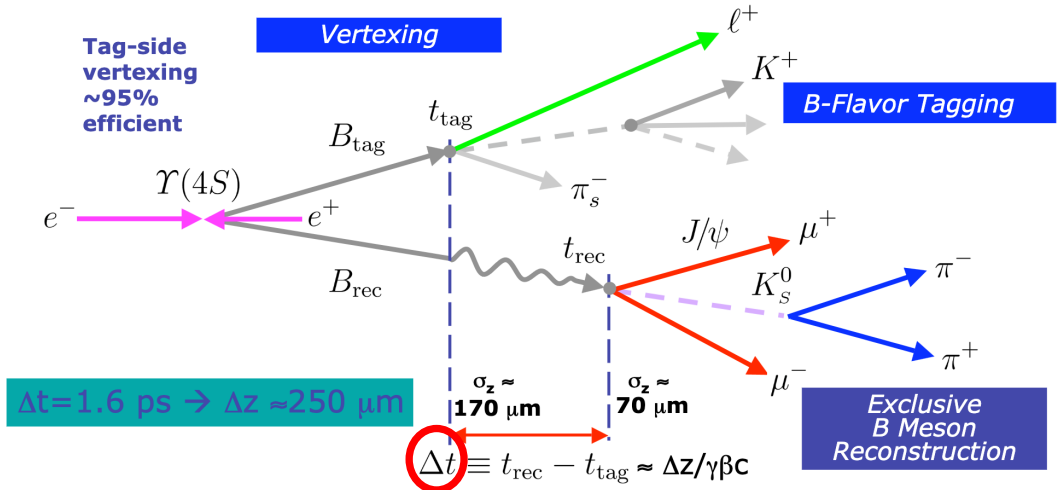
- They are however always generated by EvtGen using internally a dedicated generic generator called PHOTOS [E. Barberio, B. van Eijk, and Z. Was, Comput. Phys. Commun. 66 (1991) 115.] for all kind of decays, with many charged particles in the final state.
- The effect of the radiated photons in visible on the generated decays:

Generator Level



Adaptation of EvtGen for LHCb

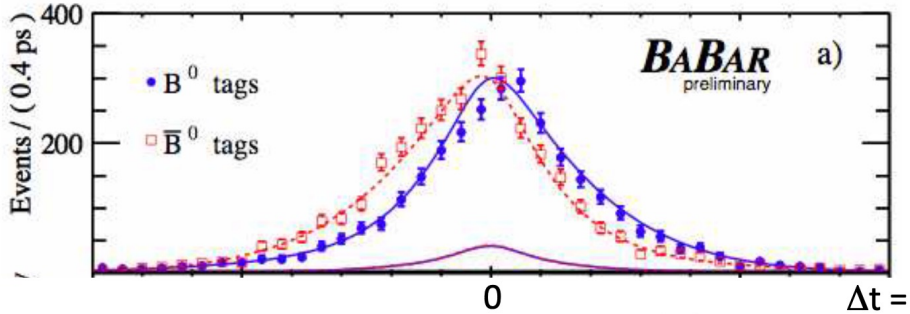
- A major difference between BaBar/Belle (original EvtGen implementation) and LHCb is that B are produced coherently in BaBar/Belle and incoherently in LHCb.
- This impact how the generation of CP violation (and mixing) is done in EvtGen between both configurations.
- The relevant variable is **Δt** in the case of BaBar/Belle



CP violation – EvtGen – BaBar/Belle case

- The problem is illustrated in time dependent CP violation in $B^0 \rightarrow J/\psi\, K_S^0$

$\Upsilon(4S) \rightarrow B^0\overline{B^0}$



Δt = time between tag and signal decays

A realistic Monte Carlo signal sample contains N(B^0 tags)=N($\overline{B^0}$ tags)

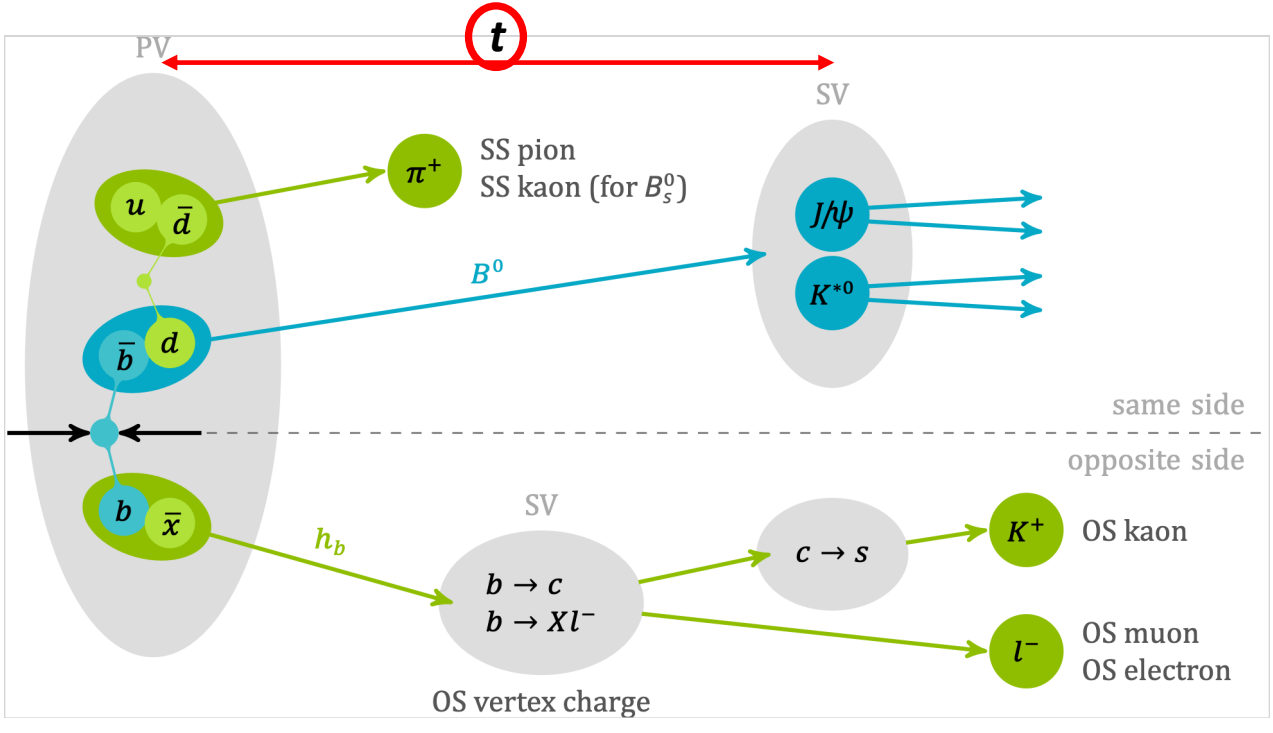
Time distribution is generated following the probabilities, in EVTGEN:

B^0 tags $\quad \Gamma(t) = \frac{\Gamma_B}{4} e^{-\Gamma_B t}\left[1 + \sin(2\beta)\sin(\Delta m t)\right]$

$\overline{B^0}$ tags $\quad \Gamma(t) = \frac{\Gamma_B}{4} e^{-\Gamma_B t}\left[1 - \sin(2\beta)\sin(\Delta m t)\right]$

Adaptation of EvtGen for LHCb

- For incoherent *B* production in LHCb, the relevant variable is ***t***

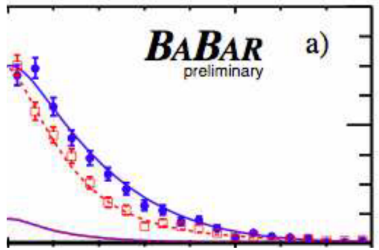


CP violation – EvtGen – LHCb case

- The problem is illustrated in time dependent CP violation in $B^0 \to J/\psi\, K_S^0$

$pp \to B^0\overline{B}X$

B tags
$\overline{B}$ tags



t = time between production and decay

A realistic signal sample contains N(B tags) ≠ N($\overline{B}$ tags), but PYTHIA produces equal numbers of B^0 and $\overline{B}^0$.
In EVTGEN, generate time and B flavour according to

$$\Gamma(t,tag) = \begin{cases} \frac{\Gamma_B}{4} e^{-\Gamma_B t}\left[1+\sin(2\beta)\sin(\Delta m t)\right], tag = B \\ \frac{\Gamma_B}{4} e^{-\Gamma_B t}\left[1-\sin(2\beta)\sin(\Delta m t)\right], tag = \overline{B} \end{cases}$$

and keep only events where PYTHIA and EVTGEN agree.