Cinema and Prompt

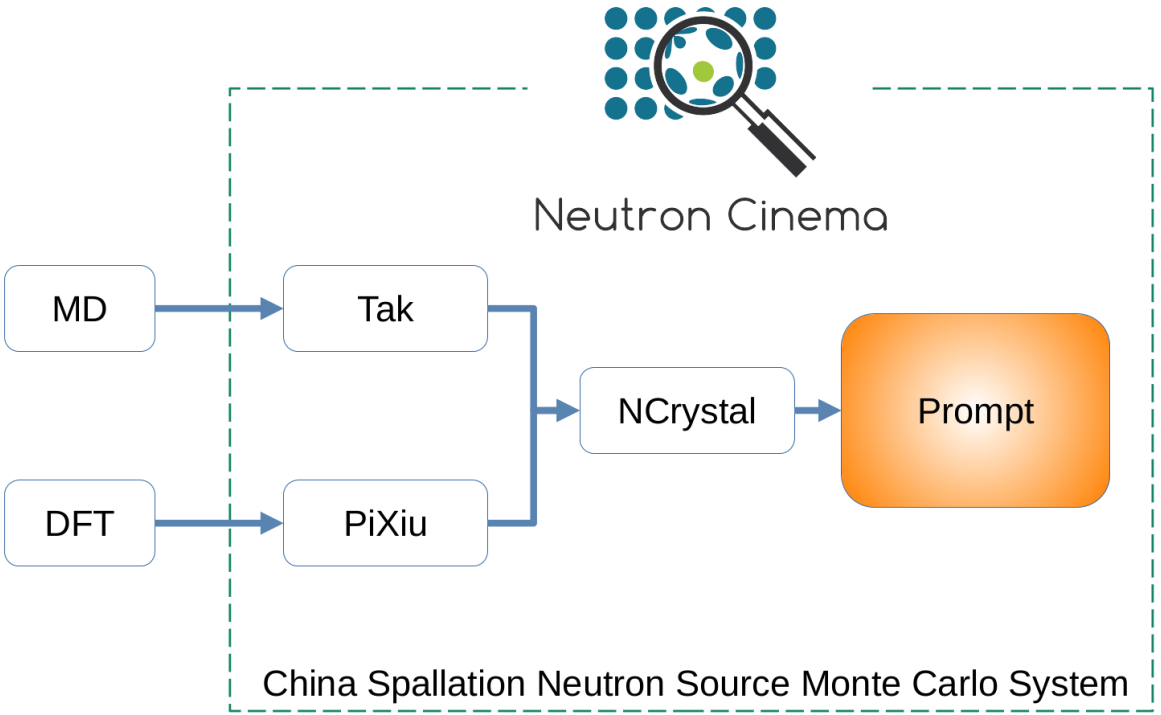
In a nutshell

Xiao-Xiao Cai
caixx@ihep.ac.cn

China Spallation Neutron Source

26 May 2023

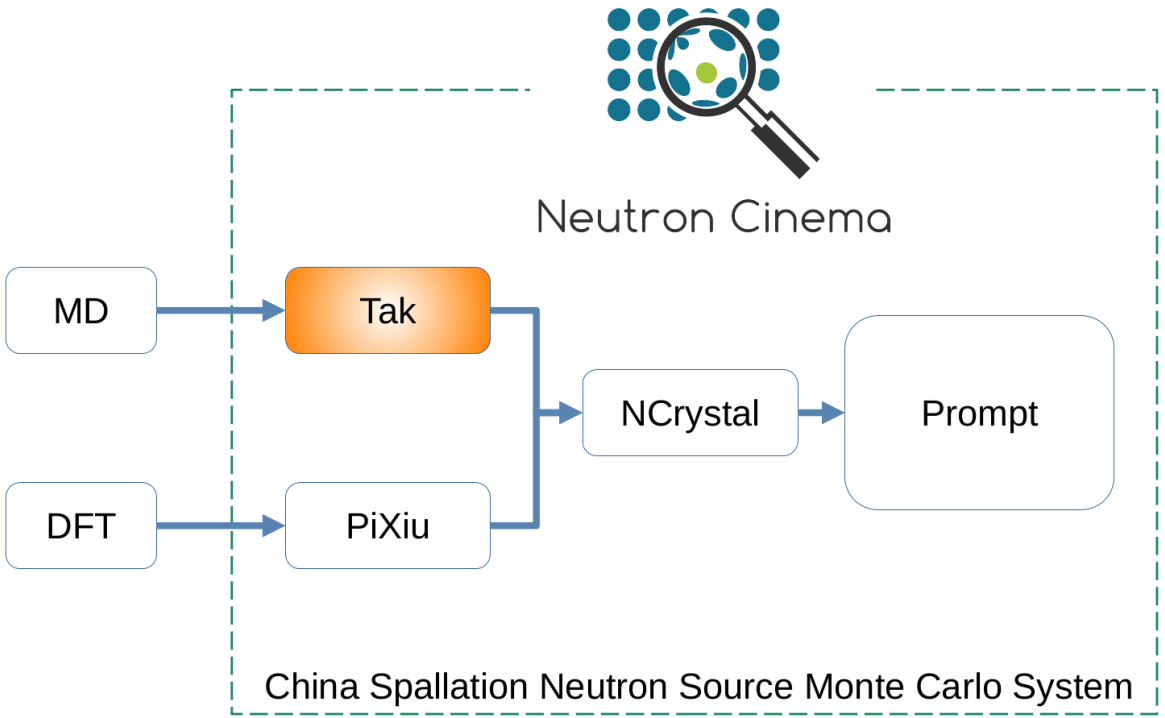
Cinema: China Spallation Neutron Source Monte Carlo System



Prompt, the dual mode Monte Carlo engine:

- Paper @https://arxiv.org/abs/2304.06226.
- Source @https://gitlab.com/cinema-developers/prompt.
- Installing as `pip install neutron-cinema`.

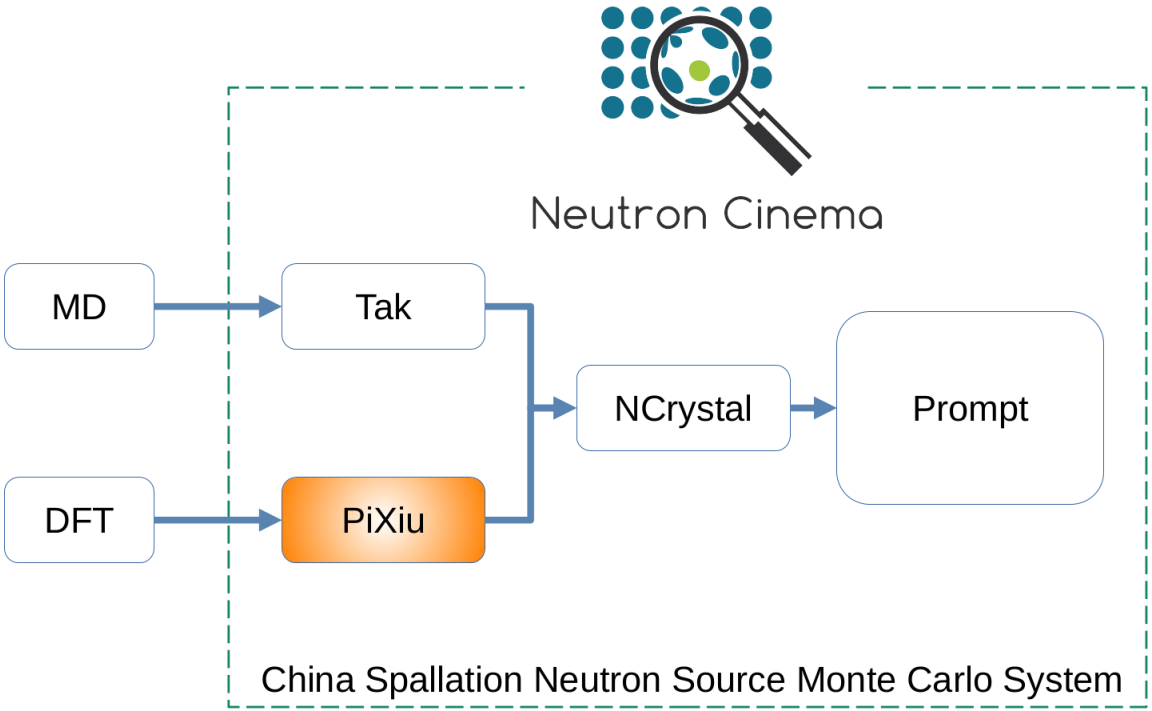
Cinema: China Spallation Neutron Source Monte Carlo System



Tak, Trajectory Analysis Toolkit:

- Incoherent inelastic Calculator published in 2022 [1].
- Coherent inelastic calculator is under development.
- More introduction in the next talk.

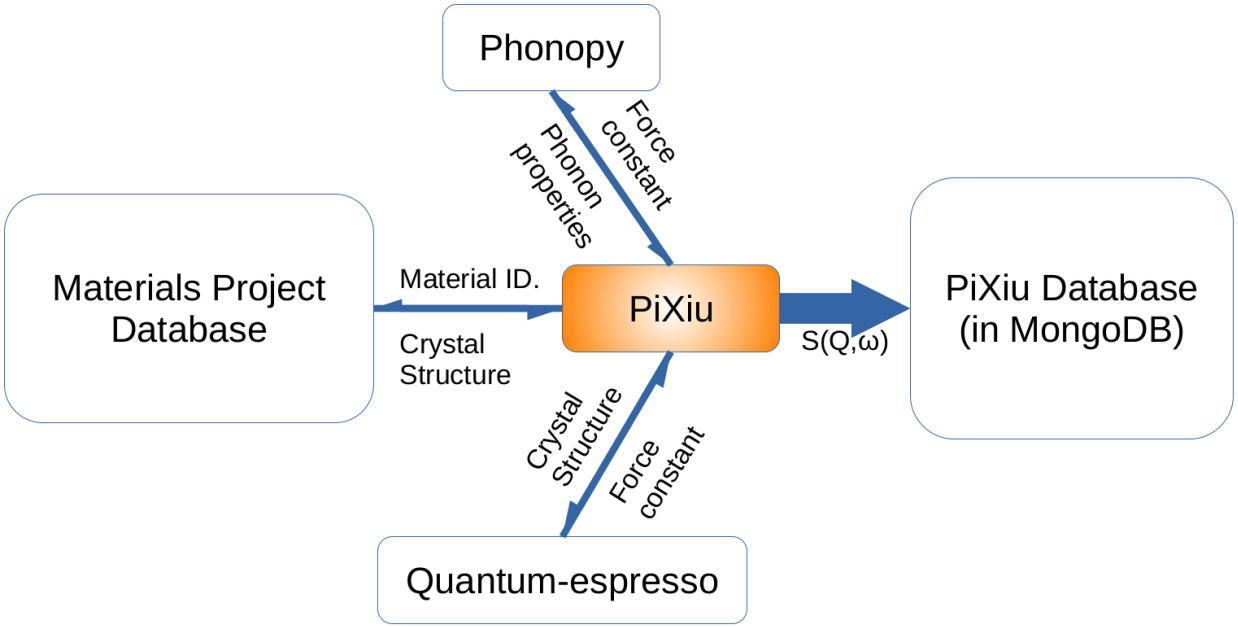
Cinema: China Spallation Neutron Source Monte Carlo System



PiXiu, bridging the gap between DFT and $S(\vec{Q},\omega)$:

- DFT post analysis code for single phonon scattering.
- A reference paper is on preparation.

PiXiu(貔貅)



- PiXiu is a dragon that constantly seeks out treasures to eat, and always keeps them in its belly.
- PiXiu package utilises the Apache Spark engine for parallelisation and high-throughput computation.

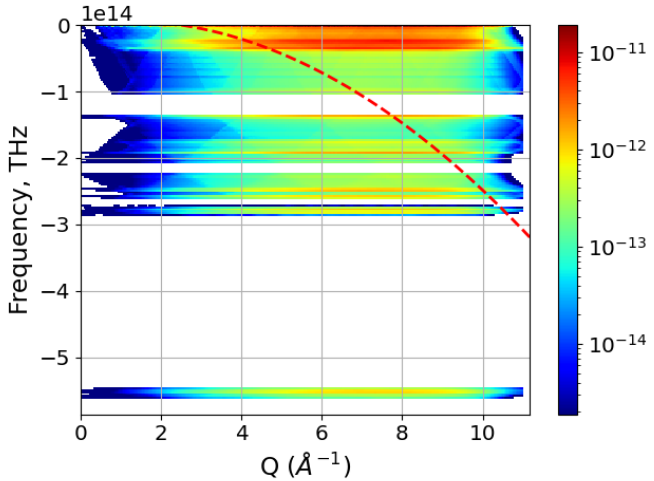
Using PiXiu

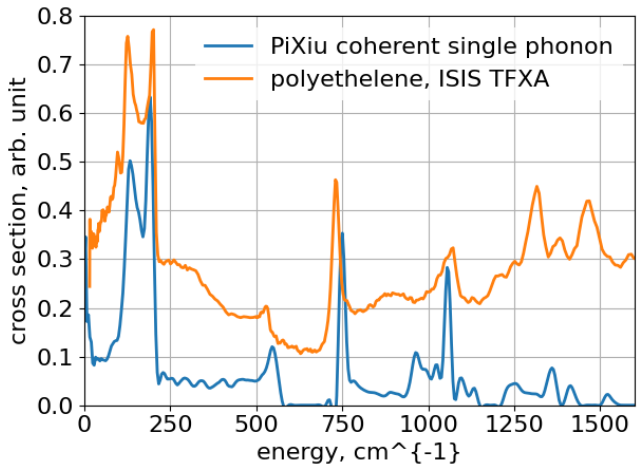
Procedure:

1. Calculate force constant after a default relaxation stage.
2. Generate $S(Q, \omega)$ at a given temperature.
3. Interpolate the trajectory for the given angle and energy.

An example of calculating polyethylene:

```
px_pre.py --mp-id mp-985782 --vanDerWaals --numcpu 256 --rundft
px_inelastic_direct.py --temperature 20 --upper-limit-Q 10
px_inelastic_indirect.py --scattering-angle 135 --energy-out 0.00384
```

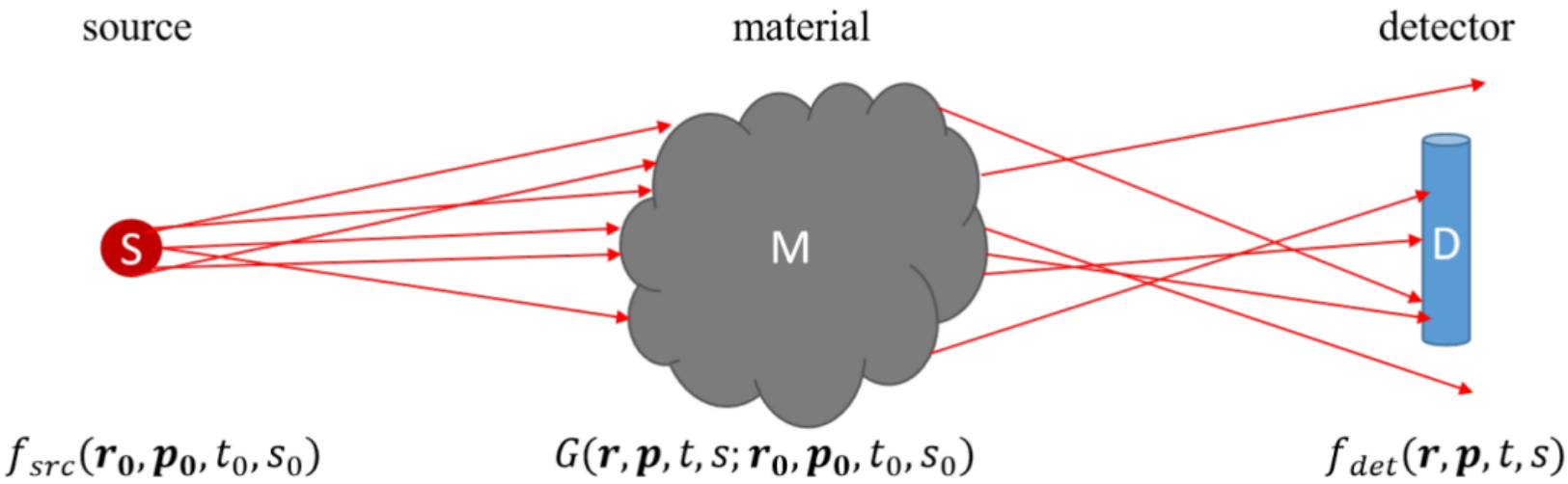




Challenges and Outlook for PiXiu

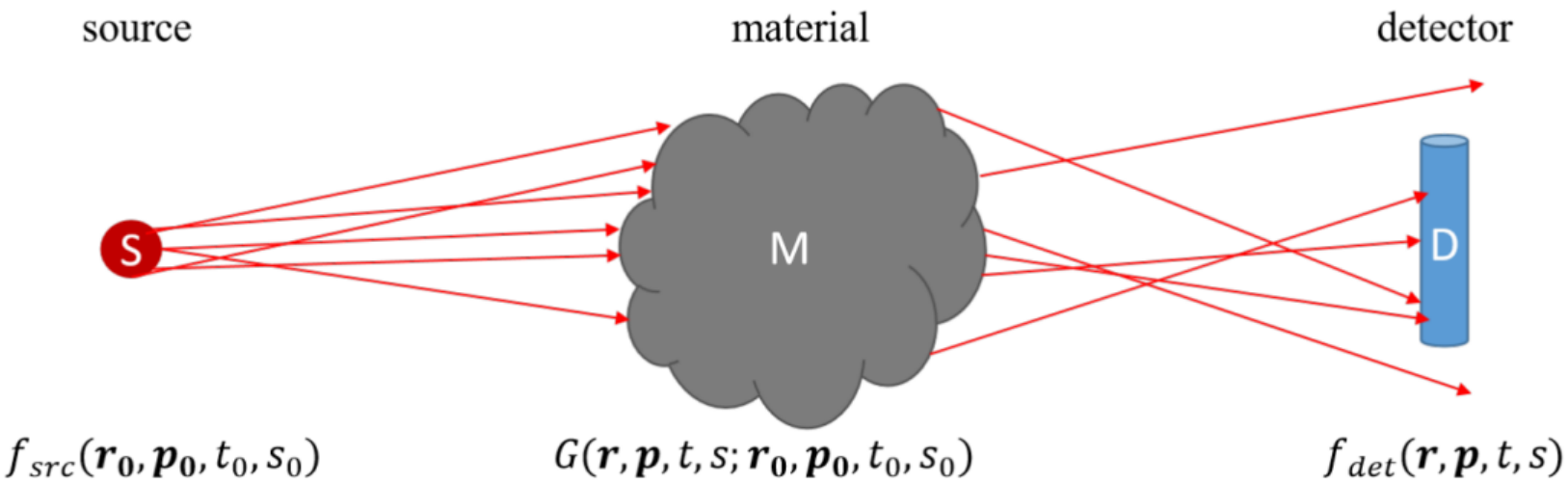
- Challenges:
  1. The $S(Q, \omega)$ is in fact a histogram. Very fine phonon meshes are required to obtain smooth data. An efficient statistical method is under development to reduce the computational cost.
  2. The small Q region is important but numerically and/or physically unstable.
- Outlook:
  1. The software reference paper is under preparation.
  2. Coherent/incoherent single phonon scattering calculator for NCrystal.

Prompt: Probability-Conserved Cross Section Biasing Monte Carlo Particle Transport System



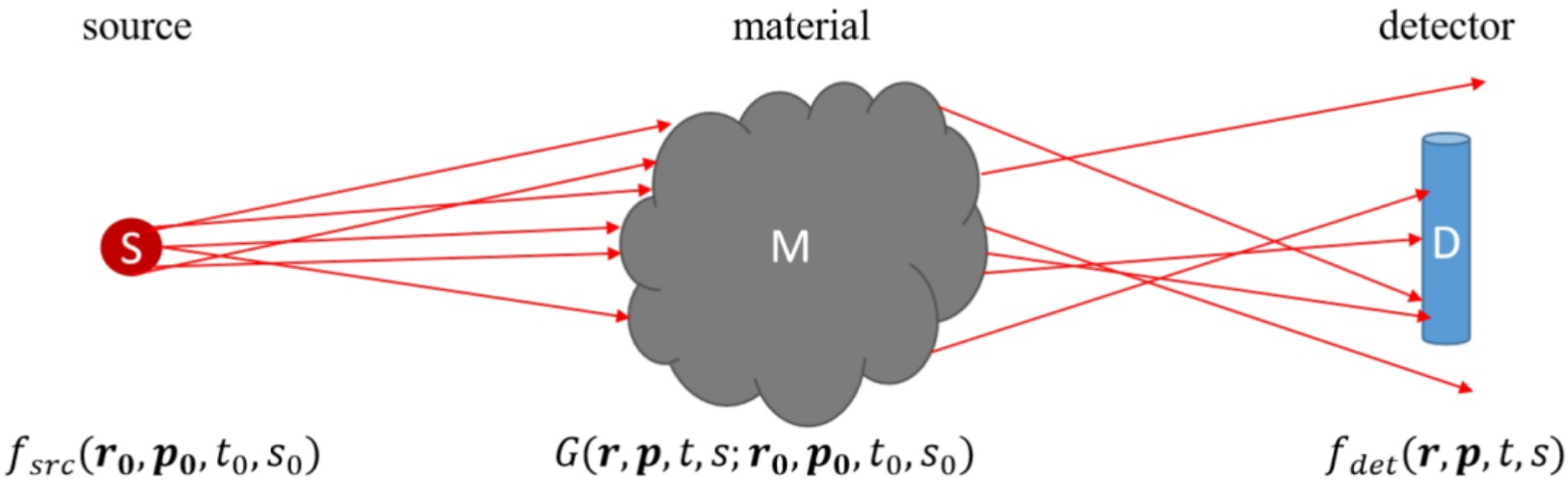
- The dual mode Monte Carlo engine in Cinema. To make cross comparisons with McStas.
- G is the propagator, $\boldsymbol{r}$ is the position, $\boldsymbol{p}$ is the momentum, t is the time, s is the spin.

The transport mode



$$f_{det}(\boldsymbol{r}, \boldsymbol{p}, t, s) = \int \mathrm{d}^3\boldsymbol{r}_0 \int \mathrm{d}^3\boldsymbol{p}_0 \int \mathrm{d}t_0 \int \mathrm{d}s_0 \, G(\boldsymbol{r}, \boldsymbol{p}, t, s; \boldsymbol{r}_0, \boldsymbol{p}_0, t_0, s_0) f_{src}(\boldsymbol{r}_0, \boldsymbol{p}_0, t_0, s_0)$$

The ray-tracing mode



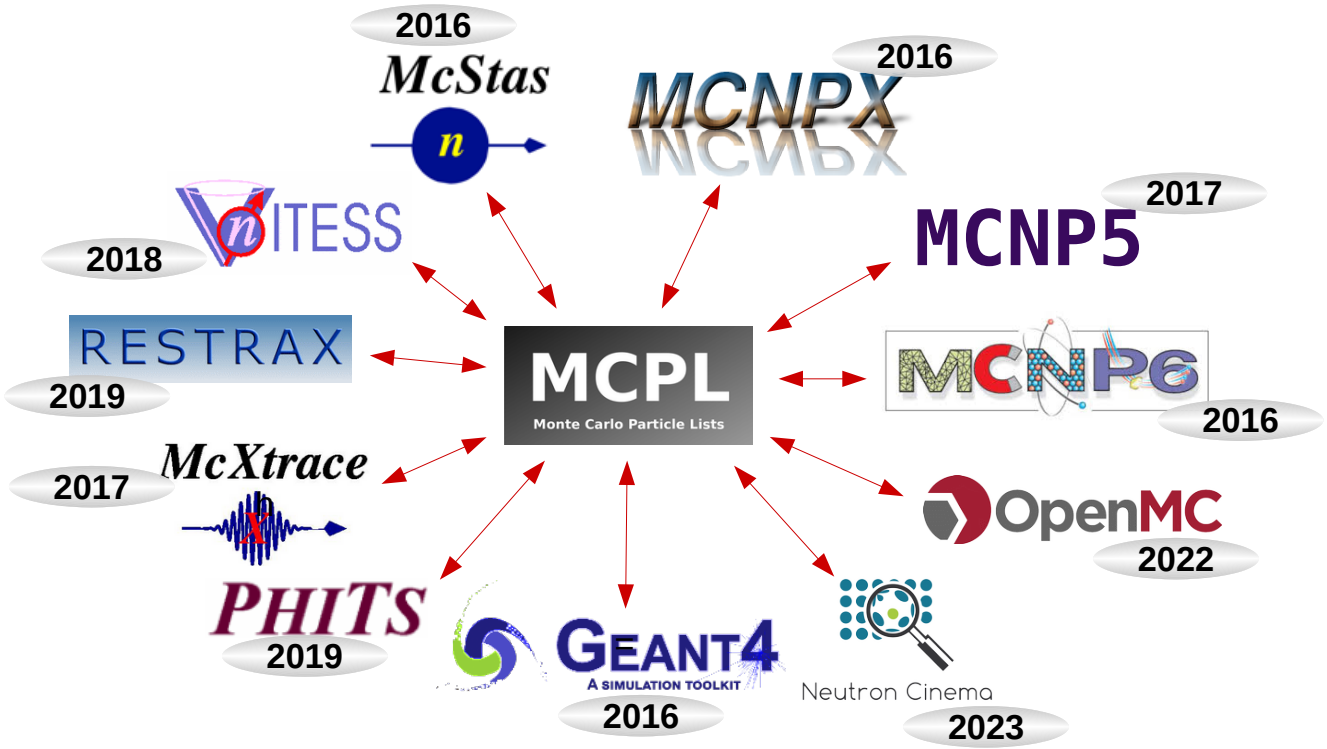
The high order integration is approximated by an efficiency function. Mainly for chopper.

$$f_{det}(\boldsymbol{r}, \boldsymbol{p}, t, s) = \epsilon(\boldsymbol{r}, \boldsymbol{p}, t, s; \boldsymbol{r}_0, \boldsymbol{p}_0, t_0, s_0) f_{src}(\boldsymbol{r}_0, \boldsymbol{p}_0, t_0, s_0)$$

MCPL ecosystem

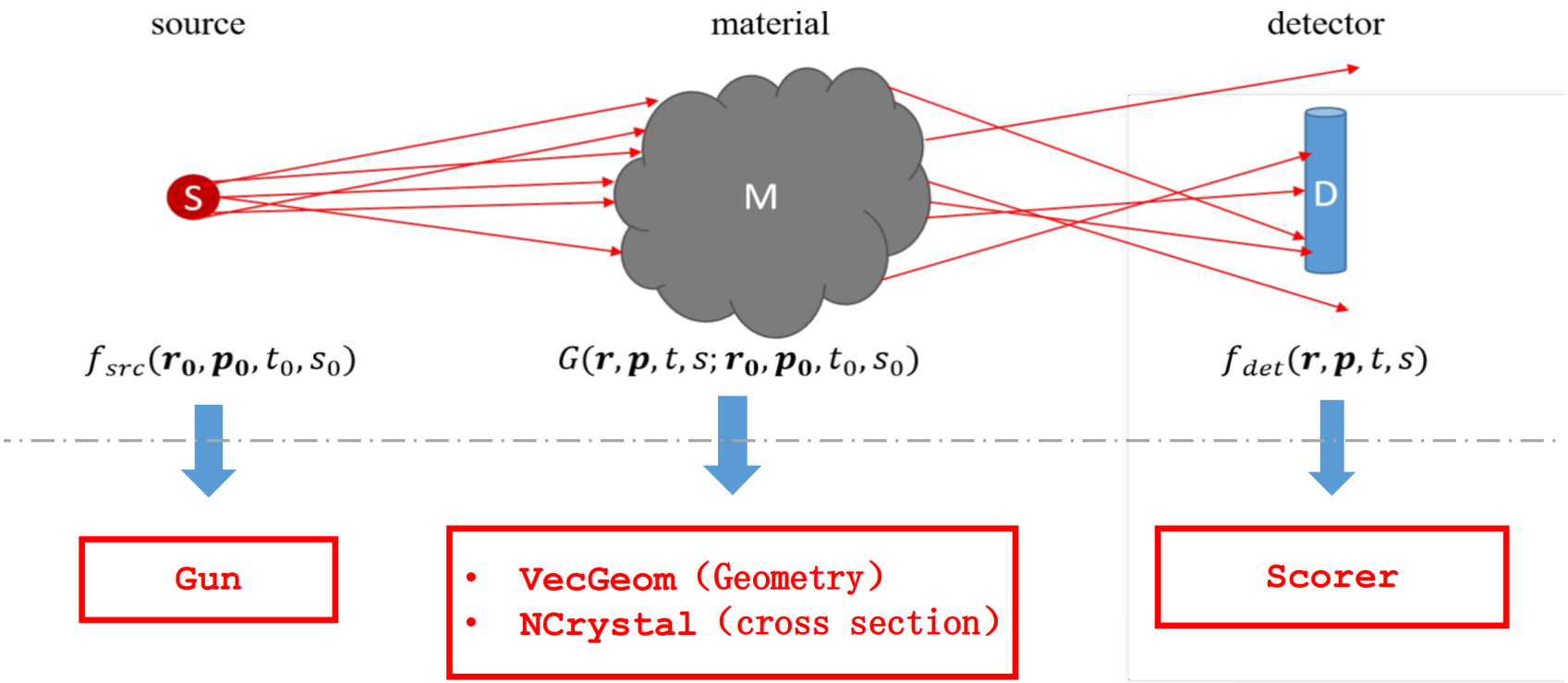
T. Kittelmann MCPL

Codes with MCPL support



1 / 1

Implementation



Multiple scattering in samples

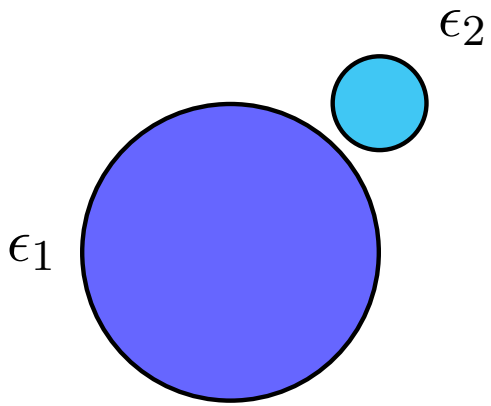
Single scattering ϵ_1 is typically weak ($<10\%$) but important.

- The number of multiple scattering events decay exponentially with the order.

Multiple scattering in samples

Single scattering ϵ_1 is typically weak ($<10\%$) but important.

- The number of multiple scattering events decay exponentially with the order.



The problem:
Convectional weight window splitting is not always helpful.

Multiple scattering in samples

Single scattering ϵ_1 is typically weak ($<10\%$) but important.

- The number of multiple scattering events decay exponentially with the order.

The idea:

- Scale up the total cross section by a factor g, so that the multiple scatterings is increased exponentially.

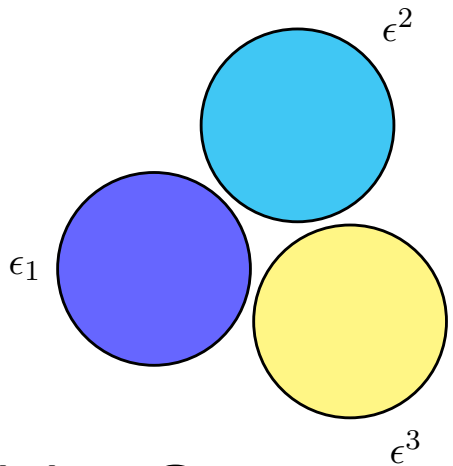
Multiple scattering in samples

Single scattering ϵ_1 is typically weak ($<10\%$) but important.

- The number of multiple scattering events decay exponentially with the order.

The idea:

- Scale up the total cross section by a factor g, so that the multiple scatterings is increased exponentially.



Probability Conservation?

The impact of scaling the cross section

$$\because \quad P(x) = \exp(-\rho\sigma x)$$

$$\therefore \quad \frac{\mathrm{d}P(x)}{P(x)} = -\rho(x)\sigma \mathrm{d}x \quad \Rightarrow \quad \frac{\mathrm{d}P_g(x)}{P_g(x)} = -\rho(x)\sigma_g \mathrm{d}x$$

By the quotient rule of derivative, the impact of cross section scaling on the probability can be seen

$$\frac{\mathrm{d}P_g(x)/P(x)}{\mathrm{d}P_g(x)/P(x)} = -(g-1)\rho(x)\sigma \mathrm{d}x$$

It is possible to perform weight corrections by the so-called *probability-conserving cross-section biasing mechanism* proposed by Mendenhall and Weller [2].

Weight correction

The correct weight of neutron at the end of the n_{th} step can be computed as

$$W_n = W_0 \prod_{l=1}^{n} \frac{1}{g_{m,l}} \prod_{m=1}^{n} \prod_{i=1}^{N_p} \exp[(g_{m,i} - 1) x_m \rho \sigma_i]$$

- W_o is the original weight of the particle.
- n is the number of interactions.
- x_m is the step length at the m_{th} step.
- N_p is the number of reaction processes.
- $g_{m,i}$ is the biasing factor for the i_{th} process at the m_{th}.
- $g_{m,l}$ is the biasing factor for the process occurs at the end of the m_{th} step.

Step length sampling

That implies the sampling method for step length should be changed accordingly

$$l = -\frac{\log_e(\xi)}{\rho\sigma} \quad \Rightarrow \quad l_g = -\frac{\log_e(\xi)}{g\rho\sigma}$$

- Weight changes alone the neutron trajectory in biased materials.
- g will be made angle and energy dependent as an option.
- $g > 1$ to promote scatterings; $g < 1$ for deep penetrations.
- In practice, for instrument samples, it is recommended to make $g\epsilon_1 \approx 0.5$ to be effective and not supercritical.

Define biasing factors in NCrystal

An aluminium polycrystal with biasing factors

```
"physics=ncrystal;nccfg='Al_sg225.ncmat';scatter_bias=2.0;abs_bias=2.0"
```

- The biasing factors are unity by default
- If a biasing factor is set to zero, the corresponding process is switched off.

An aluminium polycrystal with unity biasing factors

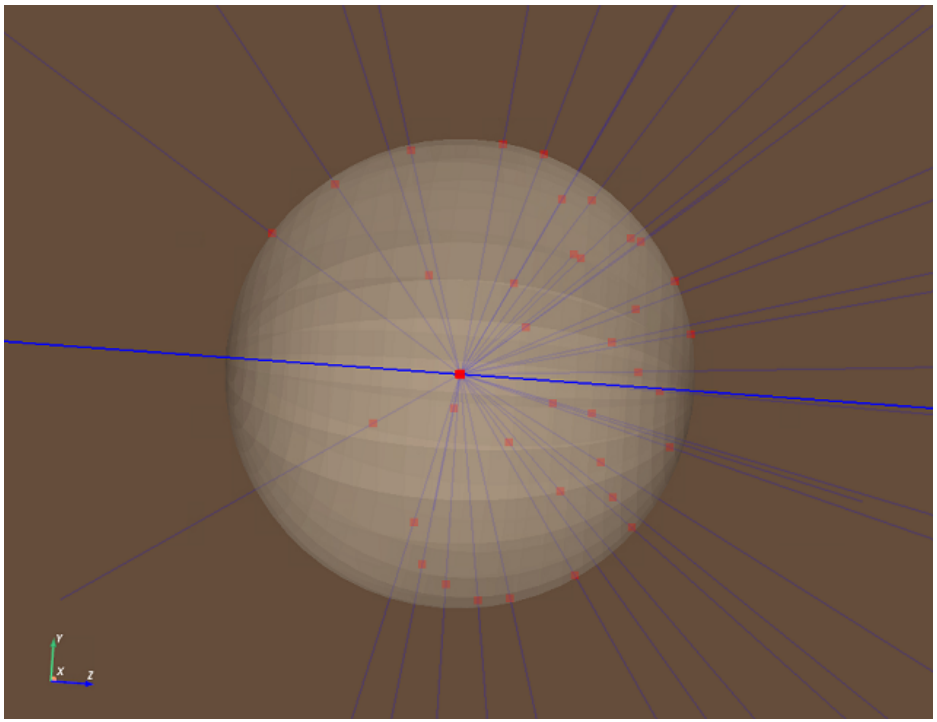
```
'Al_sg225.ncmat'
```

- This string is passed to NCrystal directly.

Example

A total scattering setup

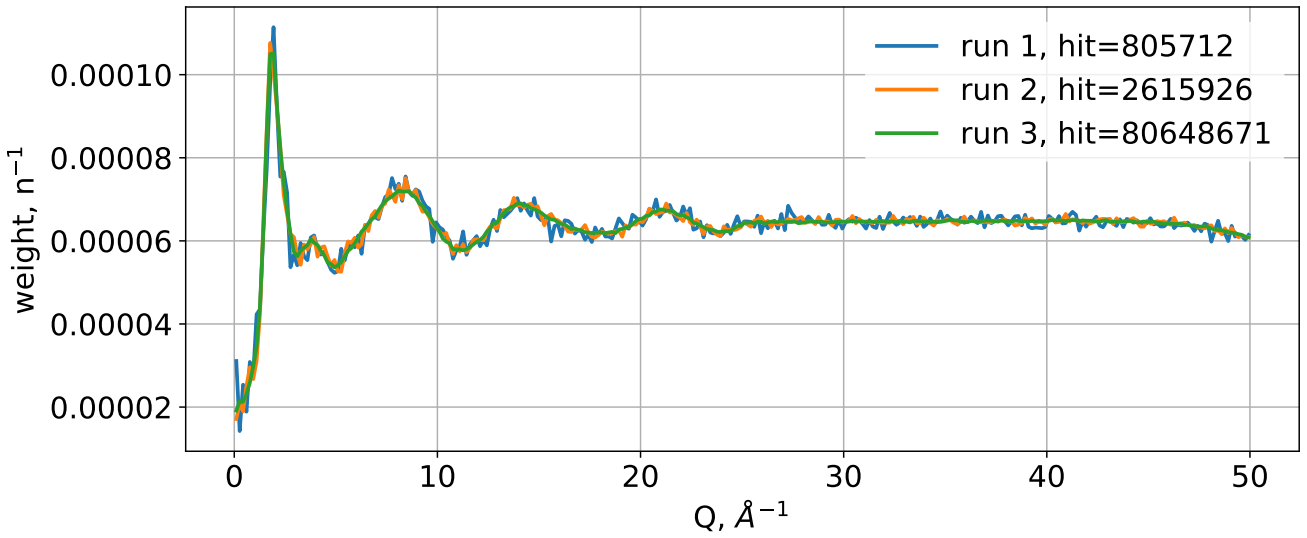
- incident monoenergetic $0.15\,\text{Å}$ neutron
- CAB heavy water spherical sample with $2\,\text{mm}$ radius
- a hypothetical energy and position sensitive 4π detector



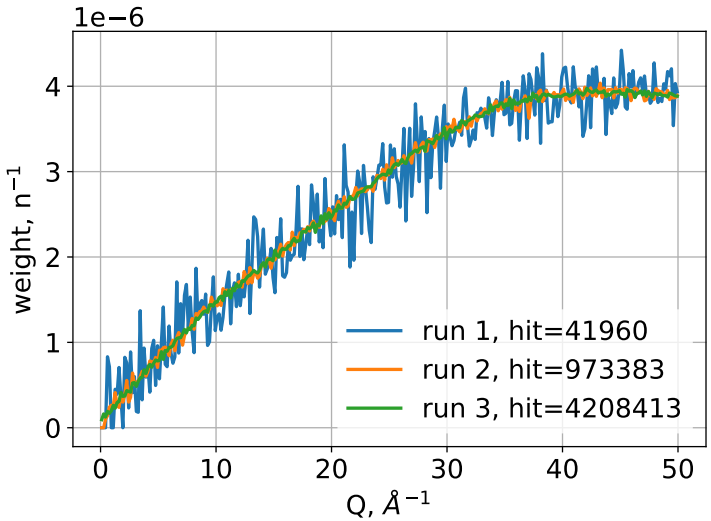
Neutron weighted structure factor

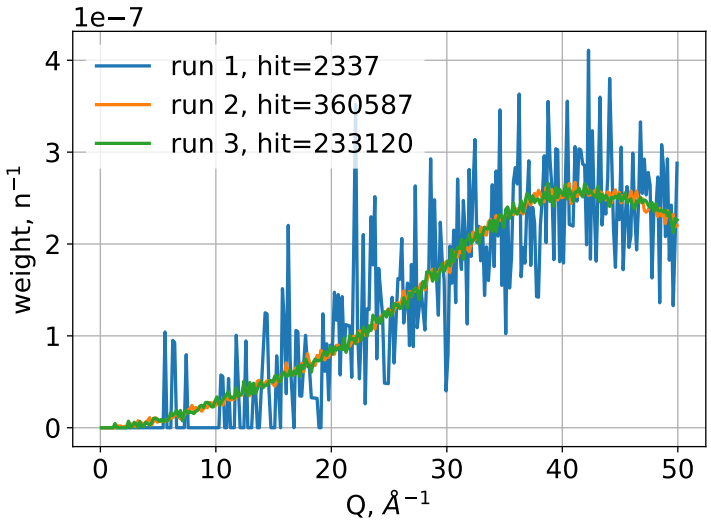
run	neutron	HW bias	time(s)
1	1e7	1.0	15
2	1e7	8.0	25
3	1e9	1.0	1461

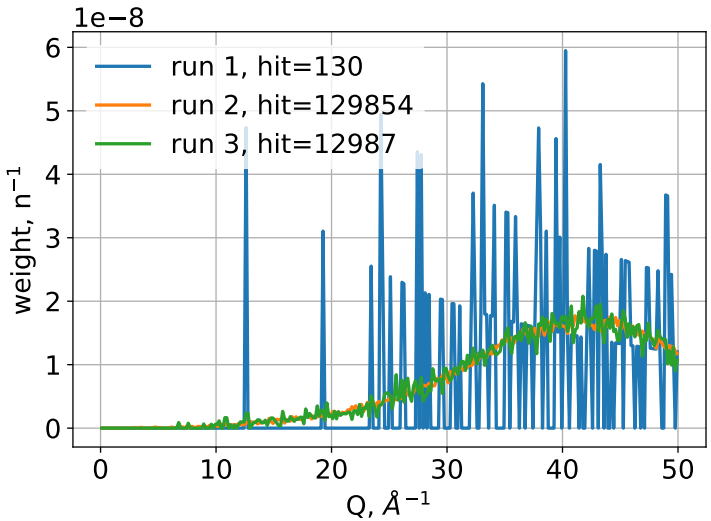
Three runs measuring the distribution of Q for different number of scatterings. The single scattering $P_1(Q)$ is followed

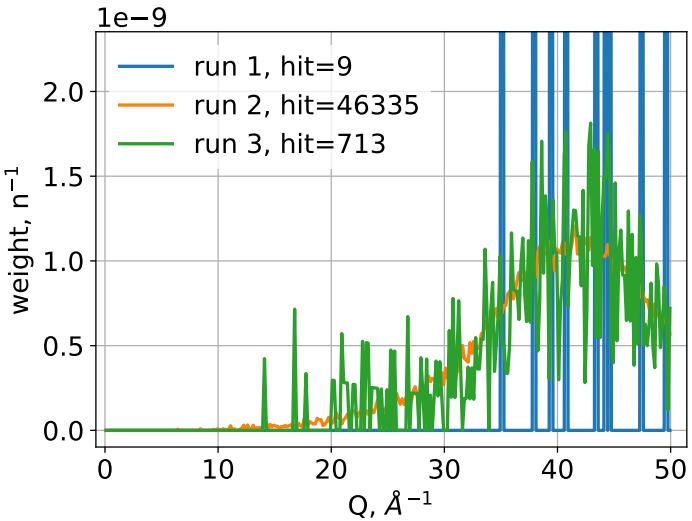


Multiple scatterings, $P_2(Q)$ to $P_5(Q)$









Using Prompt

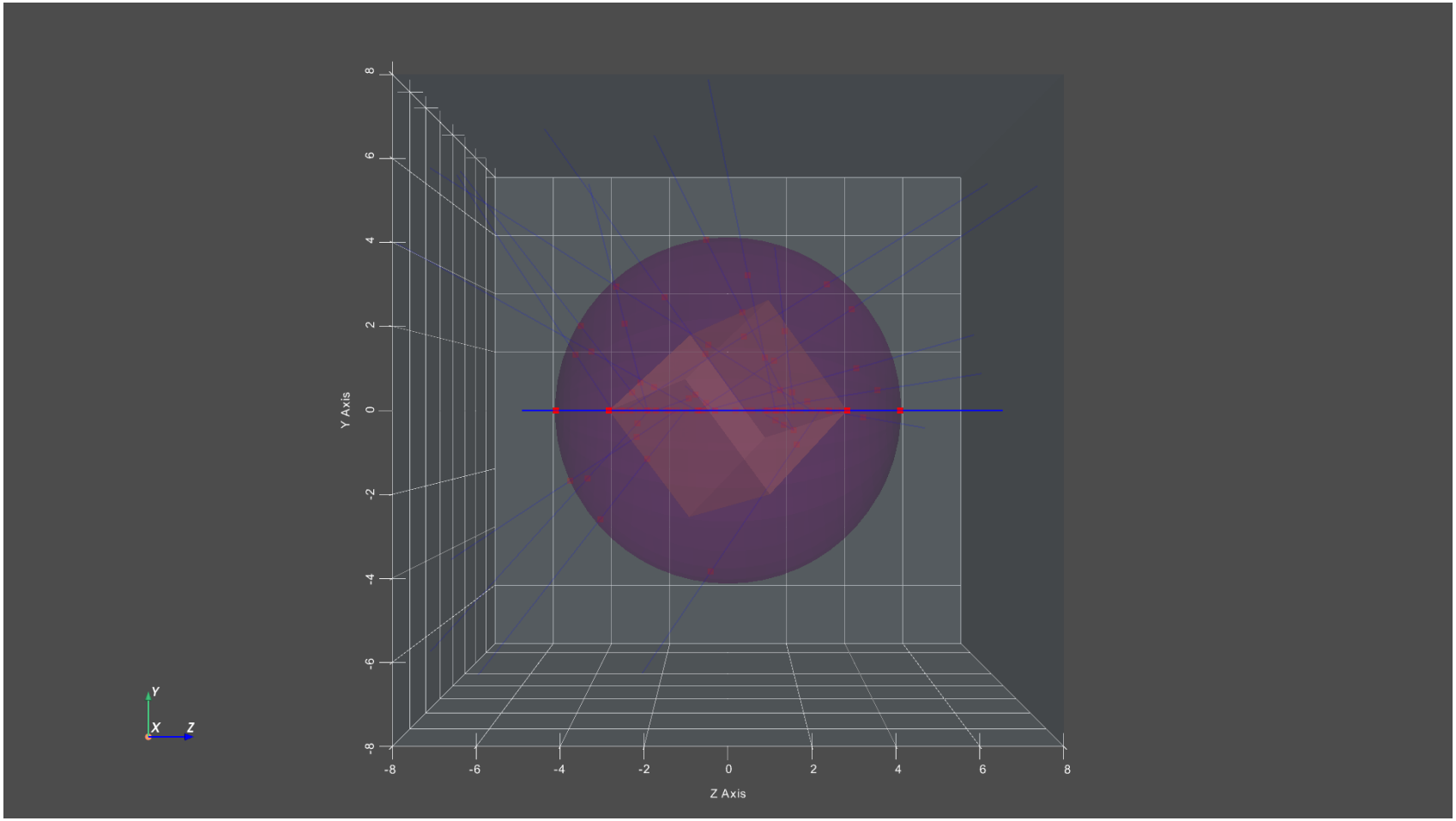
- The published version (v1) reads the input file in the GDML formatted [3].
- The development version (v2) is purely pythonic.
- Materials, scorers and guns in these two version are defined by the same configuration strings.
- The introduction in this talk are in the v2 syntax, as it is much shorter.

An example

```
1  from Cinema.Prompt import Prompt, PromptMPI
2  from Cinema.Prompt.geo import Volume, Transformation3D
3  from Cinema.Prompt.solid import Box, Sphere
4
5  class MySim(Prompt):
6      def __init__(self, seed) -> None:
7          super().__init__(seed)
8
9      def makeWorld(self):
10         self.setGun('gun=SimpleThermalGun;position=0,0,-6')
11         world = Volume('world', Box(8, 8, 8))
12         sp = Volume('sphere', Sphere(rmin=0, rmax=5), 'Al_sg225.ncmat')
13         world.placeChild('sp', sp)
14
15         box  = Volume('box', Box(2,2,2), 'V_sg229.ncmat')
16         trs = Transformation3D()
17         trs.rotAlignement([[1,1,1],[0,1,0]], [[0,0,1], [1,1,1]])
18         sp.placeChild('box', box, trs)
19         self.setWorld(world)
20
21 sim = MySim(seed=4096)
22 sim.makeWorld()
23 sim.show(num=100)
```

Visualisation

PyVista [4] is used for the geometry visualisation.



Some concepts

- `Active volume`: the volume contains the particle, at the deepest possible level in the volume hierarchy.

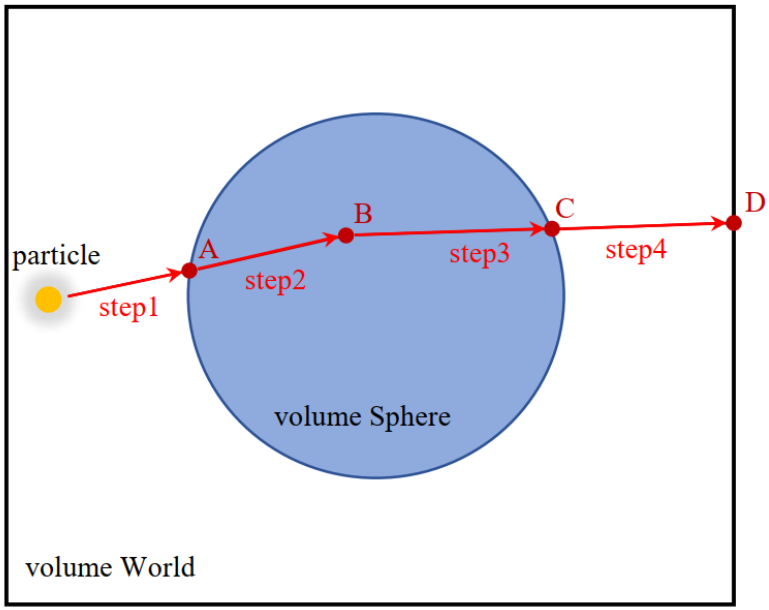
Some concepts

- `Active volume`: the volume contains the particle, at the deepest possible level in the volume hierarchy.
- `ptstate` triggers physics models and scorers.

Some concepts

- `Active volume`: the volume contains the particle, at the deepest possible level in the volume hierarchy.
- `ptstate` triggers physics models and scorers.
- `ptstate` describes the relationship between the particle and the `active volume`.
  - `SURFACE`: a particle reaches the boundary of a volume.
  - `ENTRY`: a particle entries a volume.
  - `PROPAGATE`: a particle propagates in a volume.
  - `ABSORB`: a particle is absorbed in a volume.
  - `EXIT`: a particle exits a volume.

Event triggering example



- Piont A, `EXIT` for the World and `ENTRY` for the Sphere.
- Point B, `PROPAGATE` for the Sphere.
- Point C, `EXIT` for the Sphere and `ENTRY` for the World.
- Point D, `EXIT` for the World.

Available scorers

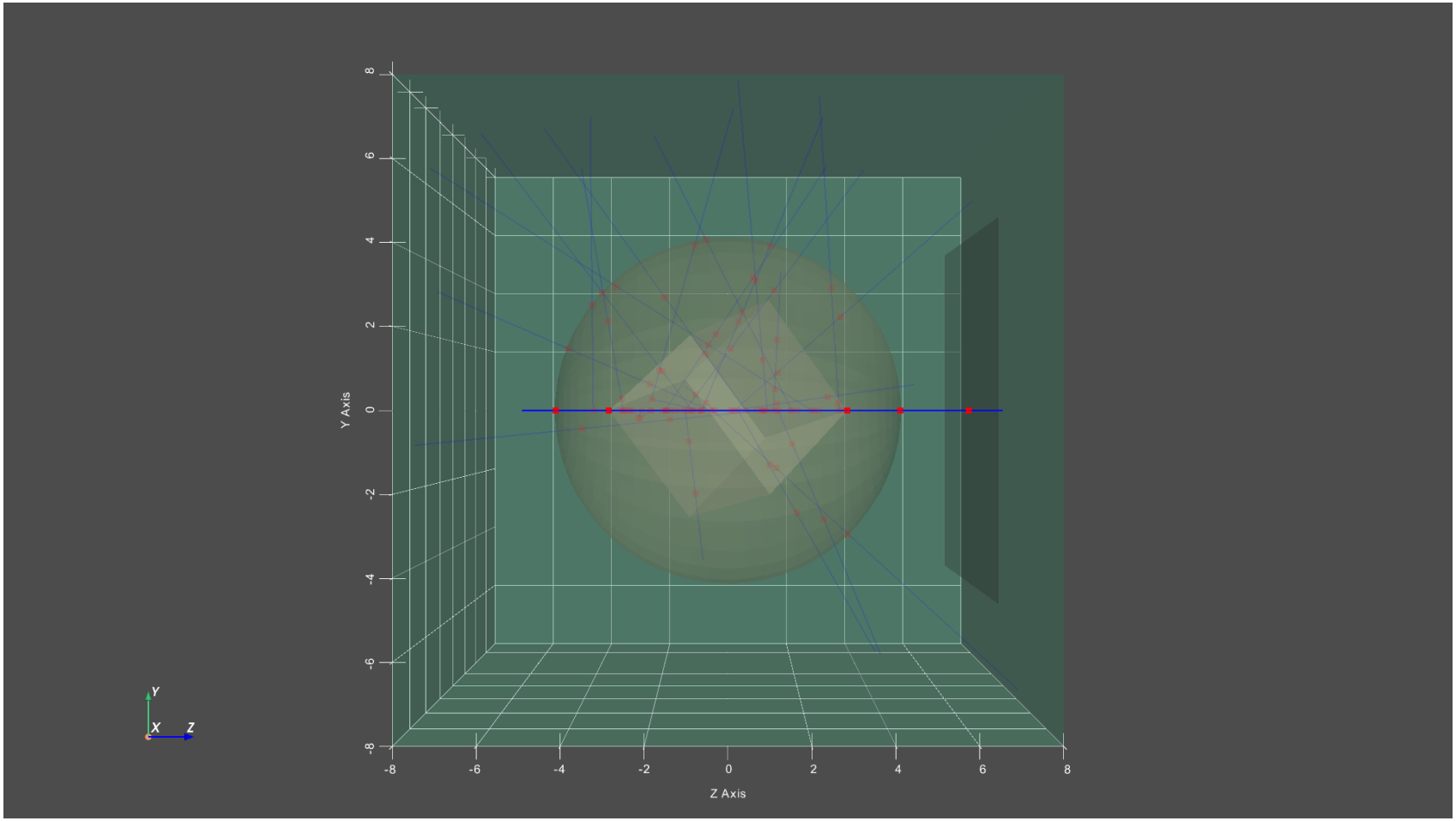
- `ESpectrum` , Energy Spectrum
- `WlSpectrum` , Wavelength spectrum
- `TOF` , Time-of-flight spectrum of particles.
- `PSD` , Two-dimensional spatial spectrum on the projection of a volume
- `MultiScat` , Scattering number spectrum of scattered in a volume
- `DeltaMomentum` , Momentum transfer spectrum
- `Angular` , Angular spectrum
- `VolFluence` , Particle fluence spectrum in a volume

Adding scorers

```
class MySim(Prompt):
    ...
    def makeWorld(self):
        ...
        self.scorer['psd'] = """Scorer=PSD;name=apsd;xmin=-10.;
                                xmax=10.;numbin_x=50;ymin=-10.;
                                ymax=10.;numbin_y=50.;ptstate=SURFACE"""

        self.scorer['wl'] = """Scorer=WlSpectrum; name=wl;
                               min=0.0; max=10; numbin=100; ptstate=SURFACE"""


        det = Volume('thinbox', Box(5, 5,1e-3))
        det.addScorer(self.scorer['psd'] )
        det.addScorer(self.scorer['wl'] )
        world.placeChild('psd', det, Transformation3D(0., 0., 7),)
        ...
```

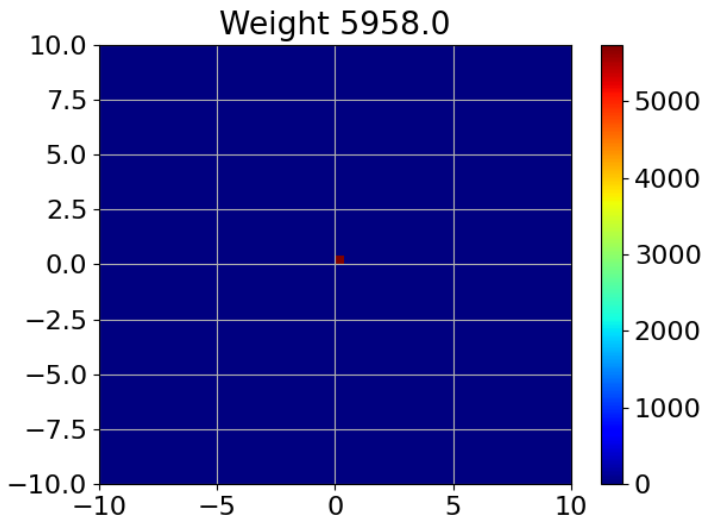
Helpers are also available to make coding more convenient.

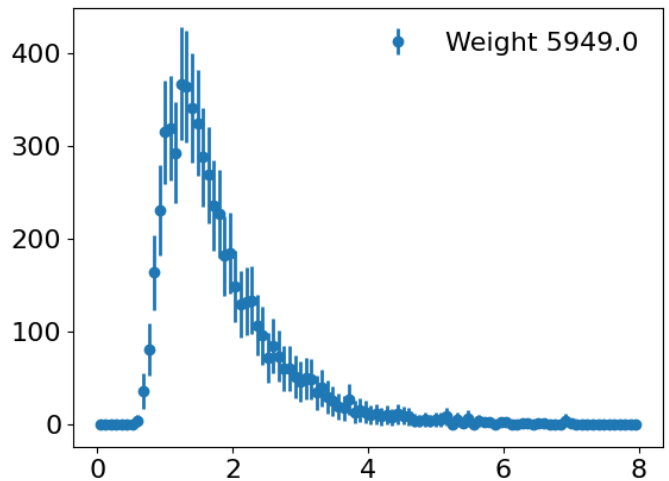
The World with a detector



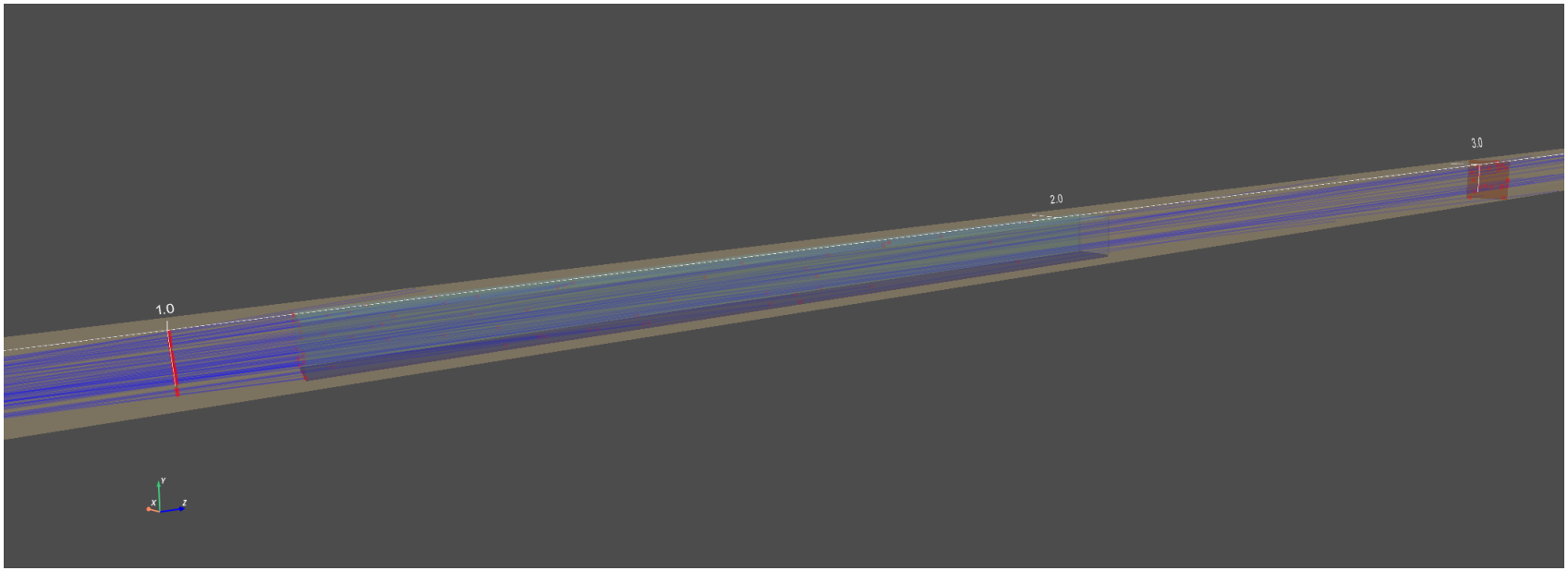
Get the results

```
...
sim.simulate(1e4)

pdfhist = sim.getScorerHist('psd')
pdfhist.plot()

wlhist = sim.getScorerHist('wl')
wlhist.plot(show=True)
```



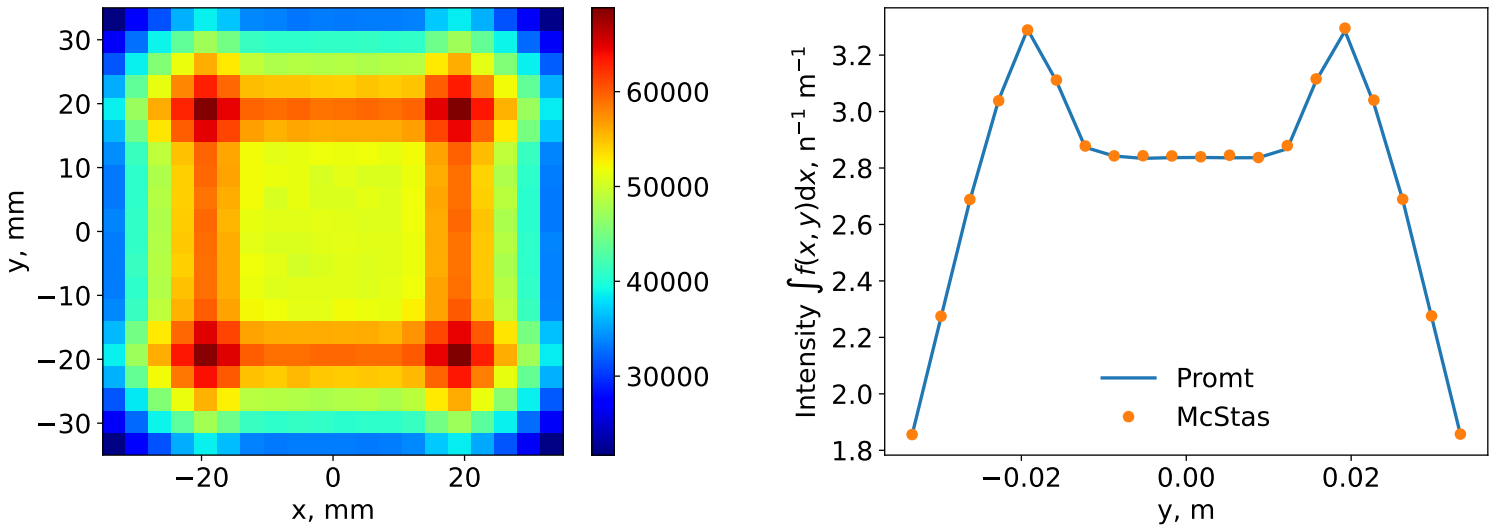


Guide



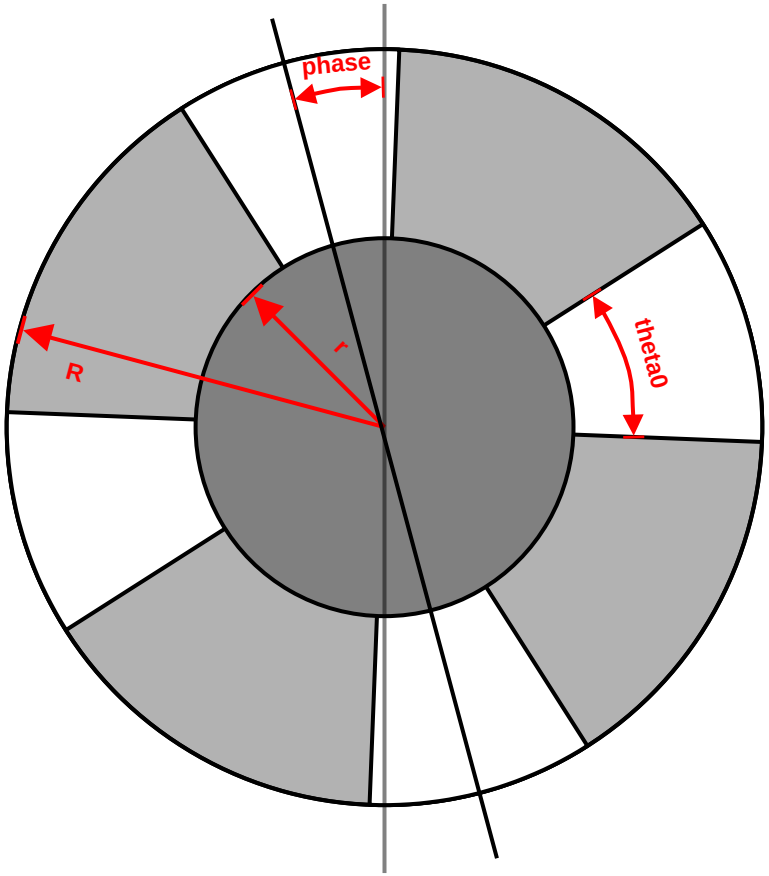
- The reflectivity model is taken from McStas.
- The `Test_Guides.instr` setup in the McStas optical examples is reproduced. The simulation world contains two position sensitive detectors and a guide.

Guide: benchmark against McStas



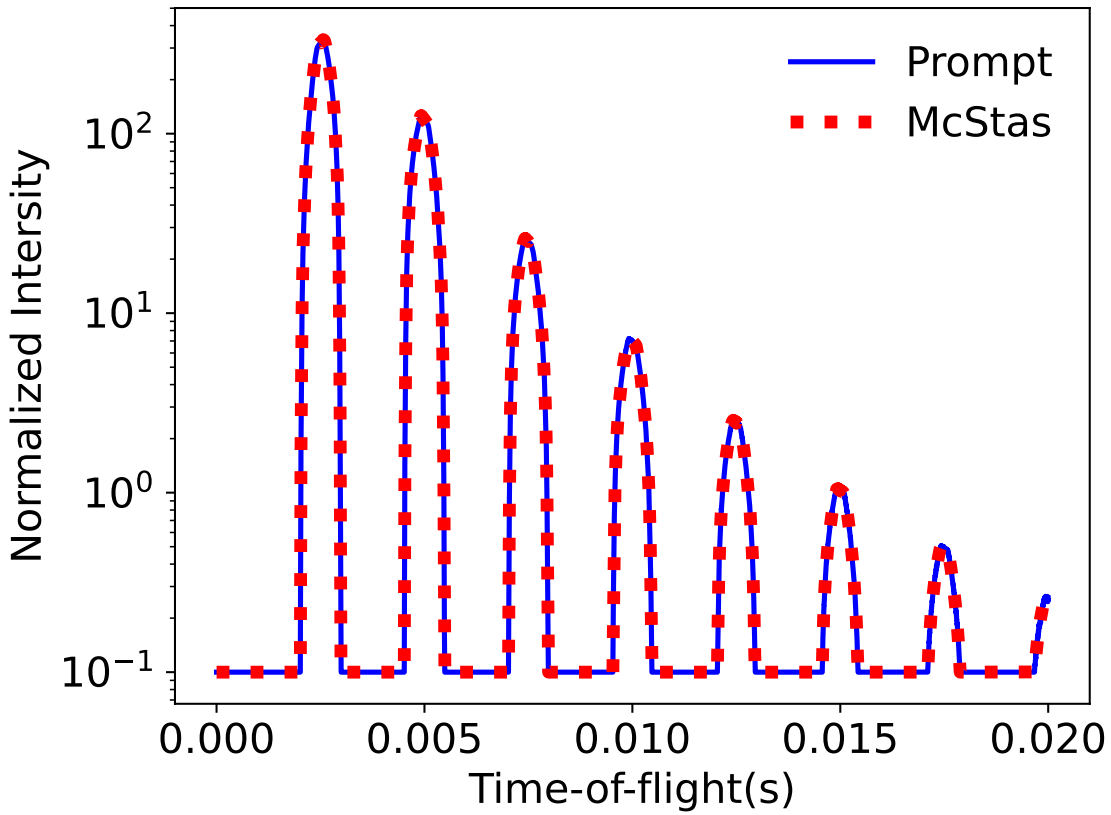
The pattern on the rear PSD looks identical as the one from McStas. The integral intensities agree closely.

Chopper



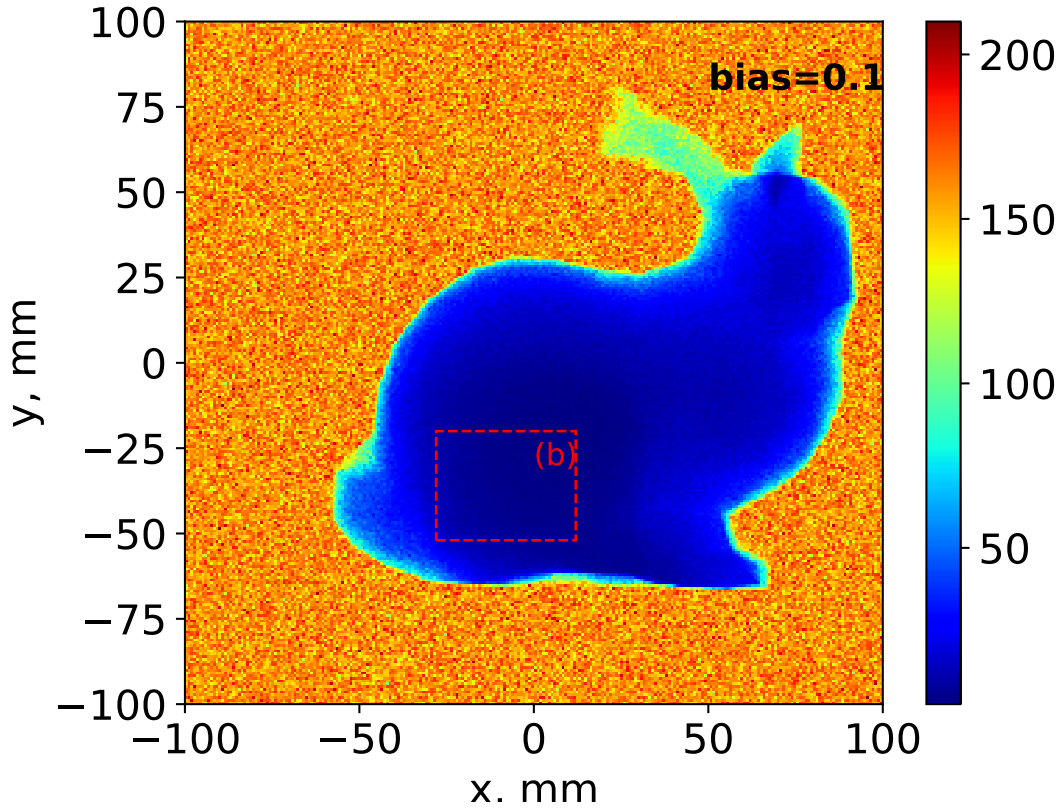
The chopper model is also from the McStas.

Chopper: benchmark against McStas



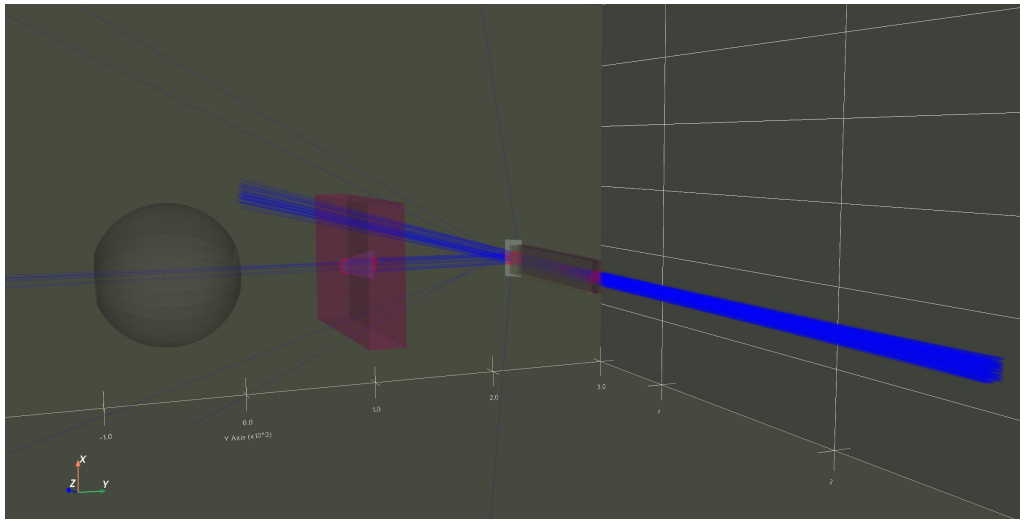
A simple Maxwell spectrum TOF simulation shows excellent agreements with McStas.

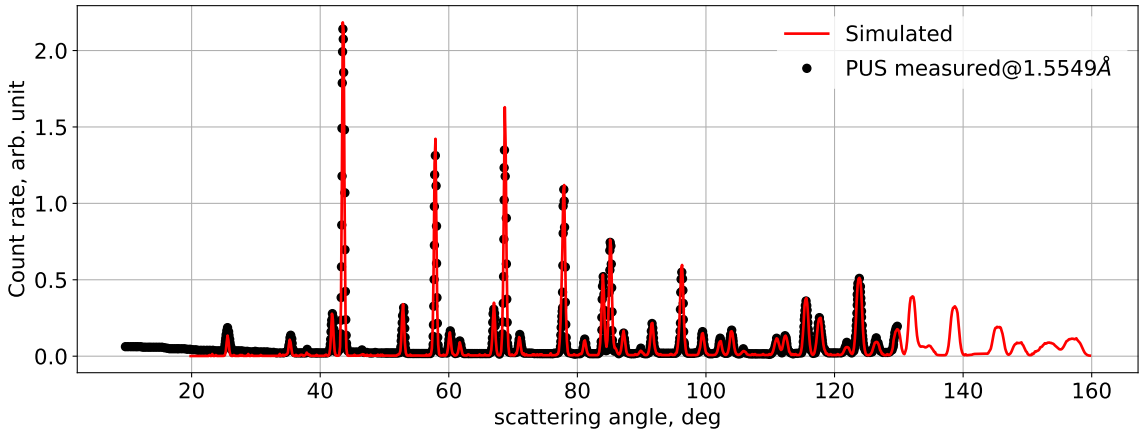
Complex geometry (from the Prompt reference paper)



Diffractometer (from the Prompt reference paper)

The PUS diffractometer in Norway.





Optimisation (experimental feature in v2)

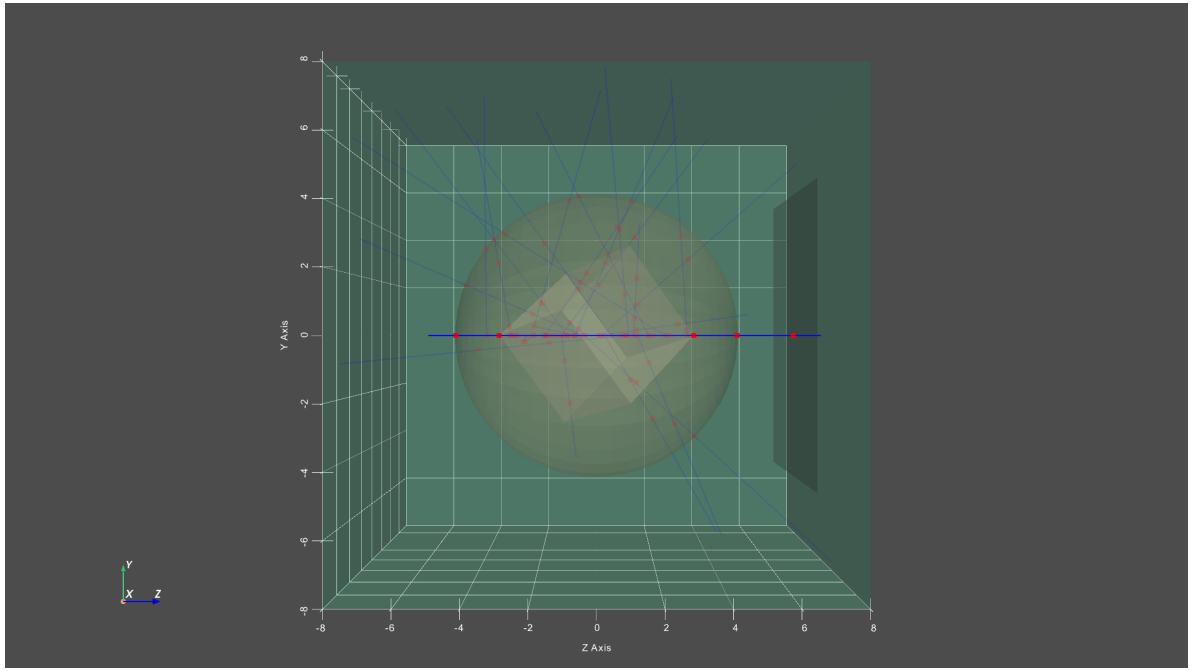
Prompt can optimise simulations, if

1. the objectives are well-defined.
2. the problem can be parameterised and the variables are important.
3. the constraints are known.

Status

1. currently based on optuna [5].
2. supports continuous, discrete and categorical distributed parameters.

Example: point source position for minimal detector count rate

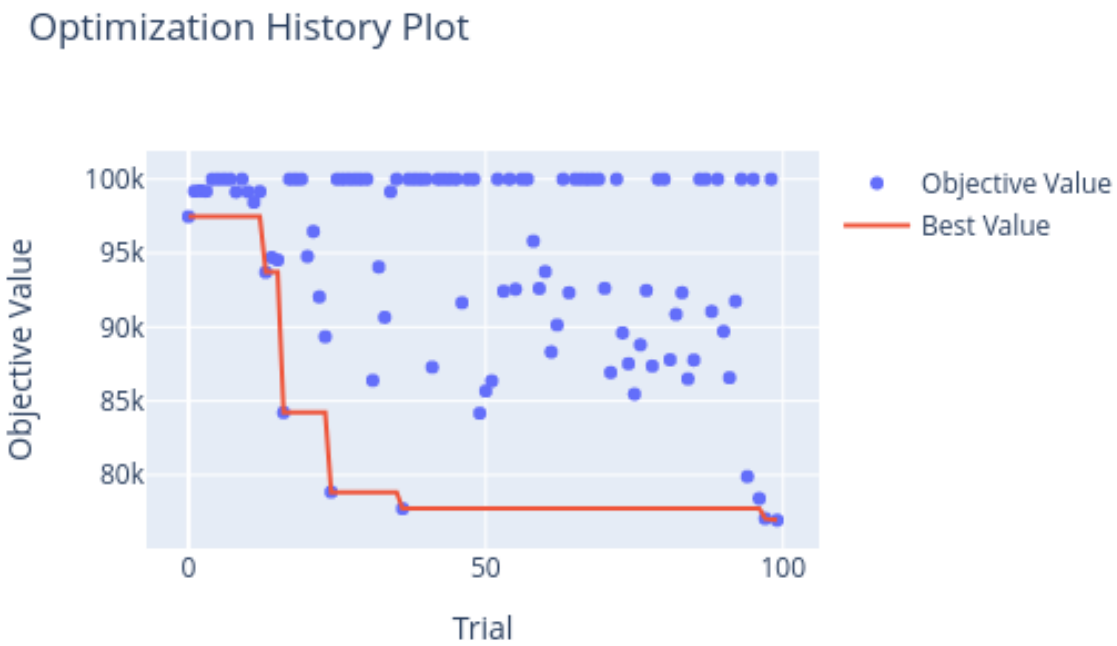


1. Objective: the count rate of the detector is minimal.
2. Variables: the point source position.
3. Constraint: the position is outside the AI sphere.

Optimisation

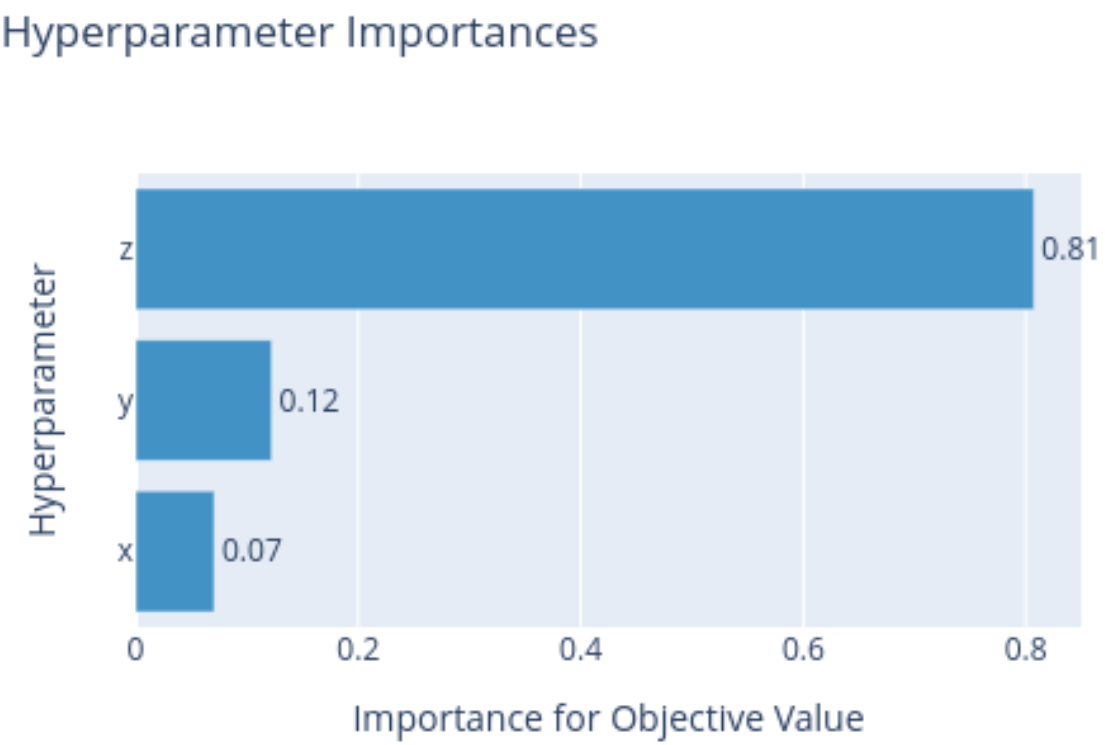
```
1  from Cinema.Prompt import Optimiser
2  class PosOpt(Optimiser):
3      def __init__(self, sim):
4          super().__init__(sim, 1e5, ['minimize'])
5          self.addParameter(['x','y','z'], lower = -5, upper = 5)
6
7      def objective(self, trial):
8          p = self.getParameters(trial)
9          c = 25-(p['x']**2+p['y']**2+p['z']**2)
10         trial.set_user_attr('constraint', (c))
11         if c < 0: # allowed space
12             self.sim.clear()
13             self.sim.makeWorld(p)
14             self.sim.simulate(self.trailNeutronNum)
15             return self.sim.getScorerHist('wl').getWeight().sum()
16         else:
17             return self.trailNeutronNum
18
19 opt = PosOpt(sim)
20 # opt.visInitialGeometry()
21 opt.optimize(name = 'pos_opt', n_trials = 100, localhost=True)
22 opt.analysis()
```

Results: optimised position



- The optimal position is expected at [0,0,-5]
- In 100 iterations, the best trial position is [-1.31, 1.09, -4.85]
- In 1000 iterations, the best trial position is [0.10, 0.16, -5.00]

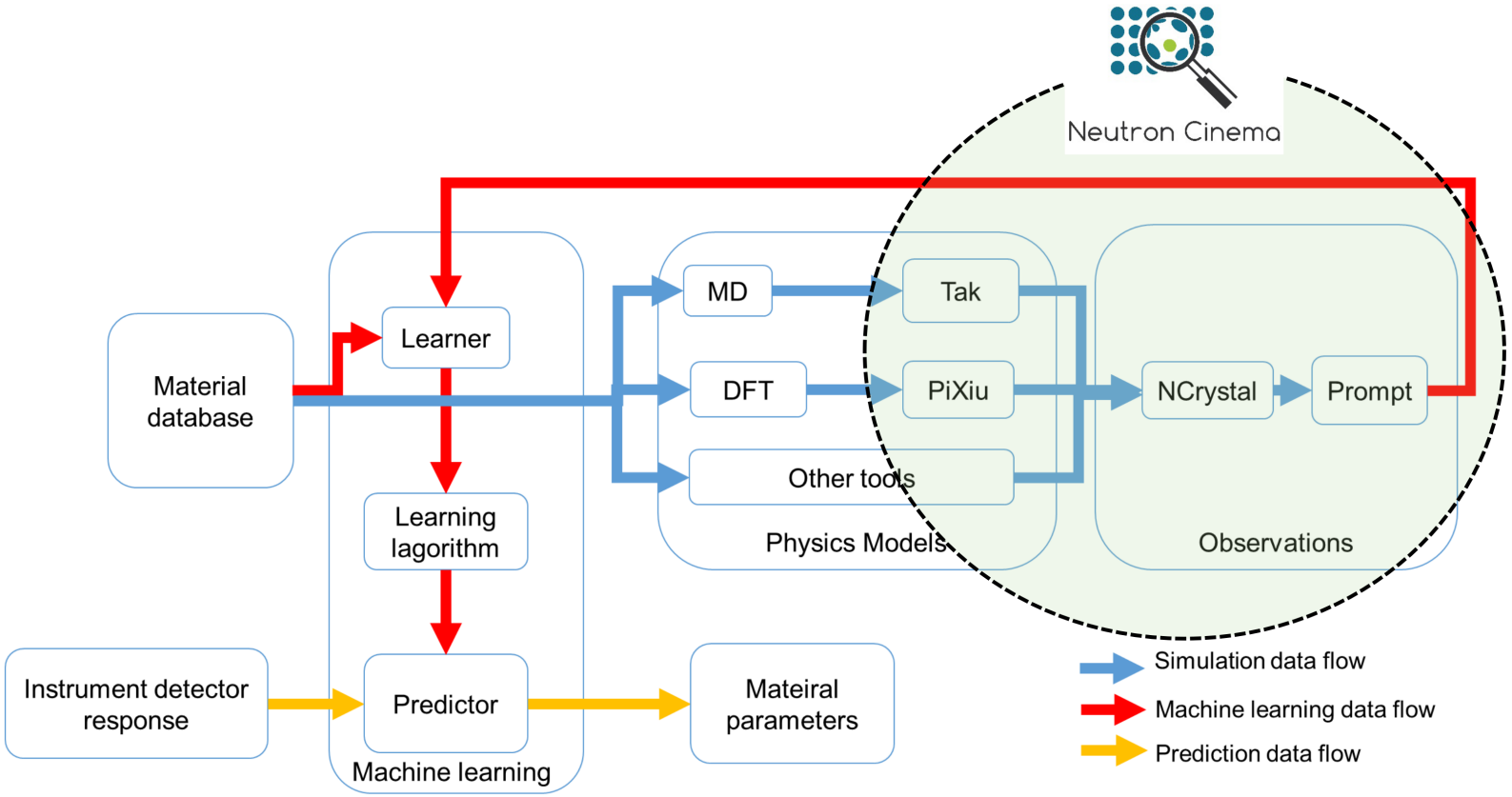
Results: importance analysis



- z position is found to be more important than the others

The long-term system development

- A data production system predicts realistic detector response, e.g. signal+noise+resolution.
- Fill the similarity gap between the simulated and measured raw data.



[1] R. Du, X.-X. Cai. Convolutional discrete fourier transform method for calculating thermal neutron cross section in liquids[J]. Journal of Computational Physics, 2022, 466: 111382

[2] M. H. Mendenhall, R. A. Weller. A probability-conserving cross-section biasing mechanism for variance reduction in monte carlo particle transport calculations[J]. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2012, 667: 38-43

[3] R. Chytracek, J. McCormick, W. Pokorski, et al. Geometry description markup language for physics simulation and analysis applications[J]. IEEE Trans. Nucl. Sci., 2022, 53: 2892-2896

[4] C. B. Sullivan, A. Kaszynski. PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK)[J]. Journal of Open Source Software, 2019, 4(37): 1450

[5] T. Akiba, S. Sano, T. Yanase, et al. Optuna: A next-generation hyperparameter optimization framework[C]. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019,