A fully featured interface between Pythia8 and Geant4

And some things that I've learned along the way

Einar Elén

Caveats

Don't start out a talk by apologizing

I will have some caveats at the end instead

But that's cheating

Caveats

Probably don't start out a talk with something meta either

Remember to remove this slide

Outline

- You don't need an outline
- It usually just takes up time

Outline

- What is this?
- Background
  - Event generators and detector simulation
- Vocabulary and target audiences
- Maybe some physics

What is this?

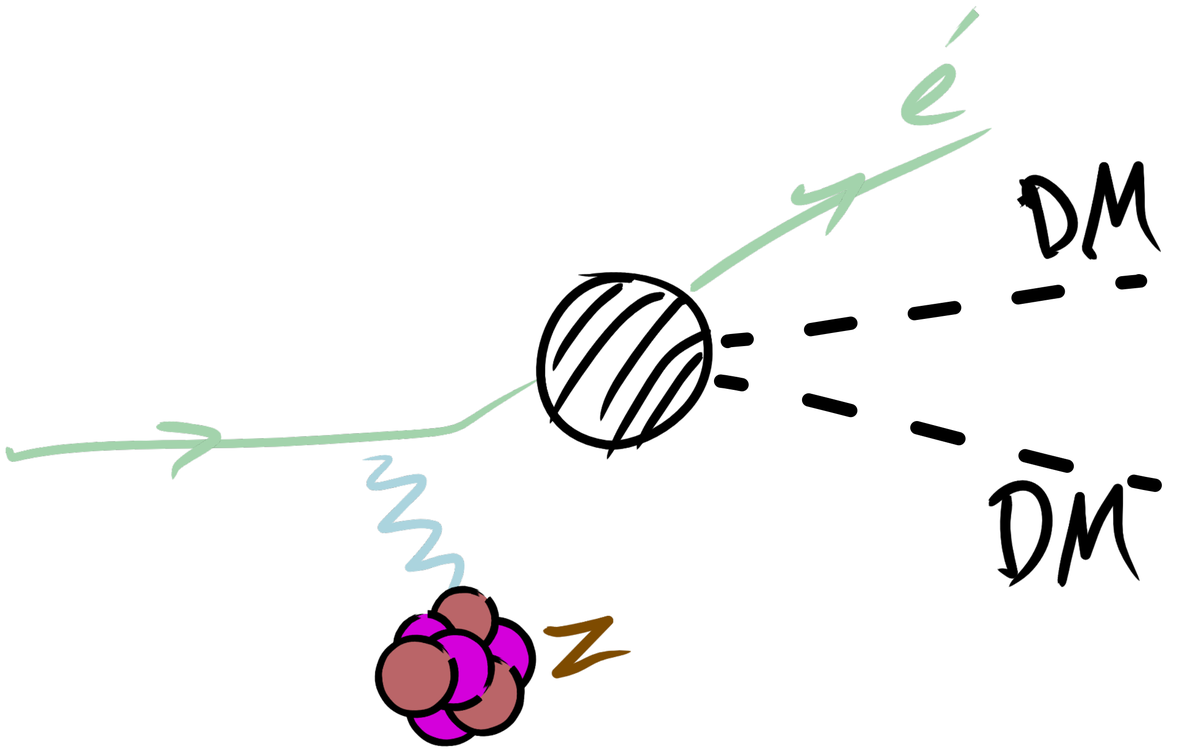
- A software library and framework for embedding Pythia8 programs inside Geant4 simulations
  - Built as a separate library
    - Works with ancient versions of Geant4 if needed :)
- Currently lacking branding
  - Can’t call it Apollo any more [2403.19452]

Goals and challenges

- Event generation and detector simulation have developed rather independently
  - Both are sophisticated and highly technical endeavours
  - Simultaneous expertise in both is rare
- We cannot require that users are fluent in both
- Two target audiences with very different backgrounds

Unexpected challenges

- Language and vocabulary is hard
- Most terms are overloaded
  - What does the word "Process" mean?
- One person's precise technical language is another person's jargon
- Software makes it exponentially worse



This doesn't count as apologizing

Brief background

Why?

You can't always factorize event generation from detector simulation

There's physics in detector simulations as well

Geant4 in brief

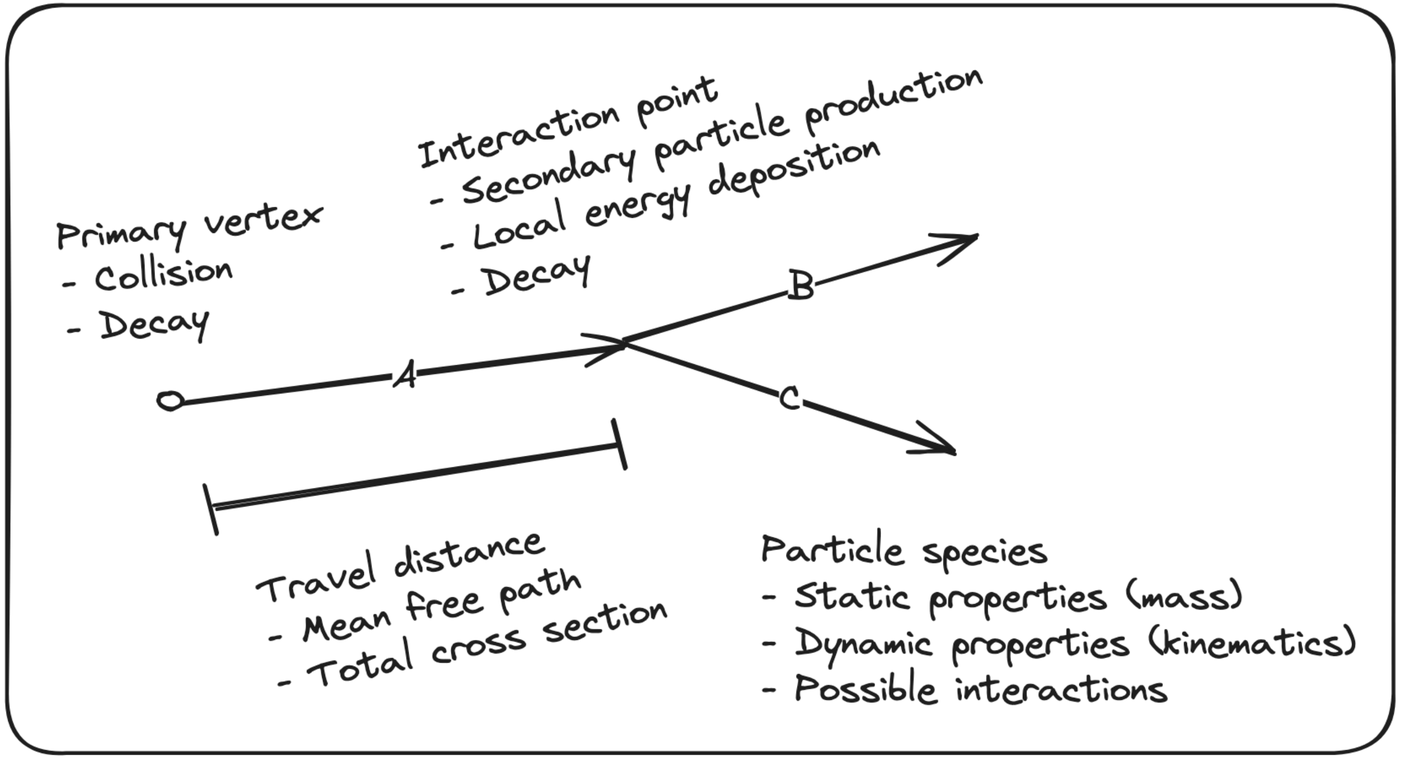
- C++ library for designing your own detector simulation application
  - Covers a huge range of different use cases and areas of physics
- A big collection of ”components”
  - Pick what you need for your simulation
  - All made of further subcomponents...

When I say comprehensive:

We cover the vast majority of relevant components and some of the irrelevant ones as well

Geant4 in brief

- Key physics objects:
  - Particle definitions
  - Processes



Geometry, material, and field configuration

Is there physics yet?

Soon

Maybe

What can an event generator bring to Geant4?

Cross section estimates

Final states

"Exotic" particles

Interacting with the framework

Mental model

Theory/Pheno expertise

Physics configuration

+

SetBeams(projectile, target)

Experiment expertise

Geant4 Component choice

+

Experiment-specific standard integration

From the theory/pheno perspective

- I have
  - A Pythia 8 program simulating dark bremsstrahlung based on a MadGraph model
- I want to
  - Simulate this in my detector model
- I need to provide Geant4 with
  - New particles ($A', \chi, \bar{\chi}$)
  - $\sigma(eW \to e\,W A')$
  - Events

```
Pythia8::Pythia pythia{};
pythia.setSigmaProces(TaylorsDarkPhotonModel{});

pythia.readString("Aprime:mass = 1.0");
pythia.readString("DarkPhoton:epsilon = 1e-3");
pythia.readString("Beams:idA = 11"); // electron
// The code used for the nucleus in the model, defined in the UFO code
pythia.readString("Beams:idB = 900003");
pythia.readString("Beams:eCM = 4");
pythia.init();

for (int i{0}; i < 1000; ++i) {
    pythia.next();
}
std::cout << "Dark bremsstrahlung cross section is "
    << pythia.info.sigmaGen() << std::endl
```

From the experiment point of view

```
G4PythiaEMProcess DarkBrem{};

DarkBrem.setSigmaProcess(TaylorsDarkPhotonModel{});
DarkBrem.readString("Aprime:mass = 1.0");
DarkBrem.readString("DarkPhoton:epsilon = 1e-3");

// Geant4-syntax to tell all electrons to do a new process
G4Electron::GetProcessManager()->AddProcess(DarkBrem);
G4Electron::GetProcessManager()->AddCrossSection(DarkBrem);
```

Add the component exactly as you would any other part of Geant4

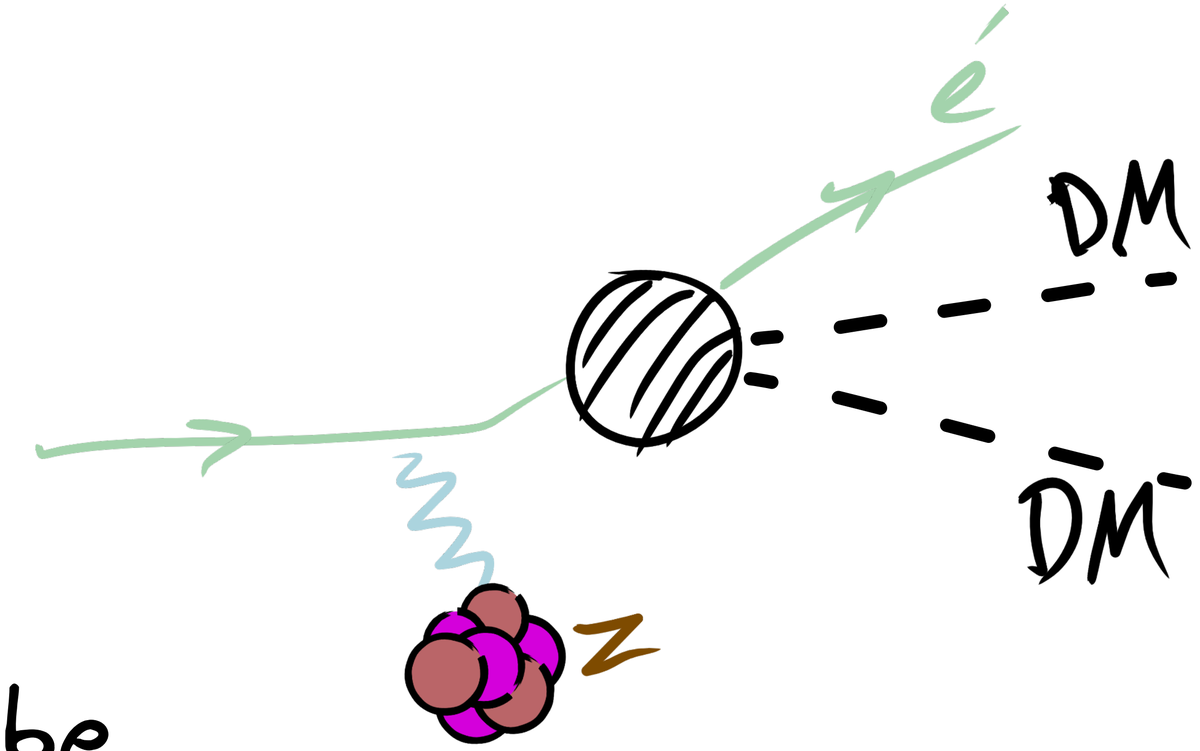
Tell Apollo the physics configuration:

Exactly the same syntax as the original Pythia program for the physics configuration

What about the physics?

Are you even doing any?

Ok here is some physics

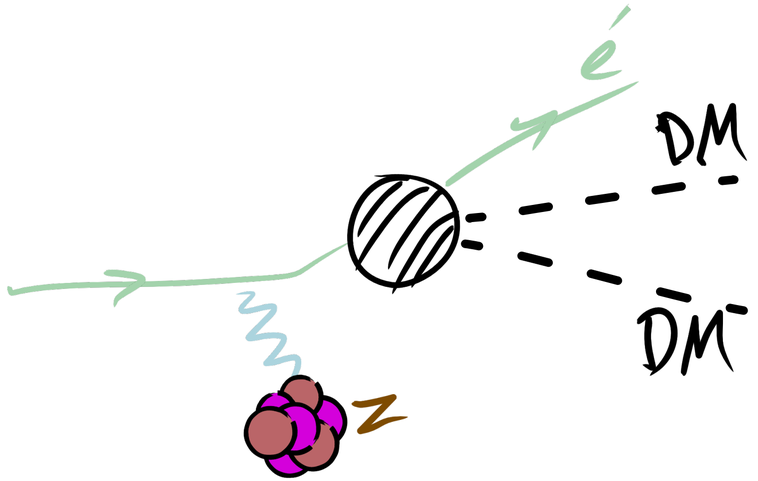


How hard could it be

How hard can it be

Its SM bremsstrahlung with extra steps

From Blinov et al: Dark fluxes from electromagnetic cascades



F Dark vector bremsstrahlung

The mass of the dark sector particles qualitatively changes the energy and angular distribution of its emission compared to its SM counterpart. These important features are determined by the soft and collinear singularity structure of the amplitude. This means that accurate simulation of this interaction can be numerically challenging despite the simplicity of the underlying process. In this Appendix we validate our implementation of dark bremsstrahlung and compare it to existing tools.

Hadronic physics in Geant4

Primarily Geant4's version of FRITIOF: FTF

I would like for it to not to be

Hadronic physics in Geant4 from Pythia

hh hA AA

PythiaCascade
Angantyr
Pythia

Alternatives for both cross sections and final states

Cross sections

Coherent elastic ($h + A \rightarrow h + A$)

Quasi elastic ($h + A \rightarrow h + (A-1) + N$)

Inelastic ($h + A \rightarrow X$)

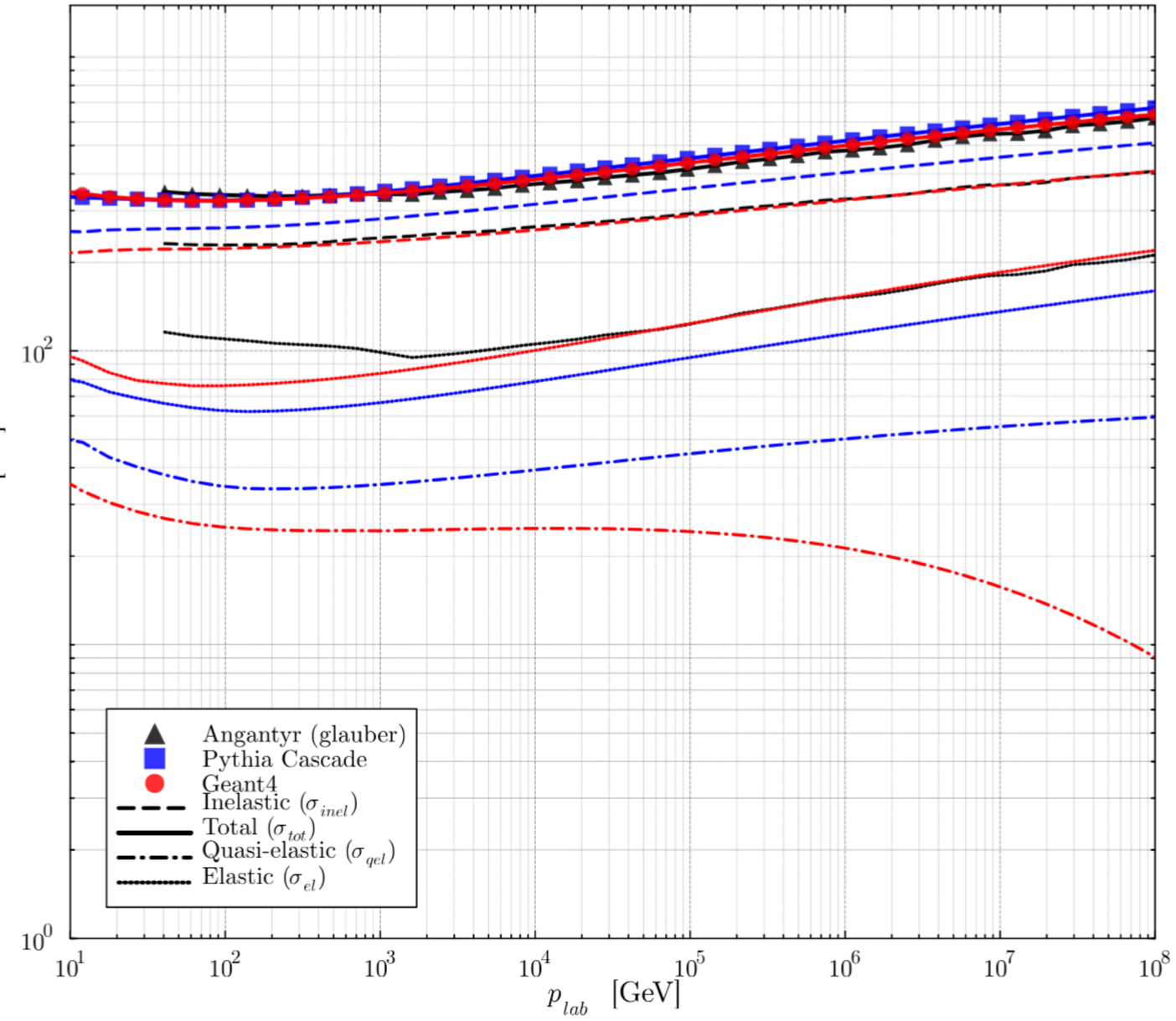
Big caveat

Some extensions made by me

An idiot

Not validated anywhere

Don't spread them around

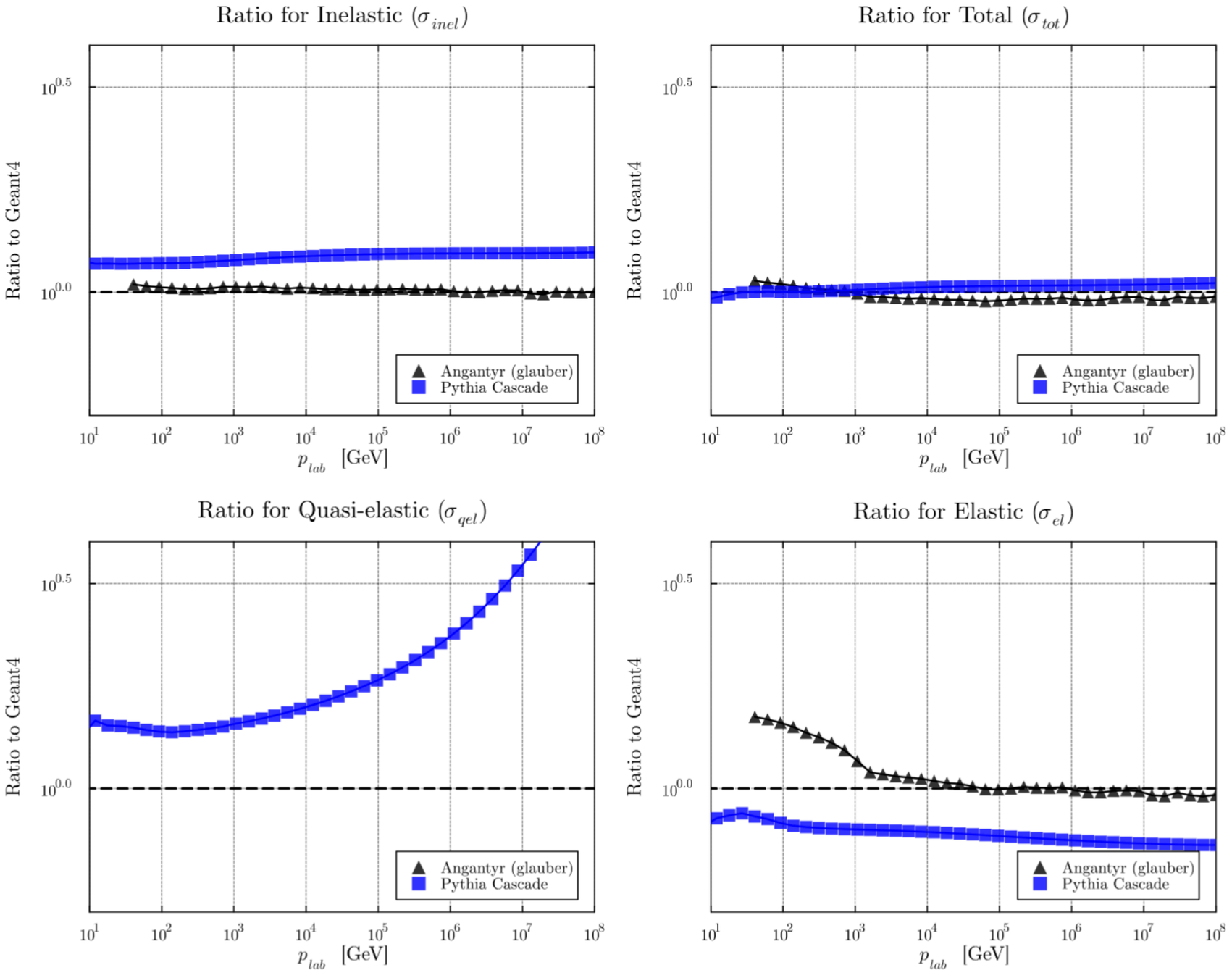


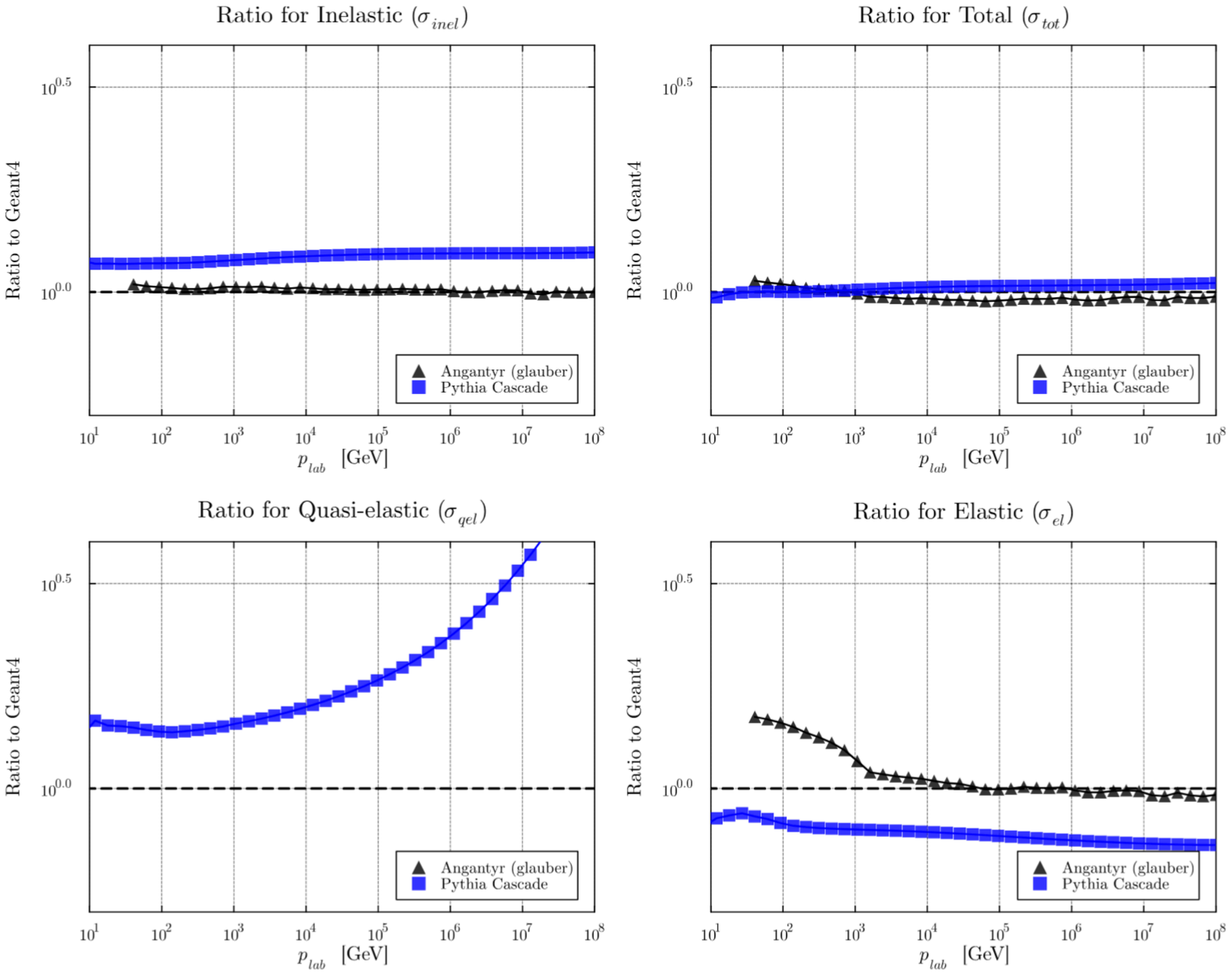
Proton on carbon collisoins

Coherent elastic (h + A -> h + A)

Quasi elastic (h + A -> h + (A-1) + N)

Inelastic (h + A -> X)

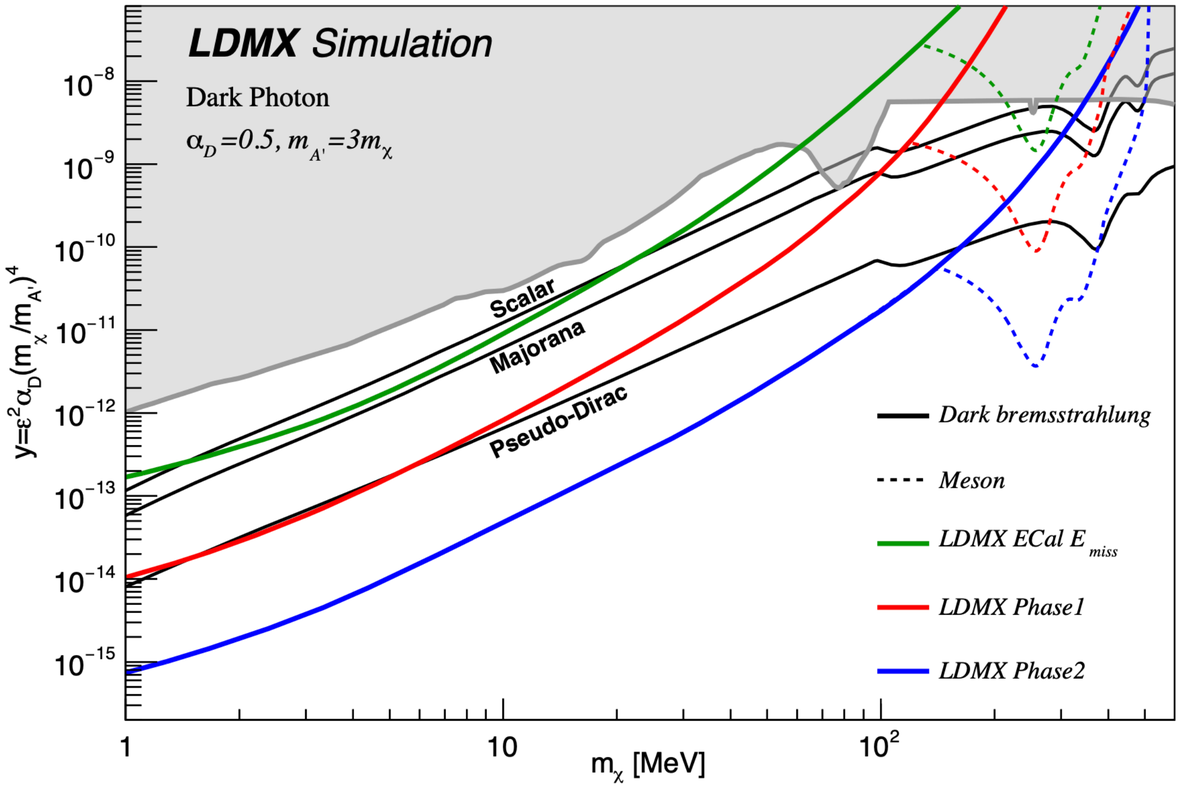

Ratio for Inelastic (σ_{inel})
Ratio for Total (σ_{tot})
Ratio for Quasi-elastic (σ_{qel})
Ratio for Elastic (σ_{el})
Ratio to Geant4
$10^{0.5}$
$10^{0.0}$
10^{1} 10^{2} 10^{3} 10^{4} 10^{5} 10^{6} 10^{7} 10^{8}
p_{lab} [GeV]
Angantyr (glauber)
Pythia Cascade


Ratio for Inelastic (σ_{inel})
Ratio for Total (σ_{tot})
Ratio for Quasi-elastic (σ_{qel})
Ratio for Elastic (σ_{el})
Ratio to Geant4
$10^{0.5}$
$10^{0.0}$
10^{1} 10^{2} 10^{3} 10^{4} 10^{5} 10^{6} 10^{7} 10^{8}
p_{lab} [GeV]
Angantyr (glauber)
Pythia Cascade

What I'd like to do more of

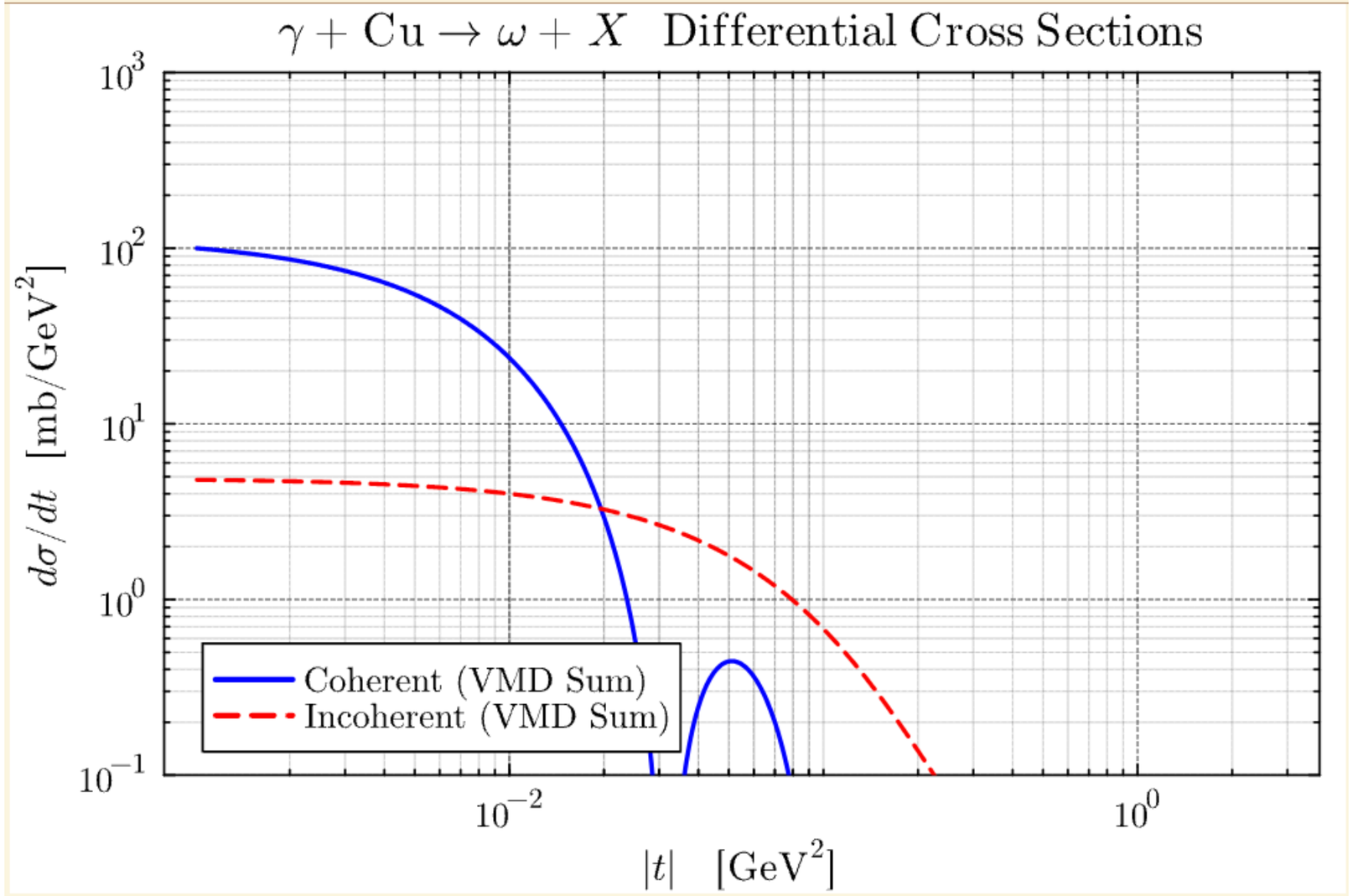
Photoproduction

In fact, a simulation-based approach is also inadequate, as neither GEANT's hadronic models nor particle physics Monte Carlo simulations such as PYTHIA [59] include careful modeling of exclusive photoproduction processes. PYTHIA's parton-based modeling is designed for the deep-inelastic regime, while our reactions of interest are in the diffractive regime. Meanwhile, GEANT4 [60] does not include short-lived resonances in its hadronic models, but rather treats reactions such as $\gamma p \to \rho p$ as a component of, e.g., $\gamma p \to \pi^+\pi^- p$. More specialized programs such as



A′ results from Åkesson et al., "Photon-Rejection Power of the Light Dark Matter eXperiment in an 8 GeV Beam." JHEP12(2023)092 [arxiv:2308.15173]

Invisible meson decays from Schuster, Toro, and Zhou, "Probing Invisible Vector Meson Decays with the NA64 and LDMX Experiments." (Phys. Rev. D 105, 035036) [arxiv:2112.02104]



Toy implementation

Caveats

Don't start out a talk by apologizing

I will have some caveats at the end instead

But that's cheating

Caveats

Don't do it at the end either

Caveats

But who's going to stop me

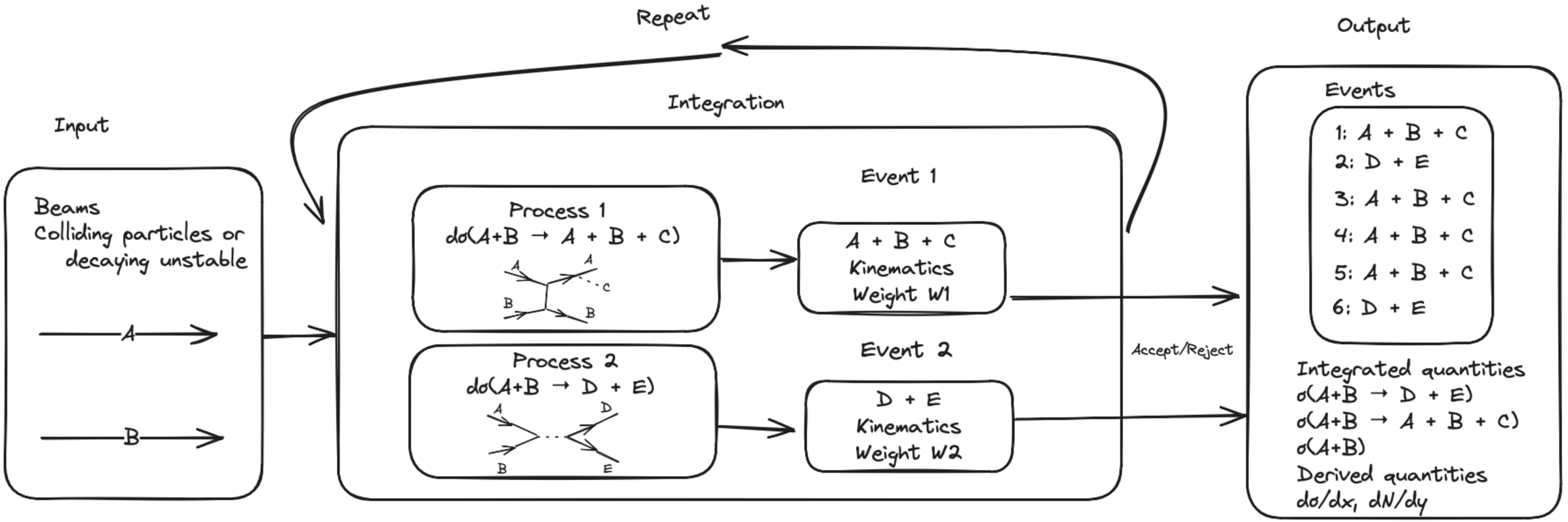
I will do it anyways

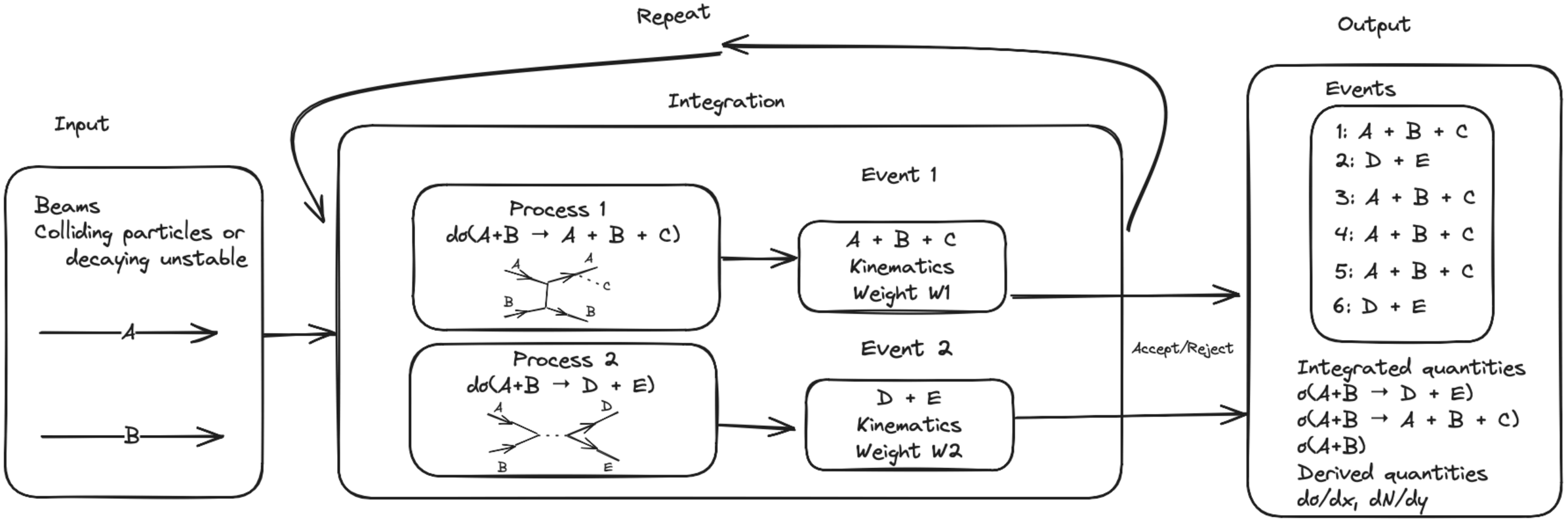
I'm not doing great

Thank you

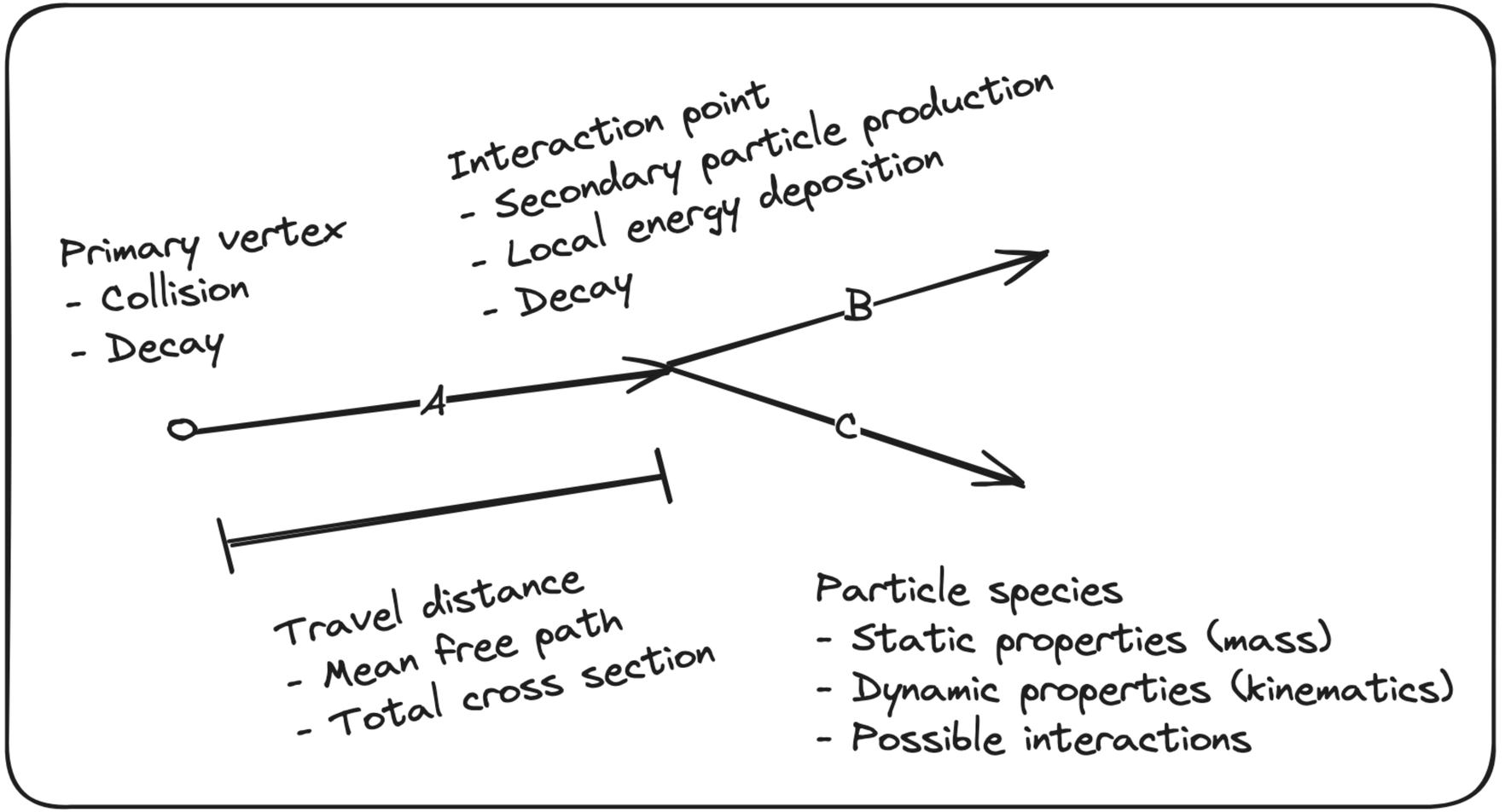

HEAVEN is
void of light, JON.

Event generation




Repeat
Output
Input
Integration
Events
Beams
Colliding particles or
decaying unstable
A
B
Process 1
do(A+B → A + B + C)
A
A
C
B
B
Event 1
A + B + C
Kinematics
Weight W1
Process 2
do(A+B → D + E)
A
D
B
E
Event 2
D + E
Kinematics
Weight W2
Accept/Reject
1: A + B + C
2: D + E
3: A + B + C
4: A + B + C
5: A + B + C
6: D + E
Integrated quantities
σ(A+B → D + E)
σ(A+B → A + B + C)
σ(A+B)
Derived quantities
dσ/dx, dN/dy

Vocabulary is hard: A rough sketch detector simulation



Geometry, material, and field configuration

Event generation

- 1: Specify the physics to simulate
  g g -> g g
- 2: Specify the beam configuration to simulate
  p + p @ 158 GeV
- 3: Run the Monte Carlo integration
- 4: Compute derived quantities of interest
  Cross sections, histograms, final states, etc
- 5: Go back to step 2

```
Pythia8::Pythia pythia{};
// Pick physics process to simulate
// gg -> ttbar
pythia.readString("Top:gg2ttbar = on");
// Set beams (proton proton at center of mass energy 8 TeV)
pythia.readString("Beams:idA = 2212");
pythia.readString("Beams:idB = 2212");
pythia.readString("Beams:eCM = 8000");
```

We can abstract that

A HEP MC simulation is a function that takes a physics specification as parameters and a beam configuration as the variables

Well, mostly

Common to all Pythia8 programs

```
Pythia8::Pythia pythia;
// Physics selection
pythia.readString("HardQCD:all = on");
pythia.readString("PhaseSpace:pTHatMin = 20.”);


// Beam configuration
pythia.readString("Beams:eCM = 8000.");
pythia.readString("Beams:idA = 2212");
pythia.readString("Beams:idB = 2212");
```

Physics settings

Beam configuration

Different between all Pythia8 programs

- What you do with your calculation depends on your needs but will generally involve

```
// Iteration over generated particles & selection
for (auto particle : pythia.event) {
    if (particle.isFinal() &&
someCondition(particle)) {
   }
}

// Accessing MC integration results
auto xs = pythia.info.sigmaGen();
```

Or maybe you do something

Unique and different