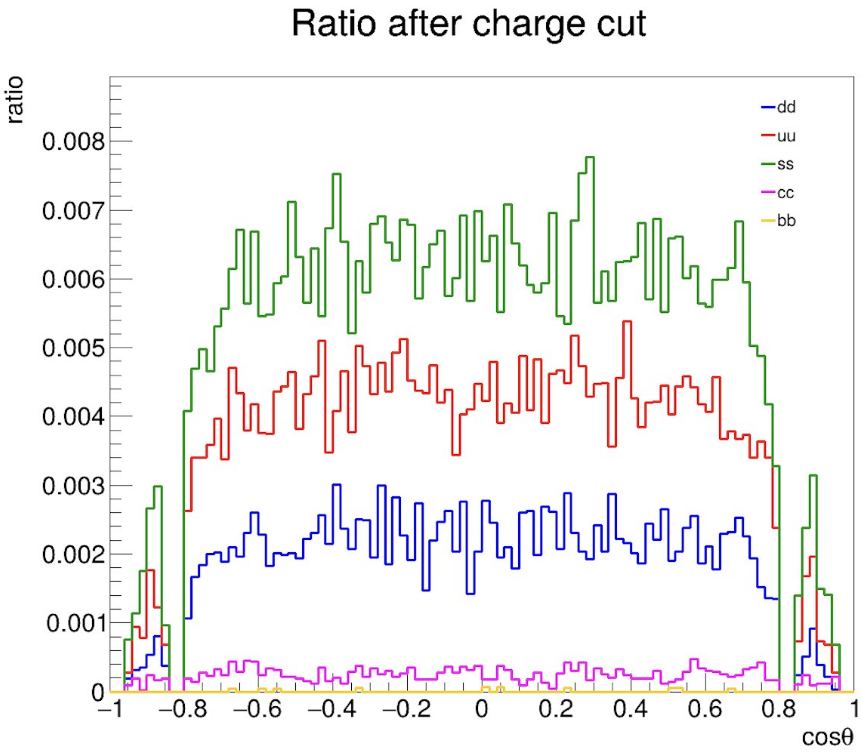
$e^- e^+ \rightarrow s\bar{s}$ at $\sqrt{s}$ = 250 GeV in future linear colliders

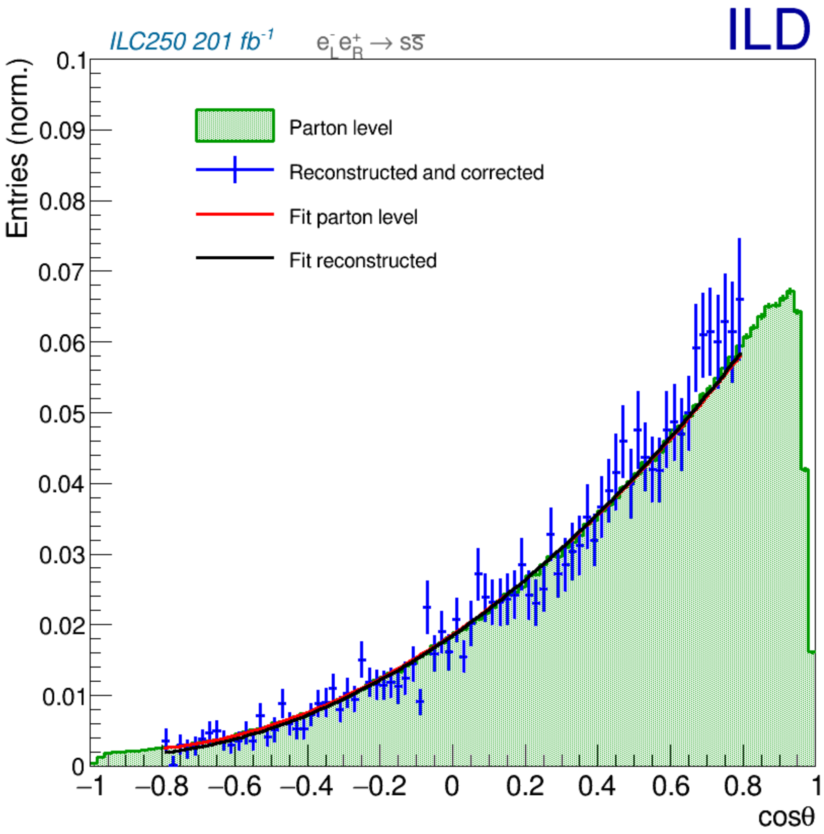
The previous updated in ssbarAnalysis (cut-based)

- Preselection of the s-quark/ud-quark signals (Modification of Y. Okugawa's analysis)
  - After the $q\bar{q}$ preselection

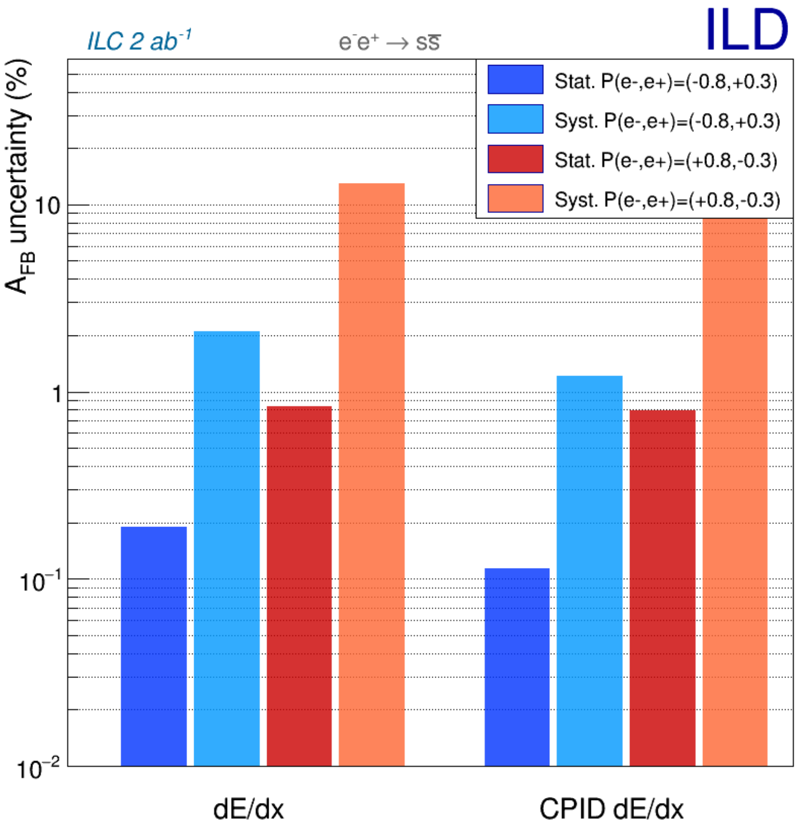
	#	Name	Quantity	Description
uds selection	1	b-tag	$btag < 0.3$	Reject events with b-like jets
	2	c-tag	$ctag < 0.65$	Reject events with c-like jets
	3	nvtx	$nvtx = 1$	Jets should have only PV as vertex
Cut-based s-tag (or ud-tag)	4	Leading momentum	$p_{LPFO} > 15\,\text{GeV}$	Leading momentum cut
	5	LPFO acollinearity	$\cos\theta_{LPFO_{1,2}} > 0.97$	LPFOs should be back-to-back
	6	Offset	$V_0 = \sqrt{d_0^2 + z_0^2} < 1\,\text{mm}$	Offset cut to reject Λ_0 contribution
	7a	dE/dx PID (π)	**New angular k-distance cuts**	$\pi^{\pm}$ identification
	7b	dE/dx PID (K)		$K^{\pm}$ identification
Migration correction	8	SPFO	Veto $p_{SPFO} > 10\,\text{GeV}$ and charge opposite to LPFO.	Attenuate the charge migration by rejecting oppositely charge LPFO competitor
	9	Charge	$Q_{LPFO1} \times Q_{LPFO2} < 0$ opposite charge.	Charge of LPFOs from both sides has

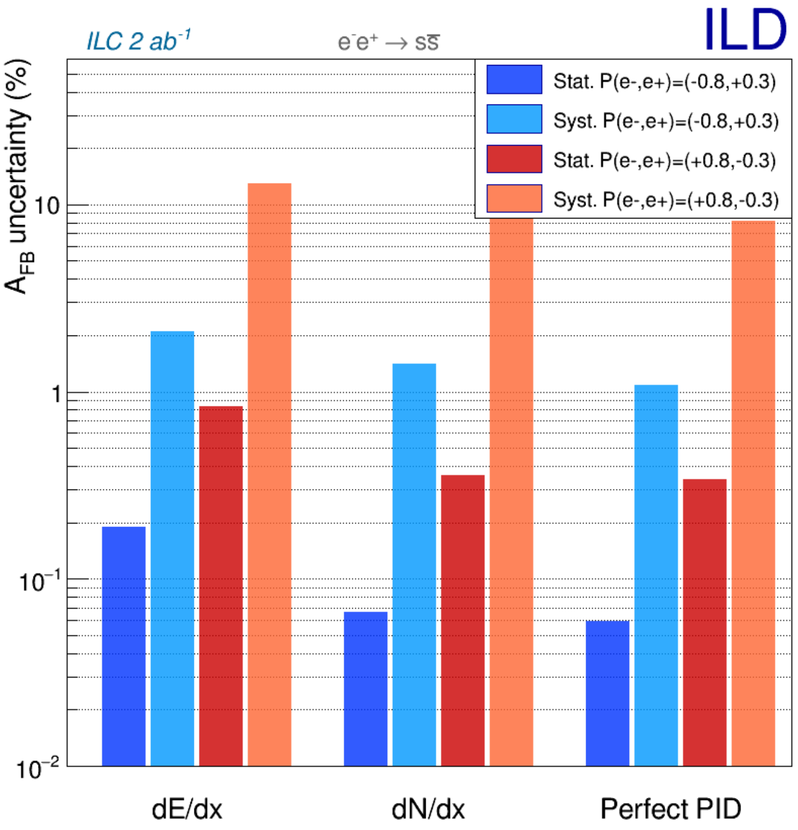
Result in previous analysis ($e^-_L e^+_R$)



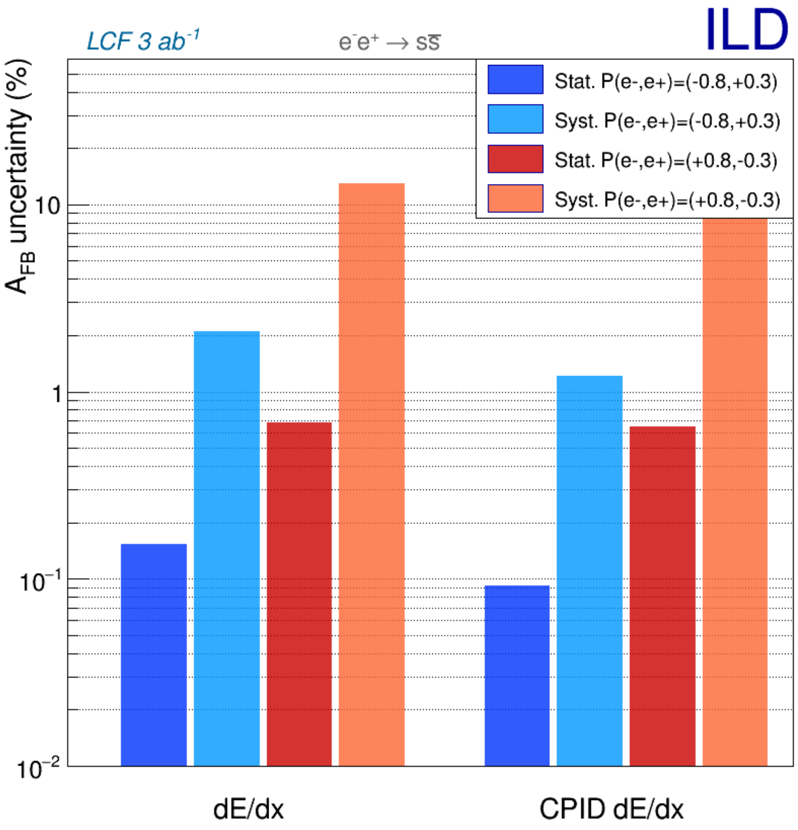


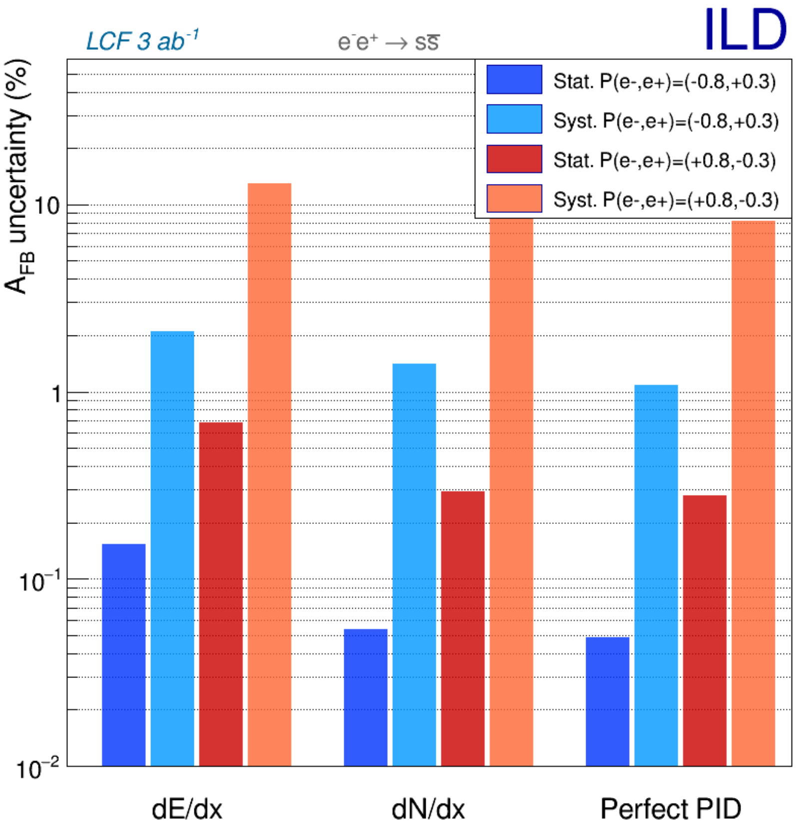
Preliminary results (ILC250) - Updated





Preliminary results (LCF250) - Updated

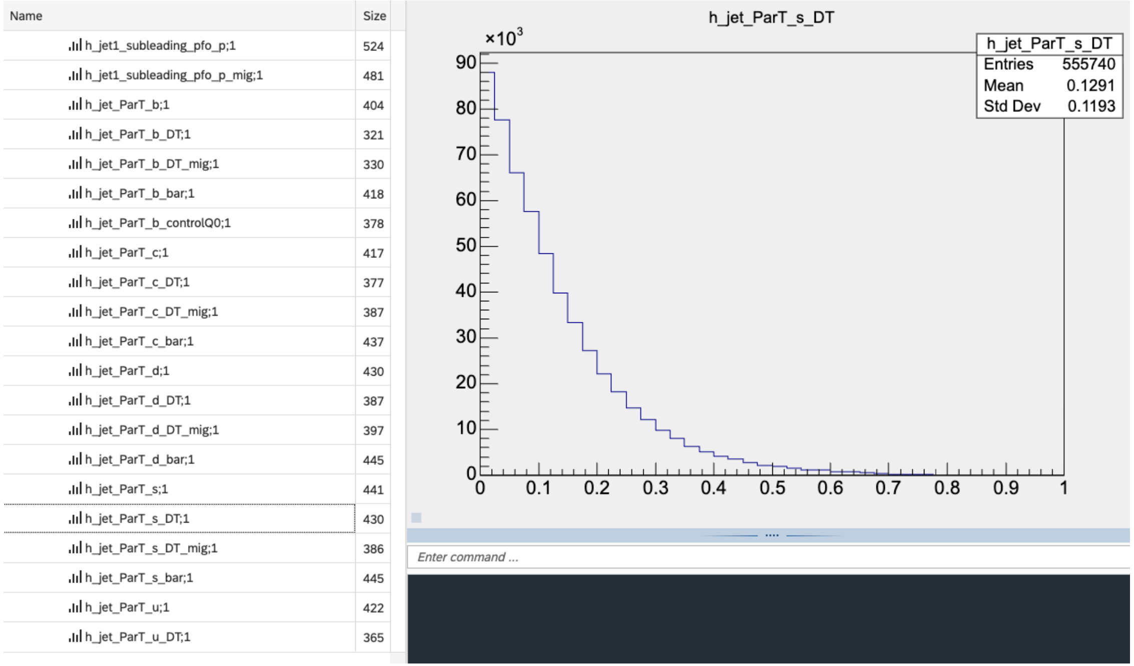




ParT DT score

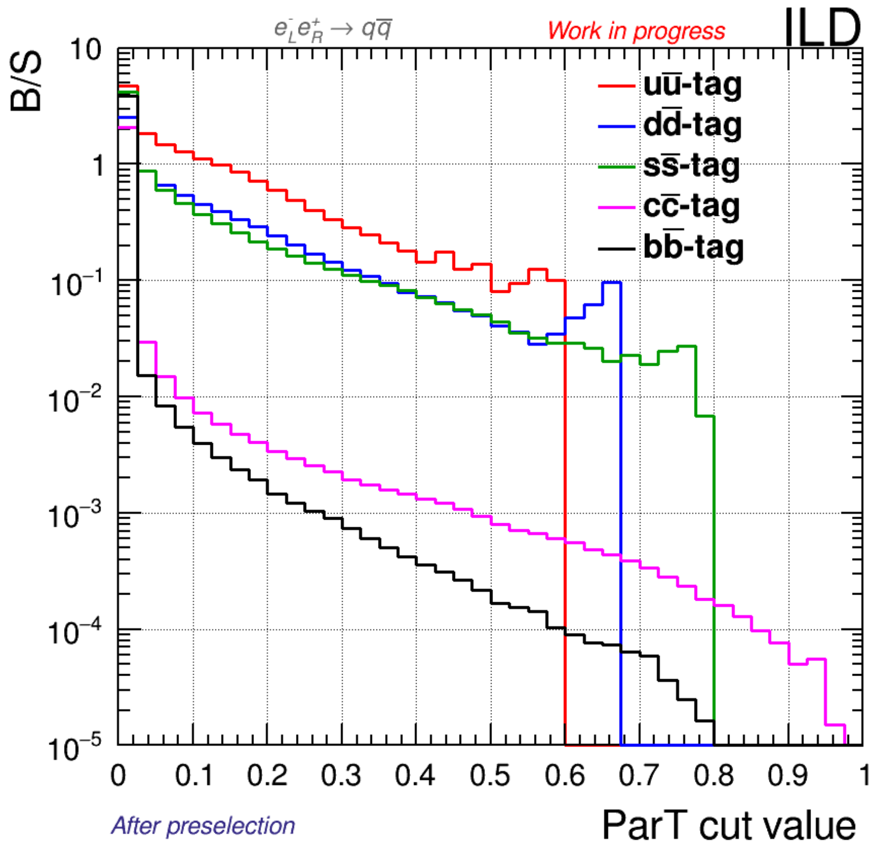
- Double-Tag score defined as: max(jN_ParT_q*jM_ParT_qbar)
  - If (j1_ParT_q>j1_ParT_qbar && (j2_ParT_q<j2_ParT_qbar) → j1_ParT_q*j2_ParT_qbar
  - Else If (j2_ParT_q>j2_ParT_qbar && (j1_ParT_q<j1_ParT_qbar) → j2_ParT_q*j1_ParT_qbar
- This way we cut off events with both jets having similar tag (qq or $\bar{q}\bar{q}$)

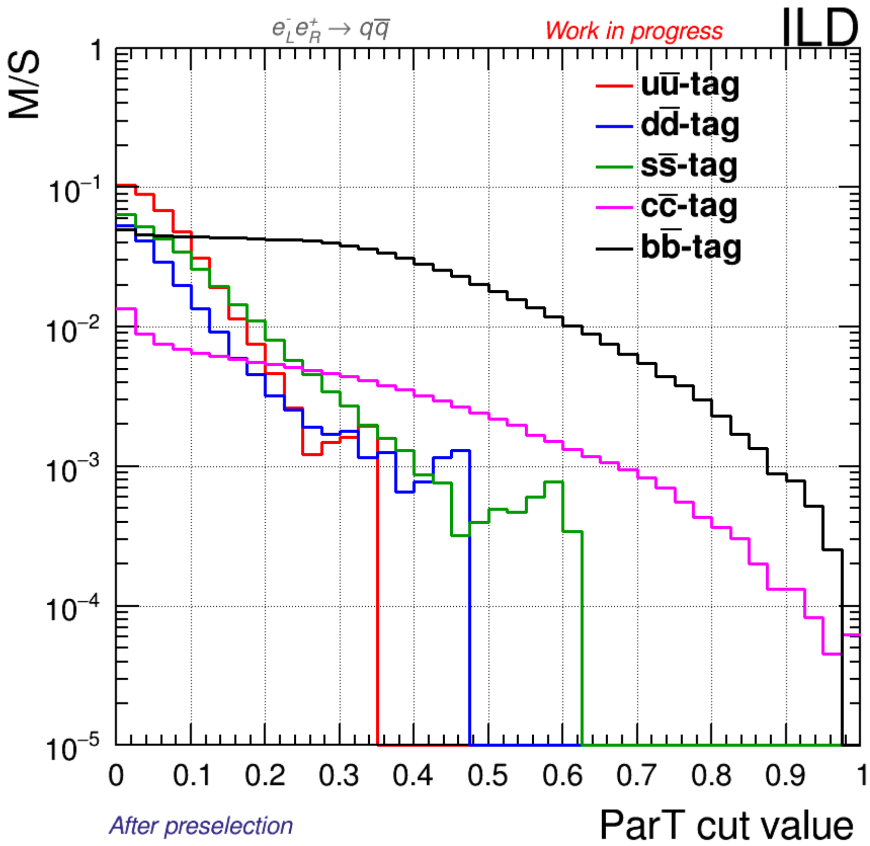
5 flavors qqbar tagging



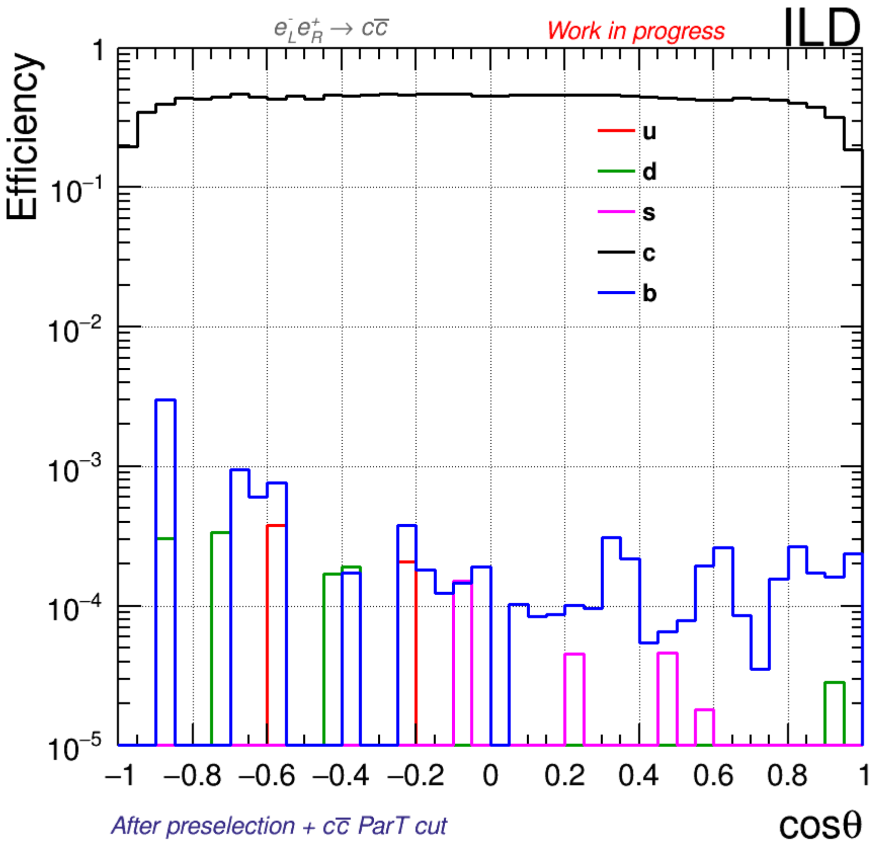
Background/signal and Migrations/signal ($e^-_L e^+_R$)

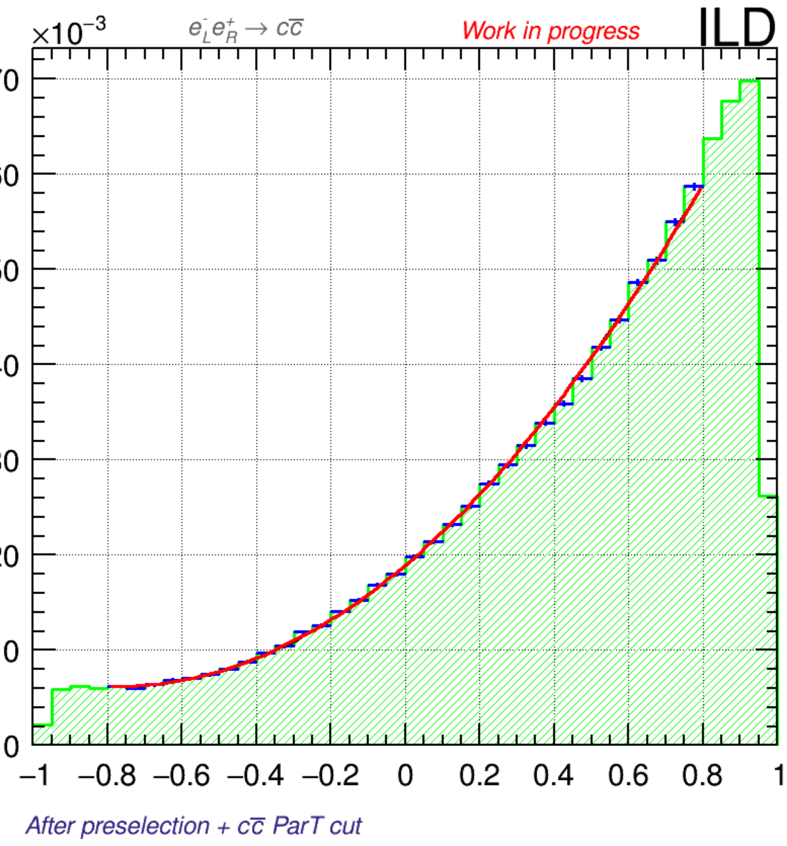
- I set different WP for each flavor



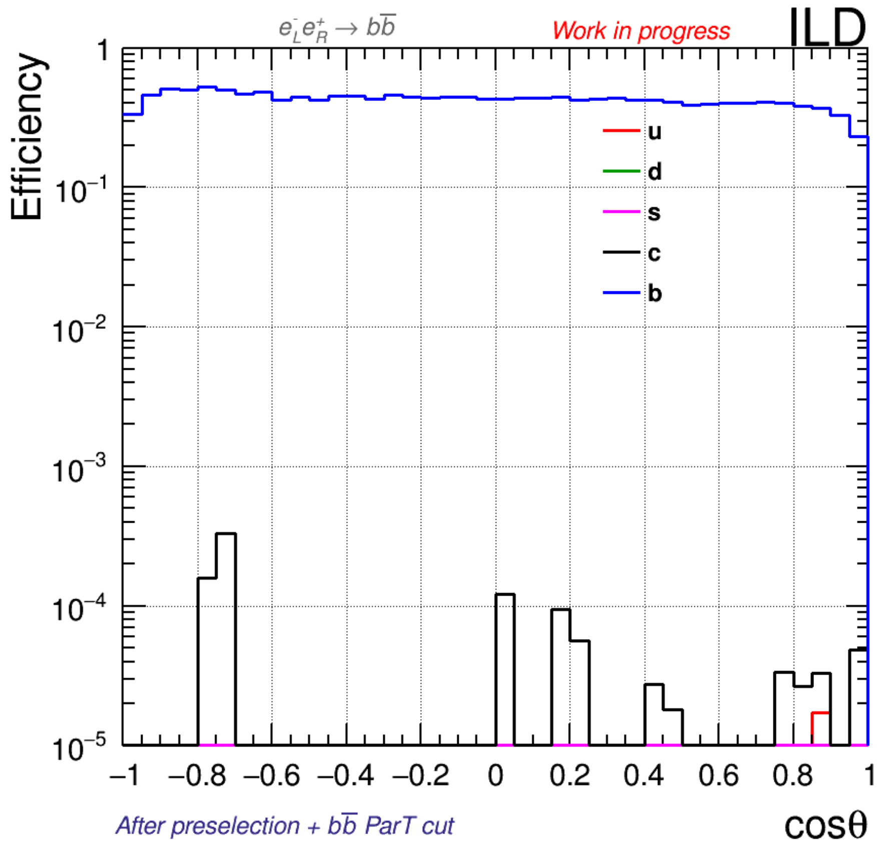


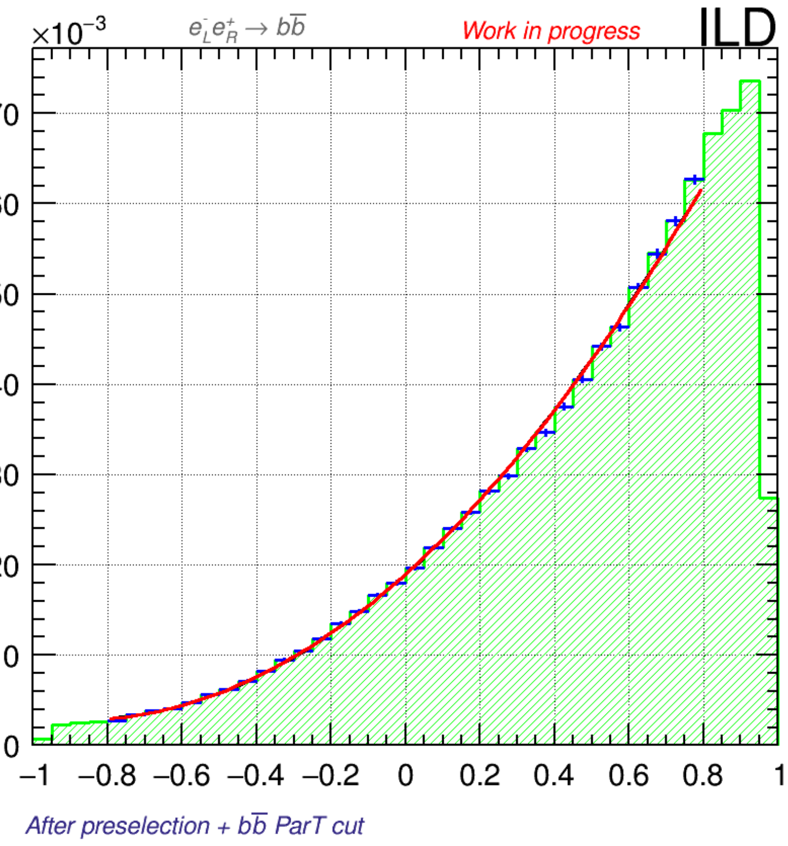
Selection and fit for $c\bar{c}$



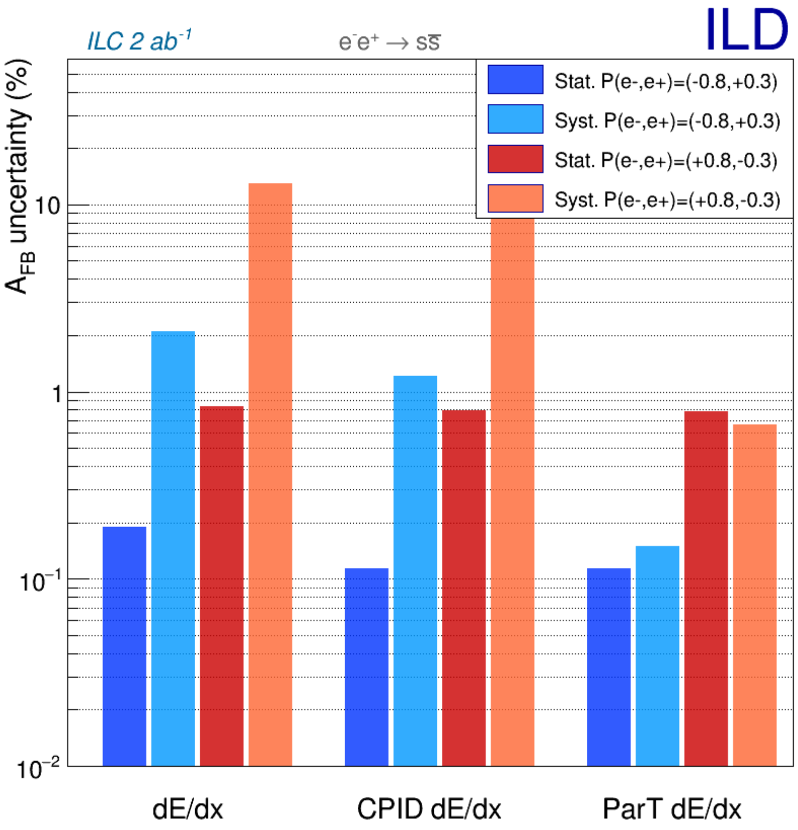


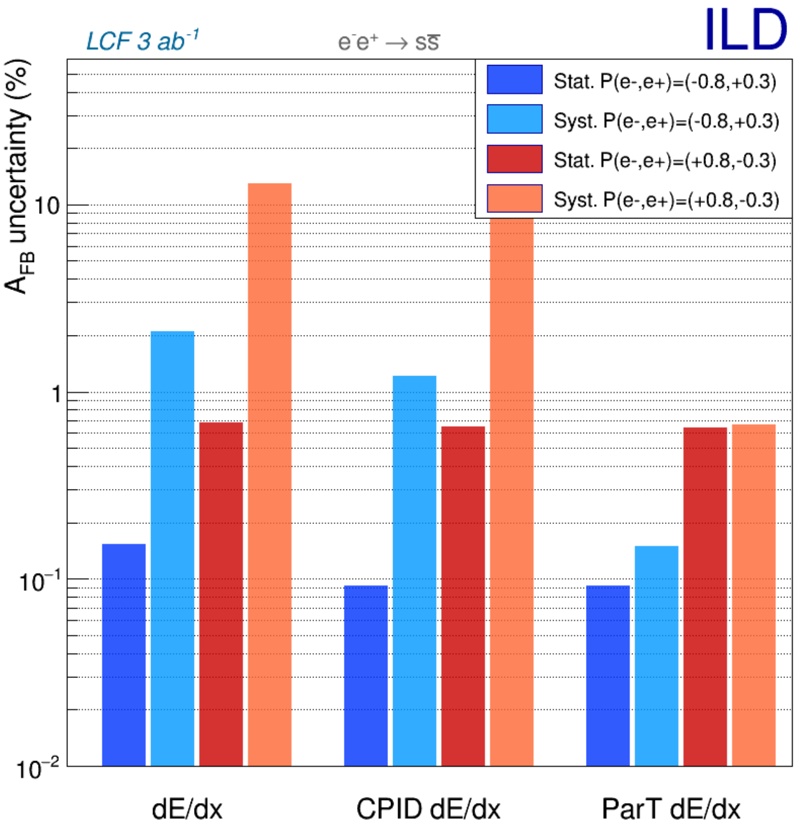
Selection and fit for $b\bar{b}$





Results (ILC250 & LCF250)

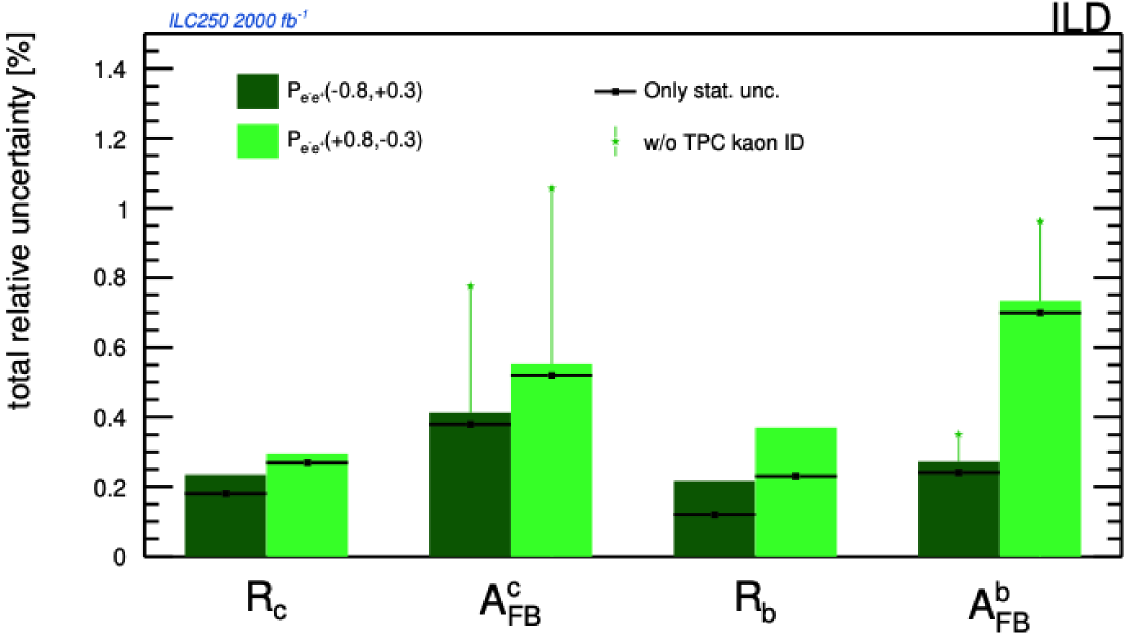




Preliminary uncertainties (b & c)

- Relative statistical uncertainties:
  - B-quark = 0.03 %
  - C-quark = 0.04 %
- Systematic uncertainties from background subtraction:
  - B-quark = < 0.01 % (~0.002 %)
  - C-quark = < 0.01 % (~0.002 %)
- These are about one order of magnitude smaller!

Previous results:



Future plans

- Calculations of more syst uncertainties due to:
  - Due to efficiency of signal selection (main source of syst in b and c!)
  - Migrations impact

- Proceedings of LCWS25: “Closure” of the cut-based analysis

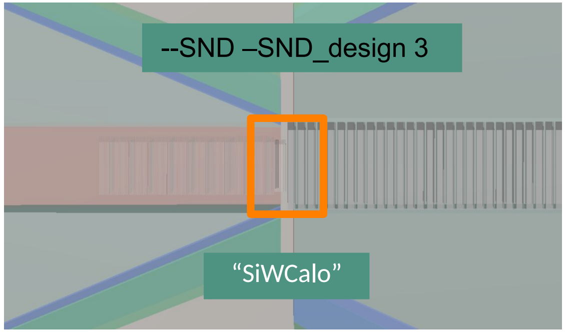
- Future paper with ParT analysis including b & c quarks + phenomenology (GHU)?

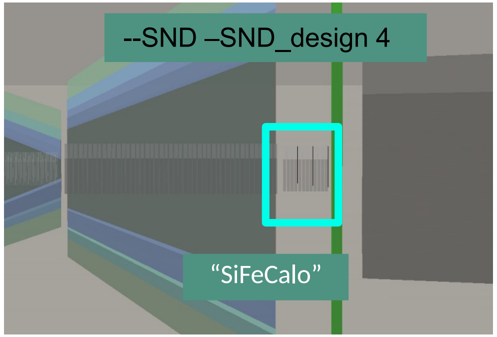
Plans for SND detector layout optimisation

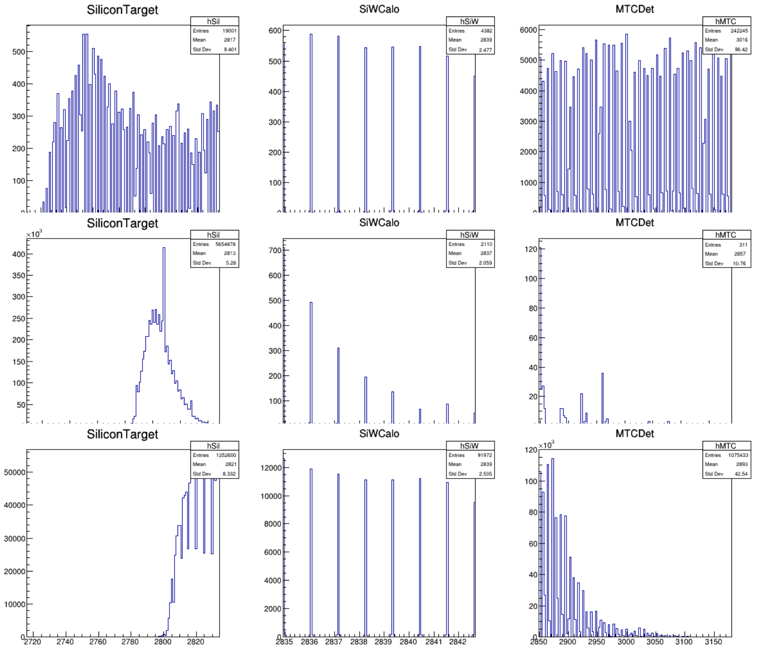
Jesús P. Márquez Hernández

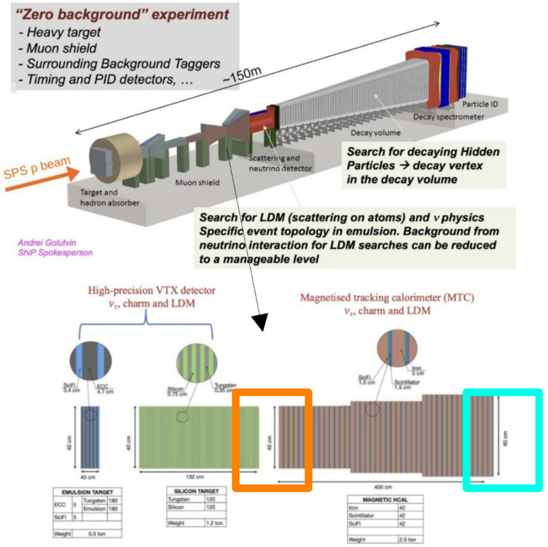
SHiP calorimeter simulation

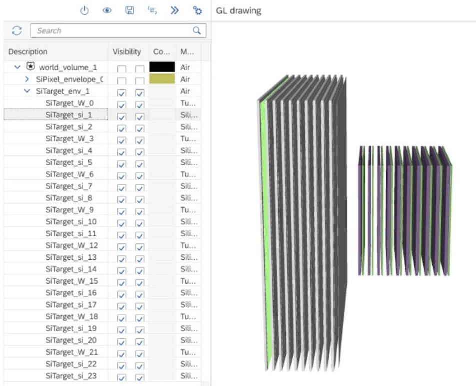
- ▶ Beam dump experiment being built at CERN
  - ▷ Neutrino Physics
  - ▷ Hidden particles (LDM, axions, etc.)
- ▶ Prototyping of the Scattering Neutrino Detector (SND)
- ▶ Inputs from key4hep for the new software
  - • dd4hep/ddsim for simulation
  - • Gaudi for reconstruction











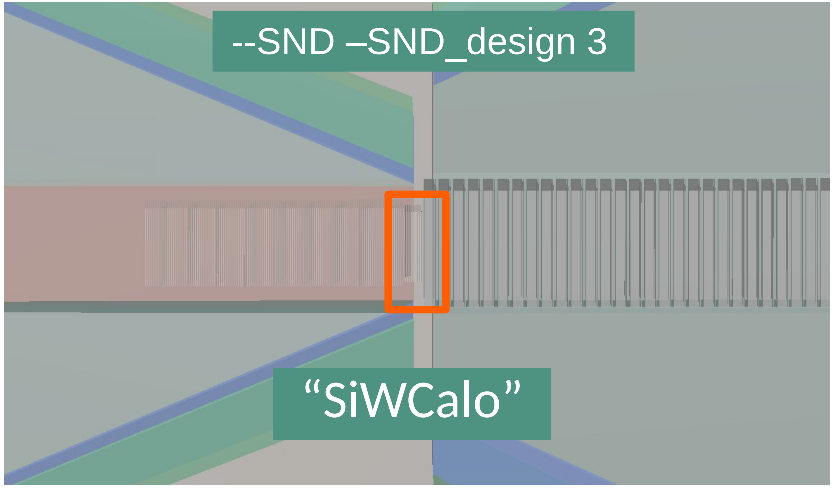
Adding our calorimeter (2 models)

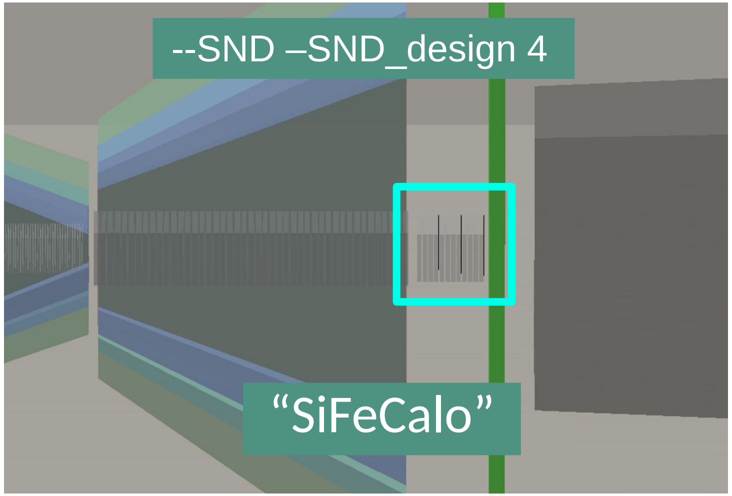
- In between SiW strip detector and MTC:
  - 5 simplified layers (5.6mm W + **0.65mm Si**)
  - Pixel size of ~**5.5mm**
  - 36x36 cm^2 layers (2x2 ASUs)

- After MTC:
  - 12 simplified layers (50mm Fe + **0.65mm Si**)
  - Pixel size of ~**5.5mm**
  - 54x54 cm^2 layers (3x3 ASUs)

This software is in legacy mode right now, the efforts are going into prototyping a new framework

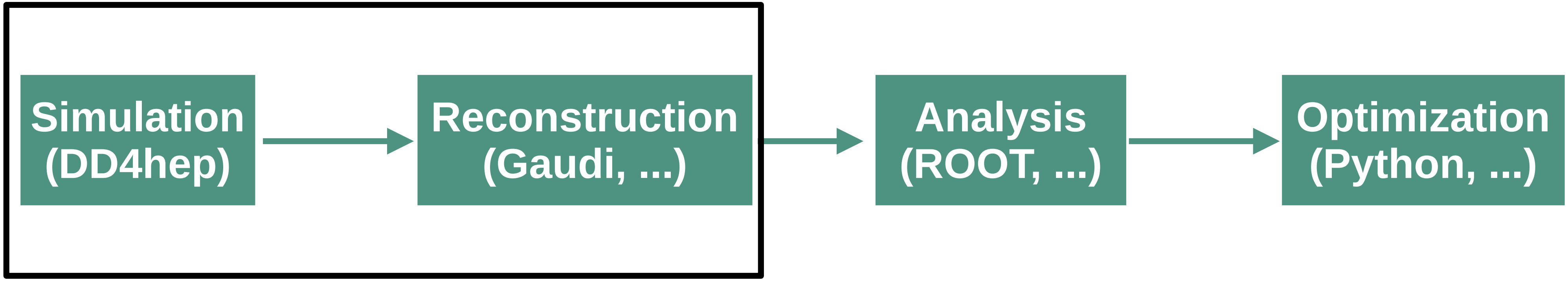
The ideas of two possible placement for a Si Pad detector are still the same





The key4ship Proof-Of-Concept (PoC)

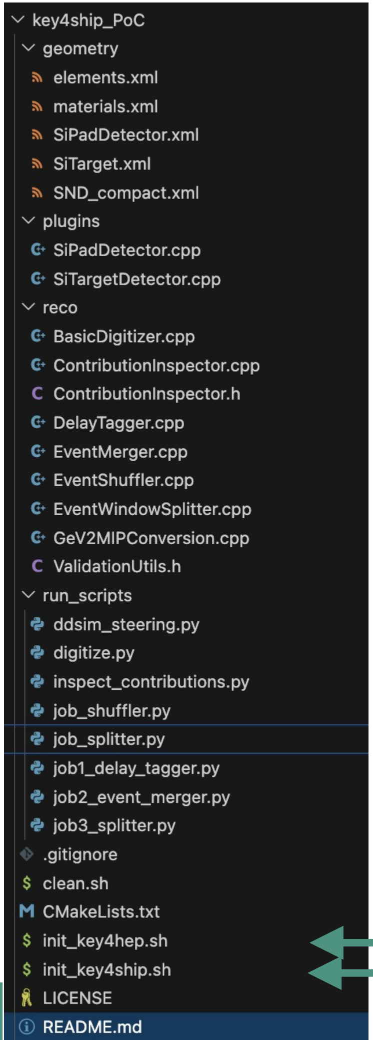
- Standard key4hep workflow:



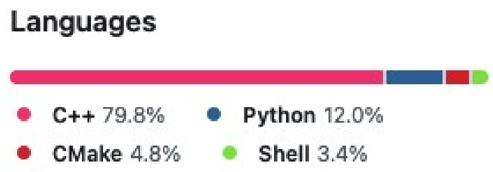
Being developed by me and covering a full pipelijne from simulation to reconstruction

https://github.com/marherje/key4ship_PoC

Proof-of-concept (key4ship)



- Geometry:
  - General definitions
  - Simulation steering files
- Plugins:
  - The C++ detector building plugins
- Reco:
  - The C++ Gaudi algorithms
- Run_scripts:
  - Python steering files for sim (ddsim) and reco (Gaudi)



Simply source key4hep setup.sh, **with a fixed date**
Same + configuring runtime paths for C++ plugins and Python modules

The PoC (II)

SND_sim proof-of-concept of full key4hep sim+reco chain: dd4hep + ddsim + gaudi

SND PoC with STarget and SiPixel — DD4hep v01-35 / key4hep 2026-02-01

Target stack (key4hep latest release in Feb 2026)

- DD4hep: v01-35
- ddsim: integrated in DD4hep v01-35
- Geant4: 11.x
- Gaudi: v4x series
- ROOT: 6.36+
- Platform: lxplus.cern.ch, AlmaLinux 9
- Source: /cvmfs/sw.hsf.org/key4hep/setup.sh -r 2026-02-01

Build and run

```
# 1. Source key4hep
source /cvmfs/sw.hsf.org/key4hep/setup.sh

# 2. Build plugin
mkdir build && cd build
cmake -DCMAKE_PREFIX_PATH="$CMAKE_PREFIX_PATH" \
      -DCMAKE_INSTALL_PREFIX=../install \
      -DCMAKE_BUILD_TYPE=RelWithDebInfo ..
make install -j4
cd ..

# 3. Expose plugins to DD4hep & Gaudi algorithms to python
export LD_LIBRARY_PATH=$PWD/install/lib64:$PWD/install/lib:$LD_LIBRARY_PATH
export PYTHONPATH=$PWD/install/lib64:$PWD/install/lib:$PWD/install/python:$PYTHONPATH

# 4. Run ddsim
cd run_scripts
ddsim --steeringFile ddsim_steering.py

# 5. Run algorithm/s
k4run digitize.py
```

This is as simple as it gets, if you have 40 subdetectors and 200 Gaudi algorithms the process is **the same**

Loading the project, once compiled, is just this:

```
#!/usr/bin/env bash

# To use only if the project is already compile and you want to play with the steering file, for example.
# If you want to compile the project, please follow the instructions in the README.md file.

source /cvmfs/sw.hsf.org/key4hep/setup.sh -r 2026-02-01
export LD_LIBRARY_PATH=$PWD/install/lib64:$PWD/install/lib:$LD_LIBRARY_PATH
export PYTHONPATH=$PWD/install/lib64:$PWD/install/lib:$PWD/install/python:$PYTHONPATH
```

Now I'll go part-by-part over the next slides

Simulation steering files

- Easy to use
- Highly flexible (Very desirable for prototyping)
  - This flexibility comes from how you prepare the plugin/s
- Compact steering file:
  - Constants for world and detector plugins
  - Constants for visualization (colors, transparency, etc.)
  - Readouts for all the subdetectors
  - Magnetic fields
  - Subdetectors
    - Can be defined here directly
    - Can be imported (better option in the long run)

My subdetectors' steerings

SiPad defined on his own file and build layer-by-layer

```
<!-- Layers 2-4: W(1.40x) + W(4.20x) + CF + Cu + glue + Si(650um) + glue + Cu + PCB + chip + gap -->
<layer repeat="10" vis="EcalVis">
  <slice material="Air"              thickness="1.40*Ecal_WThickness"     vis="AirVis"/>
  <slice material="TungstenDens1910" thickness="4.20*Ecal_WThickness"     vis="TungstenVis"/>
  <slice material="CarbonFiber"      thickness="Ecal_CFThickness"         vis="CFVis"/>
  <slice material="Copper"           thickness="Ecal_KaptonThickness"     vis="CuVis"/>
  <slice material="Air"              thickness="Ecal_GlueThickness_kap"   vis="AirVis"/>
  <slice material="Silicon"          thickness="Ecal_WaferThickness650"   vis="SiVis" sensitive="yes"/>
  <slice material="Air"              thickness="Ecal_GlueThickness_pcb"   vis="AirVis"/>
  <slice material="Copper"           thickness="Ecal_KaptonThickness"     vis="CuVis"/>
  <slice material="PCB"              thickness="Ecal_PcbThickness"        vis="PCBVis"/>
  <slice material="Air"              thickness="Ecal_ChipThickness"       vis="AirVis"/>
  <slice material="Air"              thickness="Ecal_w_slab_gap650"       vis="AirVis"/>
</layer>
```

SiTarget defined on his own file and build layer-by-layer

```
<layer repeat="SiTarget_NLayers">
  <!-- Tungsten absorber -->
  <slice material="TungstenDens1910" thickness="SiTarget_WThickness"
         sensitive="false" vis="TungstenVis"/>
  <!-- Air gap between absorber and first Si plane (module offset) -->
  <slice material="Air" thickness="SiTarget_module_offset"
         sensitive="false"/>
  <!-- X plane: measures X coordinate, no rotation -->
  <slice material="Silicon" thickness="SiTarget_sensor_thickness"
         sensitive="true" plane="0" vis="SiVis"/>
  <!-- Air gap between the two Si planes -->
  <slice material="Air" thickness="SiTarget_XY_plane_gap"
         sensitive="false"/>
  <!-- Y plane: measures Y coordinate, rotated 90 degrees around Z by the plugin -->
  <slice material="Silicon" thickness="SiTarget_sensor_thickness"
         sensitive="true" plane="1" vis="SiVis"/>
  <!-- Air gap after second Si plane to next absorber -->
  <slice material="Air" thickness="SiTarget_layer_gap"
         sensitive="false"/>
</layer>
```

The plugins

- This IS the hard part, the only hard part!
  - Not harder than the current FairShip geometry at all

- You defined here how to construct your detector:
  - Input general parameters: N layers, slices of materials, type of detector
  - Sensitive part/s description/s: Type (pixels, strips), information for encoding (system, layer, slice, plane)

- I have 2 plugins
  - SiPixelDetector: Very flexible, recycled from our previous SiWECAL TB simulations
    - Built detector *layer-by-layer*, allowing customization per slice (different W, Si, glue, PCB, whatever). All layers with similar digitization (pixels)
  - SiTargetDetector: Updated to a similar layer-by-layer and slice-by-slice build

Gaudi algorithms

```cpp
  StatusCode execute(const EventContext&) const override {
    const auto* input  = m_inputHandle->get();
    auto*       output = m_outputHandle->createAndPut();

    for (const auto& hit : *input) {
      auto nh = output->create();
      nh.setCellID(hit.getCellID());
      nh.setEnergy(static_cast<float>(hit.getEnergy() * (1/m_MIPValue)));
      nh.setPosition(hit.getPosition());
    }

    info() << "GeV2MIPConversion: " << input->size() << " hits processed" << endmsg;
    return StatusCode::SUCCESS;
  }

  StatusCode finalize() override {
    m_inputHandle.reset();
    m_outputHandle.reset();
    return Gaudi::Algorithm::finalize();
  }

private:
  // Python-configurable properties
  Gaudi::Property<std::string> m_inputName{
      this, "InputCollection", "SiTargetHits", "Input SimCalorimeterHit collection"};
  Gaudi::Property<std::string> m_outputName{
      this, "OutputCollection", "SiTargetHitsMIP", "Output SimCalorimeterHit collection"};
  Gaudi::Property<double> m_MIPValue{
      this, "MIPValue", 0.000009, "MIP value for conversion (in GeV)"}; // Default is 9 keV, for 0.3mm silicon

  // DataHandles constructed in initialize() from the properties above
  mutable std::unique_ptr<k4FWCore::DataHandle<edm4hep::SimCalorimeterHitCollection>> m_inputHandle;
  mutable std::unique_ptr<k4FWCore::DataHandle<edm4hep::SimCalorimeterHitCollection>> m_outputHandle;
```

- Every algorithm inherits from Gaudi
  - You always fit the template
- Building your own functions is not more difficult than ROOT
- Everything is read/written from/into EDM4hep
  - You always stick to the format

It protect us from ourselves (& each other)

run_scripts: Simulation steering file

```python
import os
from DDSim.DD4hepSimulation import DD4hepSimulation
from g4units import mm, GeV, MeV

SIM = DD4hepSimulation()

SIM.runType          = "batch"
SIM.numberOfEvents   = 1000
SIM.skipNEvents      = 0

SIM.compactFile = "../geometry/SND_compact.xml"
SIM.outputFile      = "output.edm4hep.root"
print("COMPACT FILE =", SIM.compactFile)

SIM.enableGun        = True
SIM.gun.particle     = "mu-"
SIM.gun.energy       = 50 * GeV
SIM.gun.position     = (0, 0, -500 * mm)
SIM.gun.direction    = (0, 0, 1)

SIM.physicsList      = "QGSP_BERT"
```

- Very simple python script
- Easy for generalization/template
- This is a simple PG, example for PoC
  - In principle ddsim can also interface Genie with Geant4 into the dd4hep geometry

run_script: Gaudi steering file

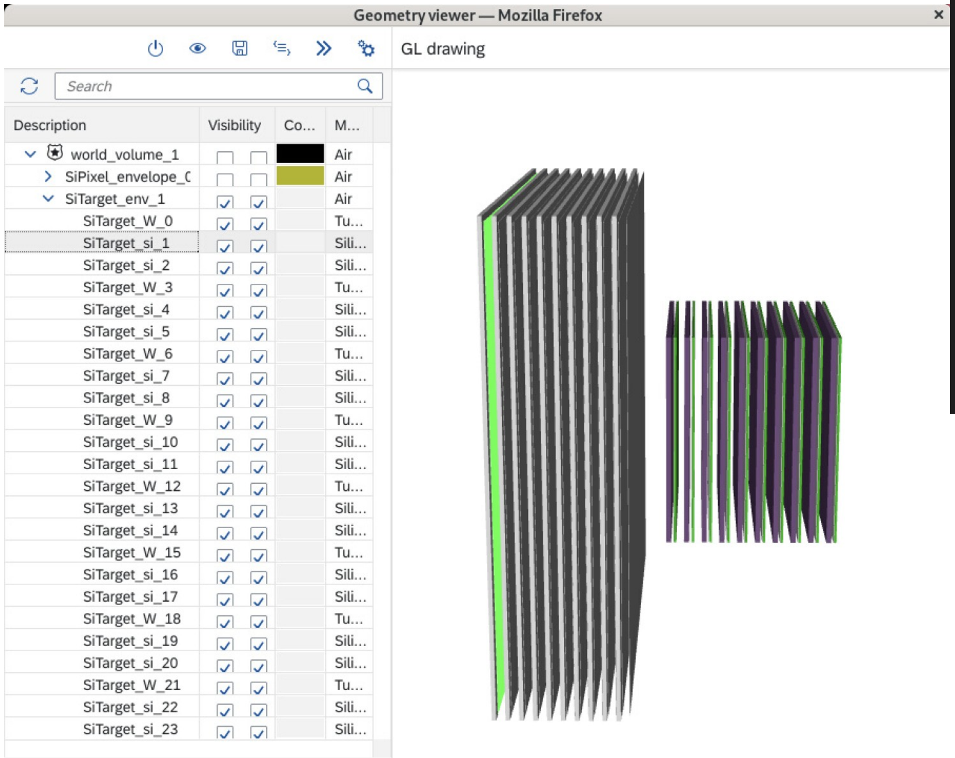
```python
from k4FWCore import ApplicationMgr, IOSvc
from Configurables import GeV2MIPConversion, BasicDigitizer
# Initializer
iosvc = IOSvc()
iosvc.Input = "output.edm4hep.root"
iosvc.Output = "snd_digi.edm4hep.root"

# Loading the digitization algorithms
mip_1 = GeV2MIPConversion("GeV2MIP_SiTarget")
mip_1.InputCollection  = "SiTargetHits"
mip_1.OutputCollection = "SiTargetHitsMIP"
mip_1.MIPValue = 0.00009

mip_2 = GeV2MIPConversion("GeV2MIP_SiPixel")
mip_2.InputCollection  = "SiPixelHits"
mip_2.OutputCollection = "SiPixelHitsMIP"
mip_2.MIPValue = 0.0002

dig_1 = BasicDigitizer("BasicDigitizer_SiTarget")
dig_1.InputCollection  = "SiTargetHitsMIP"
dig_1.OutputCollection = "SiTargetHitsDigi"
dig_1.Threshold = 0.5

dig_2 = BasicDigitizer("BasicDigitizer_SiPixel")
dig_2.InputCollection  = "SiPixelHitsMIP"
dig_2.OutputCollection = "SiPixelHitsDigi"
dig_2.Threshold = 0.5

# Application manager
ApplicationMgr(
    EvtSel  = "NONE",
    EvtMax  = -1,
    TopAlg  = [mip_1, mip_2, dig_1, dig_2],
    ExtSvc  = [iosvc]
)     Jesus Pedro Marquez Hernandez, 1 hour ago • First com
```

- I made 2 Gaudi algorithms for the PoC:
  - GeV to MIP conversion using a constant value
  - Basic “digitization”, just a MIP cut
- Again, very simple python script & easy for generalization/template work
- The workflow is trivial to follow
  - As well as adding future algorithms in the chain

AppMgr: Controls what events to process, what algorithms to run, and in what order.
What happens if you do [mip_1, dig_1, mip_2, dig_2]?
And if you do [mip_1, dig_2, mip_2, dig_1]?

Testing geoDisplay and materialScan (in my VM)

- I showed some dd4hep in-house tools

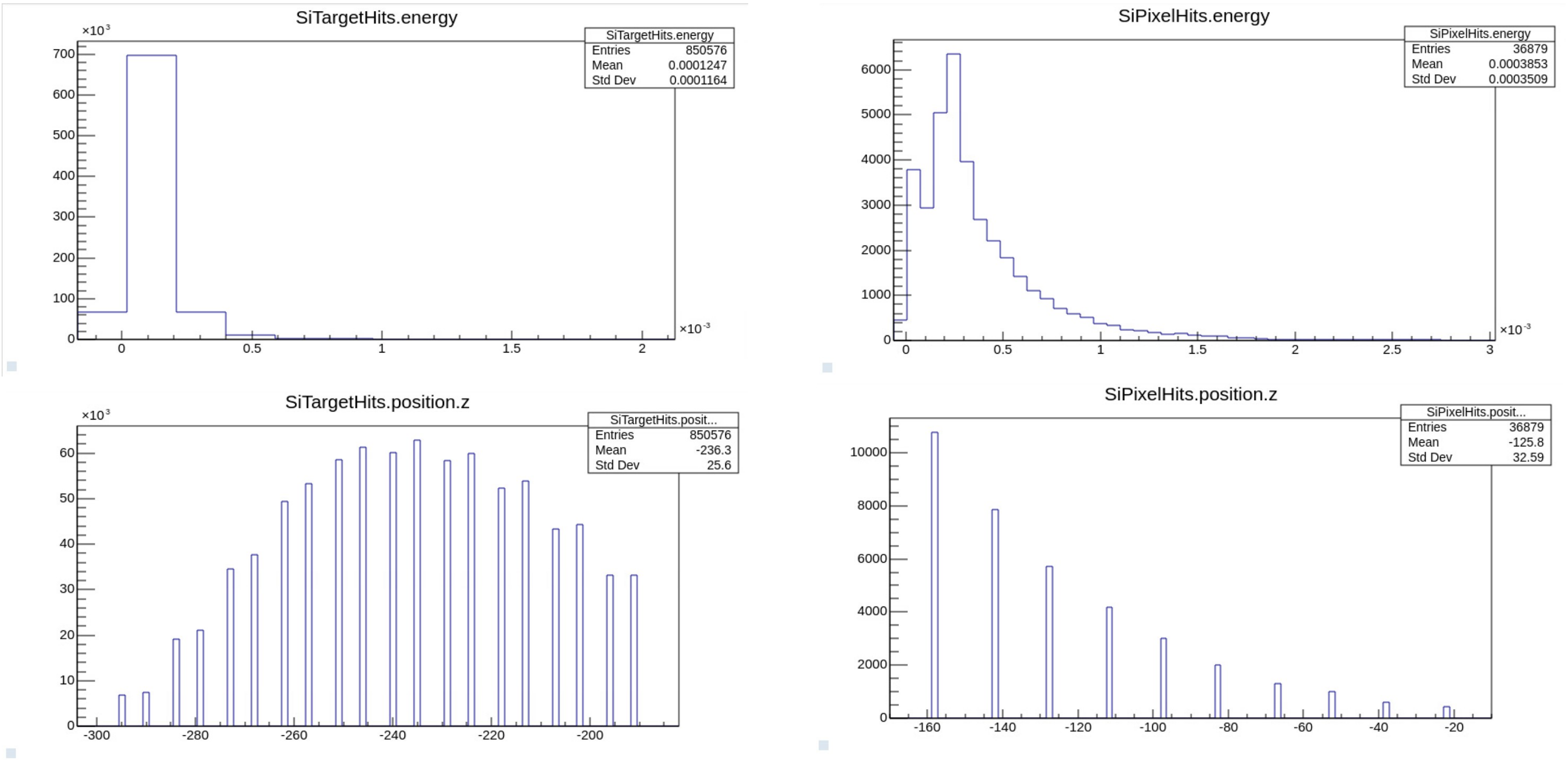


This geometry is fixed now, didn't have time to change the plot

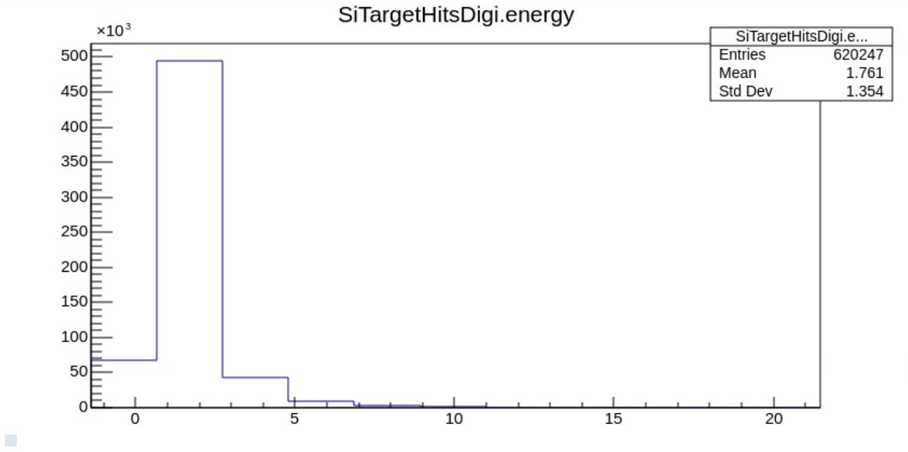
\+ Material scan between: x_0 = (0.00, 0.00,-100.00) [cm] and x_1 = (0.00, 0.00, 100.00) [cm] :

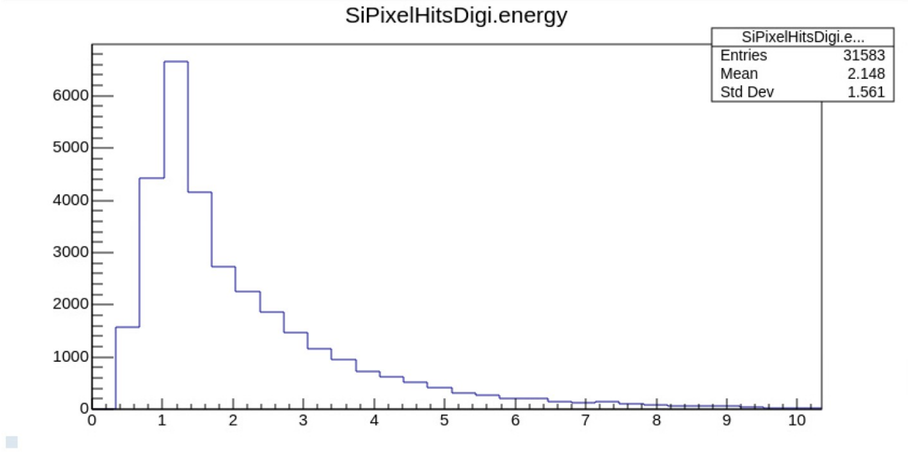
Num. Layer	Material Name	Atomic Number/Z	Mass/A [g/mole]	Density [g/cm3]	Radiation Length [cm]	Interaction Length [cm]	Thickness [cm]	Path Length [cm]	Integrated X0 [cm]	Integrated Lambda [cm]	Material Endpoint/Startpoint (cm, cm, cm)
(start)	Air	7	14.801	0.0012	30392.1308	71716.4382	0.000	0.00	0.000000	0.000000	(0.00, 0.00,-100.00)
1	Air -> Air	7	14.801	0.0012	30392.1308	71716.4382	69.900	69.90	0.002300	0.000975	(0.00, 0.00, -30.10)
2	Air -> TungstenDens1910	74	183.842	19.1000	0.3541	10.4515	0.100	70.00	0.002303	0.000976	(0.00, 0.00, -30.00)
3	TungstenDens1910 -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.350	70.35	0.990758	0.034464	(0.00, 0.00, -29.65)
4	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.100	70.45	0.990761	0.034465	(0.00, 0.00, -29.55)
5	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	70.48	0.993964	0.035121	(0.00, 0.00, -29.52)
6	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.550	71.03	0.993982	0.035129	(0.00, 0.00, -28.97)
7	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	71.06	0.997185	0.035785	(0.00, 0.00, -28.94)
8	Air -> TungstenDens1910	74	183.842	19.1000	0.3541	10.4515	0.040	71.10	0.997187	0.035785	(0.00, 0.00, -28.90)
9	TungstenDens1910 -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.350	71.45	1.985641	0.069273	(0.00, 0.00, -28.55)
10	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.100	71.55	1.985644	0.069274	(0.00, 0.00, -28.45)
11	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	71.58	1.988847	0.069930	(0.00, 0.00, -28.42)
12	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.550	72.13	1.988865	0.069938	(0.00, 0.00, -27.87)
13	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	72.16	1.992068	0.070594	(0.00, 0.00, -27.84)
14	Air -> TungstenDens1910	74	183.842	19.1000	0.3541	10.4515	0.040	72.20	1.992070	0.070594	(0.00, 0.00, -27.80)
15	TungstenDens1910 -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.350	72.55	2.980524	0.104082	(0.00, 0.00, -27.45)
16	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.100	72.65	2.980528	0.104084	(0.00, 0.00, -27.35)
17	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	72.68	2.983731	0.104739	(0.00, 0.00, -27.32)
18	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.550	73.23	2.983749	0.104747	(0.00, 0.00, -26.77)
19	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	73.26	2.986952	0.105403	(0.00, 0.00, -26.74)
20	Air -> TungstenDens1910	74	183.842	19.1000	0.3541	10.4515	0.040	73.30	2.986953	0.105403	(0.00, 0.00, -26.70)
21	TungstenDens1910 -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.350	73.65	3.975408	0.138891	(0.00, 0.00, -26.35)
22	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.100	73.75	3.975411	0.138893	(0.00, 0.00, -26.25)
51	TungstenDens1910 -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.350	79.15	8.949824	0.312936	(0.00, 0.00, -20.85)
52	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.100	79.25	8.949827	0.312938	(0.00, 0.00, -20.75)
53	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	79.28	8.953030	0.313593	(0.00, 0.00, -20.72)
54	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.550	79.83	8.953048	0.313601	(0.00, 0.00, -20.17)
55	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	79.86	8.956251	0.314257	(0.00, 0.00, -20.14)
56	Air -> TungstenDens1910	74	183.842	19.1000	0.3541	10.4515	0.040	79.90	8.956253	0.314257	(0.00, 0.00, -20.10)
57	TungstenDens1910 -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.350	80.25	9.944707	0.347745	(0.00, 0.00, -19.75)
58	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.100	80.35	9.944711	0.347747	(0.00, 0.00, -19.65)
59	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	80.38	9.947914	0.348402	(0.00, 0.00, -19.62)
60	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.550	80.93	9.947932	0.348410	(0.00, 0.00, -19.07)
61	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.030	80.96	9.951135	0.349066	(0.00, 0.00, -19.04)
62	Air -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.100	81.06	9.951138	0.349067	(0.00, 0.00, -18.94)
63	Air -> Air	7	14.801	0.0012	30392.1308	71716.4382	2.440	83.50	9.951218	0.349101	(0.00, 0.00, -16.50)
64	Air -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.000	83.50	9.951218	0.349101	(0.00, 0.00, -16.50)
65	Air -> TungstenDens1910	74	183.842	19.1000	0.3541	10.4515	0.140	83.64	9.951223	0.349103	(0.00, 0.00, -16.36)
66	TungstenDens1910 -> CarbonFiber	6	11.956	1.4667	28.6083	51.8139	0.420	84.06	11.137368	0.389289	(0.00, 0.00, -15.94)
67	CarbonFiber -> Copper	29	63.546	8.9600	1.4356	15.6778	0.150	84.21	11.142612	0.392184	(0.00, 0.00, -15.79)
68	Copper -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.010	84.22	11.149577	0.392821	(0.00, 0.00, -15.78)
69	Air -> Silicon	14	28.085	2.3300	9.3661	45.7531	0.010	84.23	11.149578	0.392822	(0.00, 0.00, -15.77)
70	Silicon -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.065	84.30	11.156518	0.394242	(0.00, 0.00, -15.70)
71	Air -> Copper	29	63.546	8.9600	1.4356	15.6778	0.010	84.31	11.156518	0.394242	(0.00, 0.00, -15.69)
72	Copper -> PCB	10	20.338	1.7000	17.5049	49.2238	0.010	84.32	11.163484	0.394880	(0.00, 0.00, -15.68)
73	PCB -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.170	84.49	11.173195	0.398334	(0.00, 0.00, -15.51)
74	Air -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.120	84.61	11.173199	0.398336	(0.00, 0.00, -15.39)
75	Air -> Air	7	14.801	0.0012	30392.1308	71716.4382	0.395	85.00	11.173212	0.398341	(0.00, 0.00, -15.00)
76	Air -> TungstenDens1910	74	183.842	19.1000	0.3541	10.4515	0.140	85.14	11.173217	0.398343	(0.00, 0.00, -14.86)

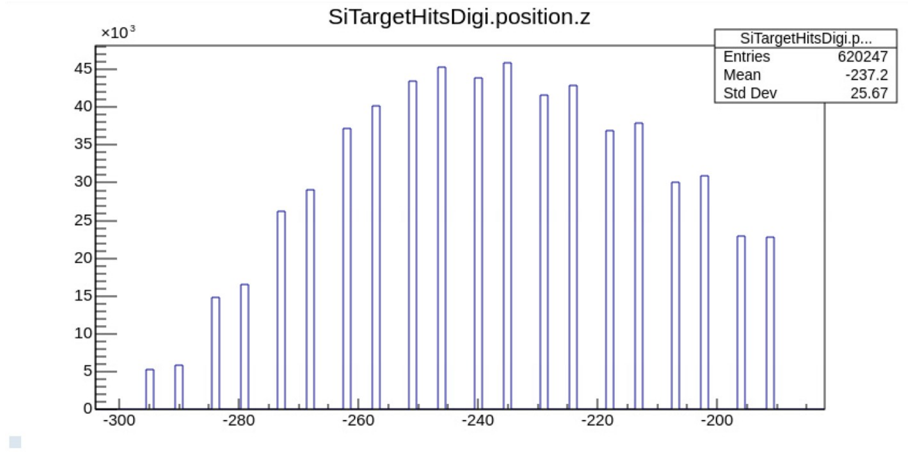
Examples of PoC (5 GeV e$^-$ shower - Sim)

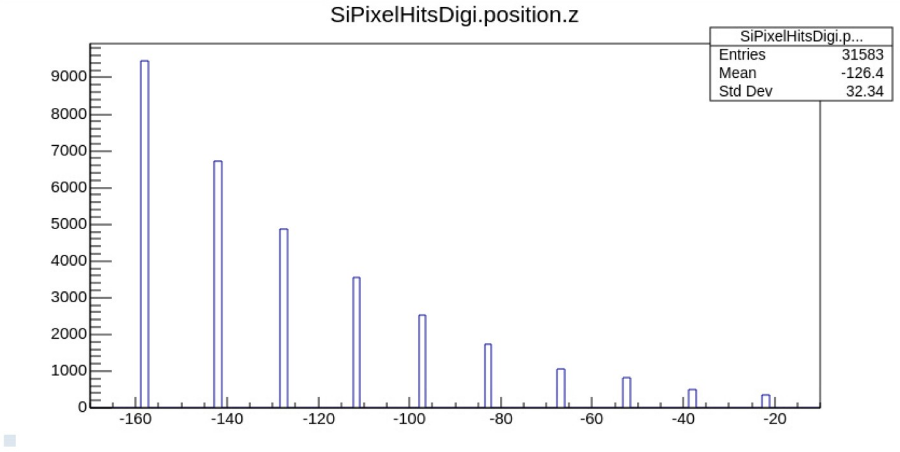


Examples of PoC (5 GeV e$^-$ shower - Digi)









Recent changes

- New Gaudi algorithms to adapt the events to SHiP triggerless operations
  - ContributionInspector: For debugging. Produce custom print output about hit info
  - DelayTagger: Rewrite hit contributions timestamps
  - EventMerger: Get all hits contributions from all events and join them into a single super-event. Can accommodate different sources at once (signal and backgrounds, etc.). Can get the events with different delays between them
  - EventShuffler: Get a super event and split it in time windows

- After running all this chain we have time window events like one could expect in a continuum triggerless experiment
  - MIP Calibration, MIP cut/digitization can be done after this point

- Considering redoing this chain but with a custom edm, will fit best the needs of SHiP
  - Doing it within edm4hep wasn’t as nice and easy as I expected…

My goals for the SHiP collaboration meeting

- Full pipeline (almost ready) but with a custom edm
  - I might keep the current pipeline and the custom edm one and show both!
- Full SND including a emulsion target and the magnetized tracker calorimeter (MTC)
- (Optional) Magnet field in MTC
- (Optional) Neutrino event with Geanie
- Open to ideas, let me know ;)

Context: Other's people contributions

- Matei has done some geometries+G4 simulation in GeoModel. Feedback:
  - Geometry building is very easy
  - G4 binding is fine
  - Simulation output and edm4 is not well defined yet
- Eduard did MTC in GeoModel
  - No sim or anything else
- Gilhermo said he prefers GeoModel for geometry because it knows it
  - But not a strong opinion or work presented

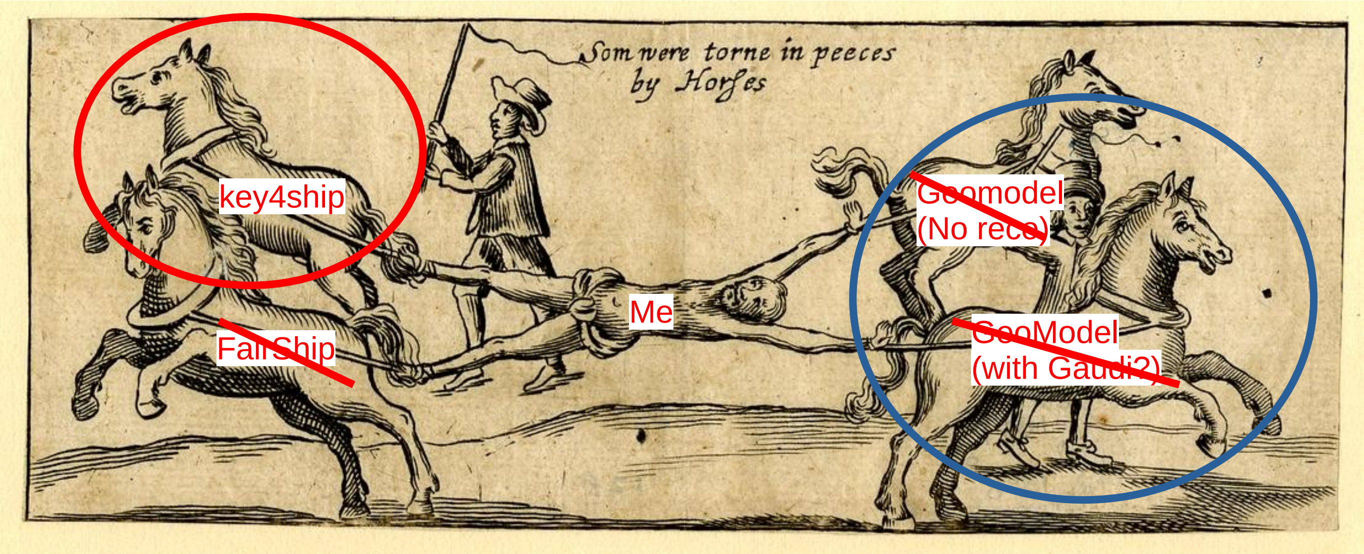
Basically **everyone wants GeoModel** because it's easy for building the geometry but there's no define pipeline after that

Current status in SHiP software

- ▶Can be reduced to a single word: uncertainty
- ▶I don’t know what to do
- ▶I don’t want to waste resources
- ▶Right now there are too many options, mostly in opposite directions

Collaboration meeting is in 2 weeks, I’ll present the key4hep PoC

For the next weeks I’m putting all the money in this horse, let’s hope is the winning one



Most people are into this side, because they like GeoModel

Back-up