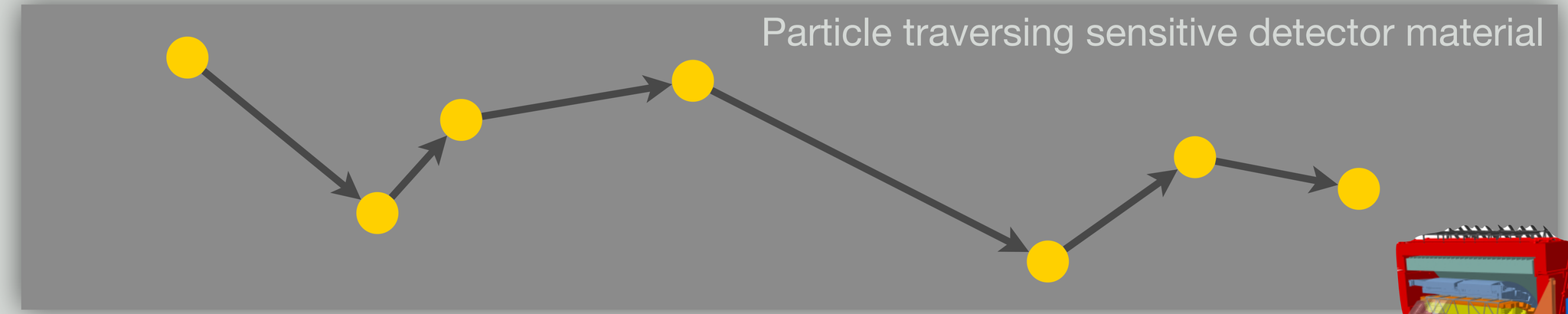
A parameter optimisation toolchain for Monte Carlo detector simulation

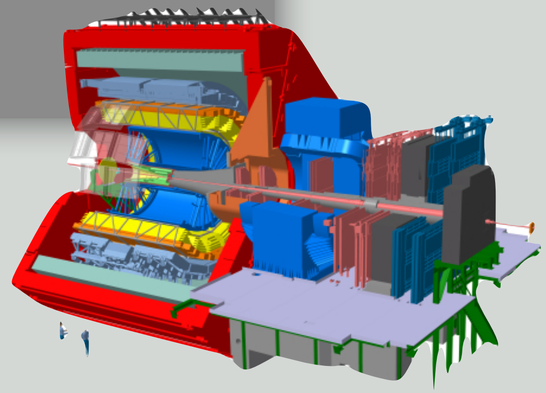
- improving the recipe for full detector simulations -

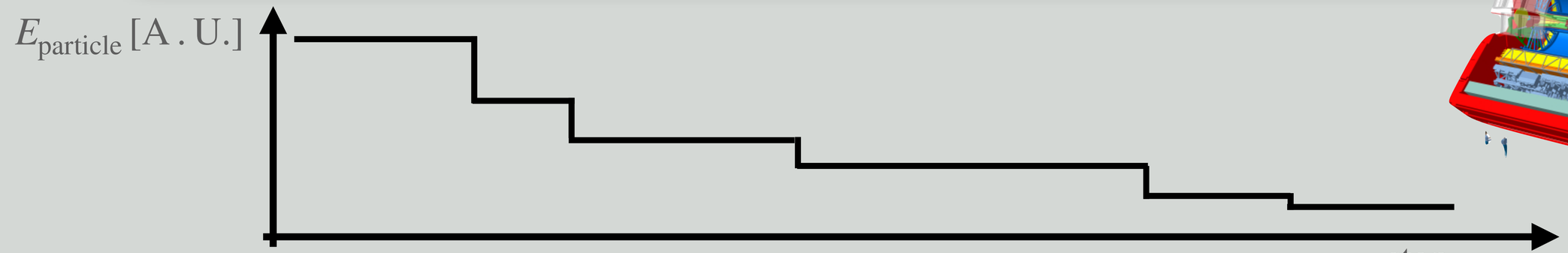
B. Volkel (CERN)
S. Wenzel (CERN), A. Morsch (CERN), M. Concas (CERN)

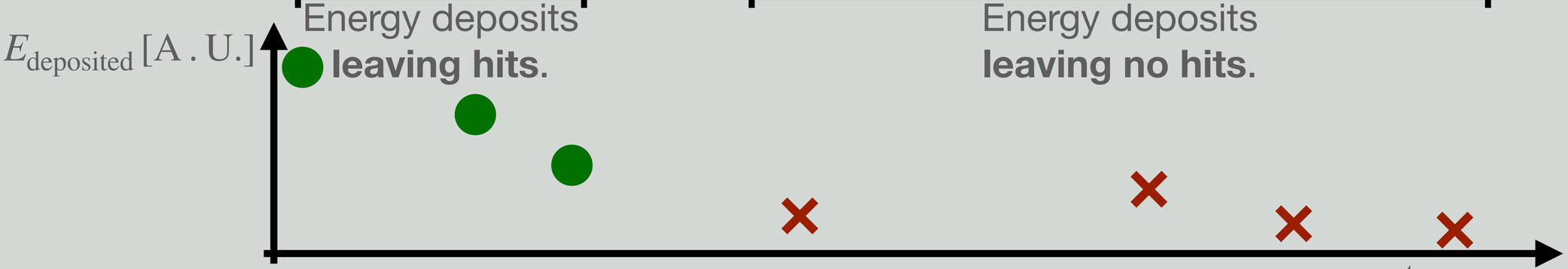
CHEP2023/Track 3 - 09.05.2023

MC transport recipe

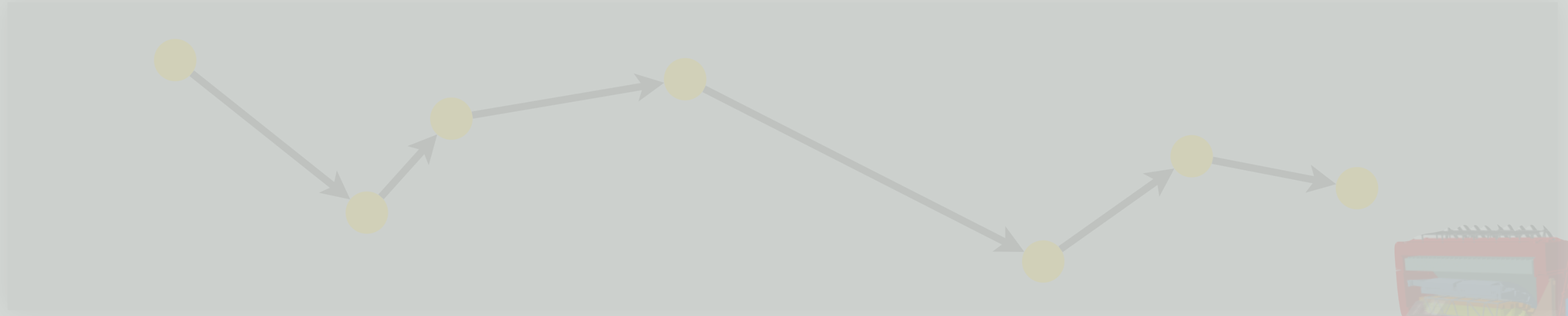


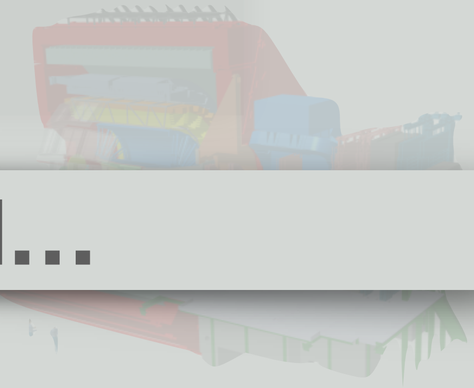




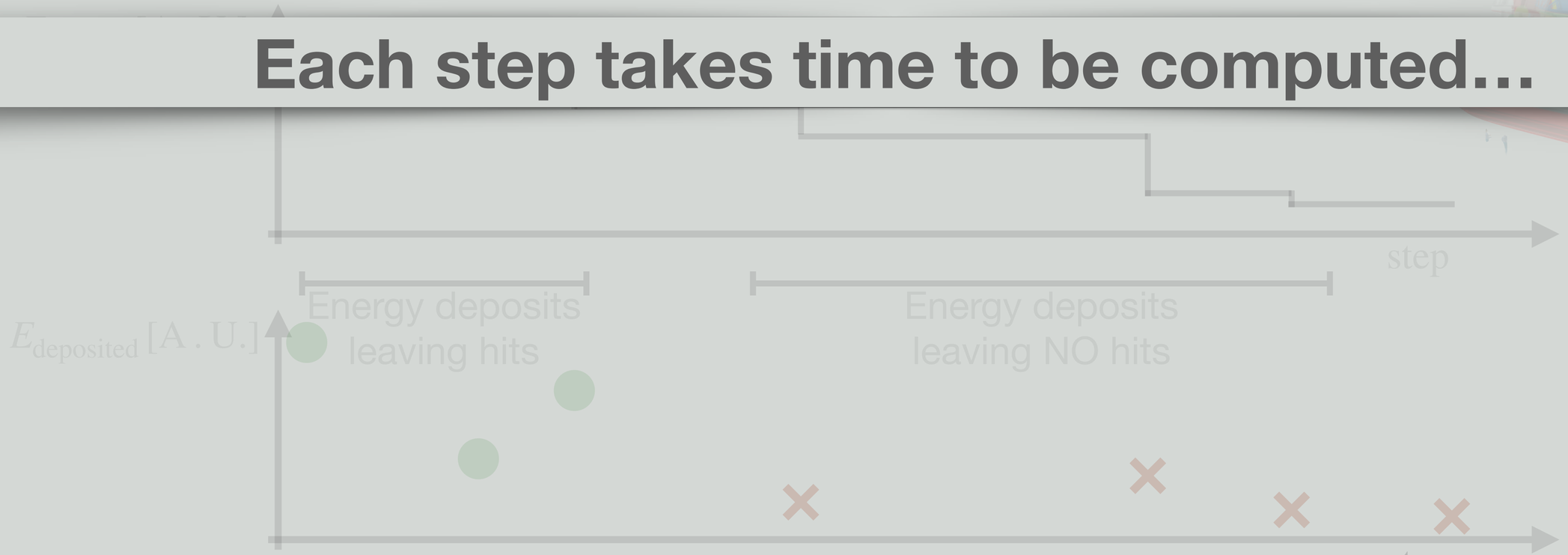


MC transport recipe

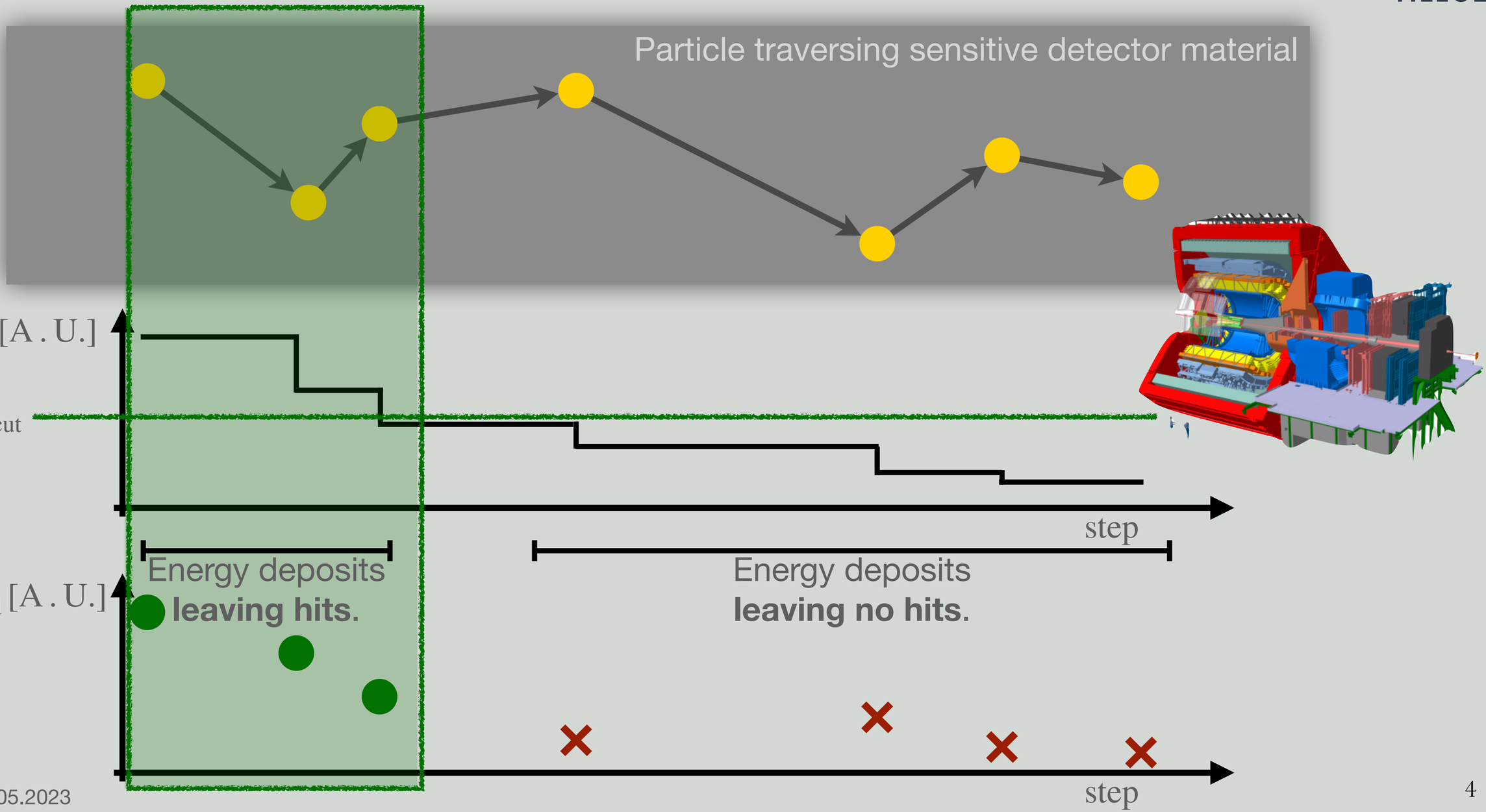




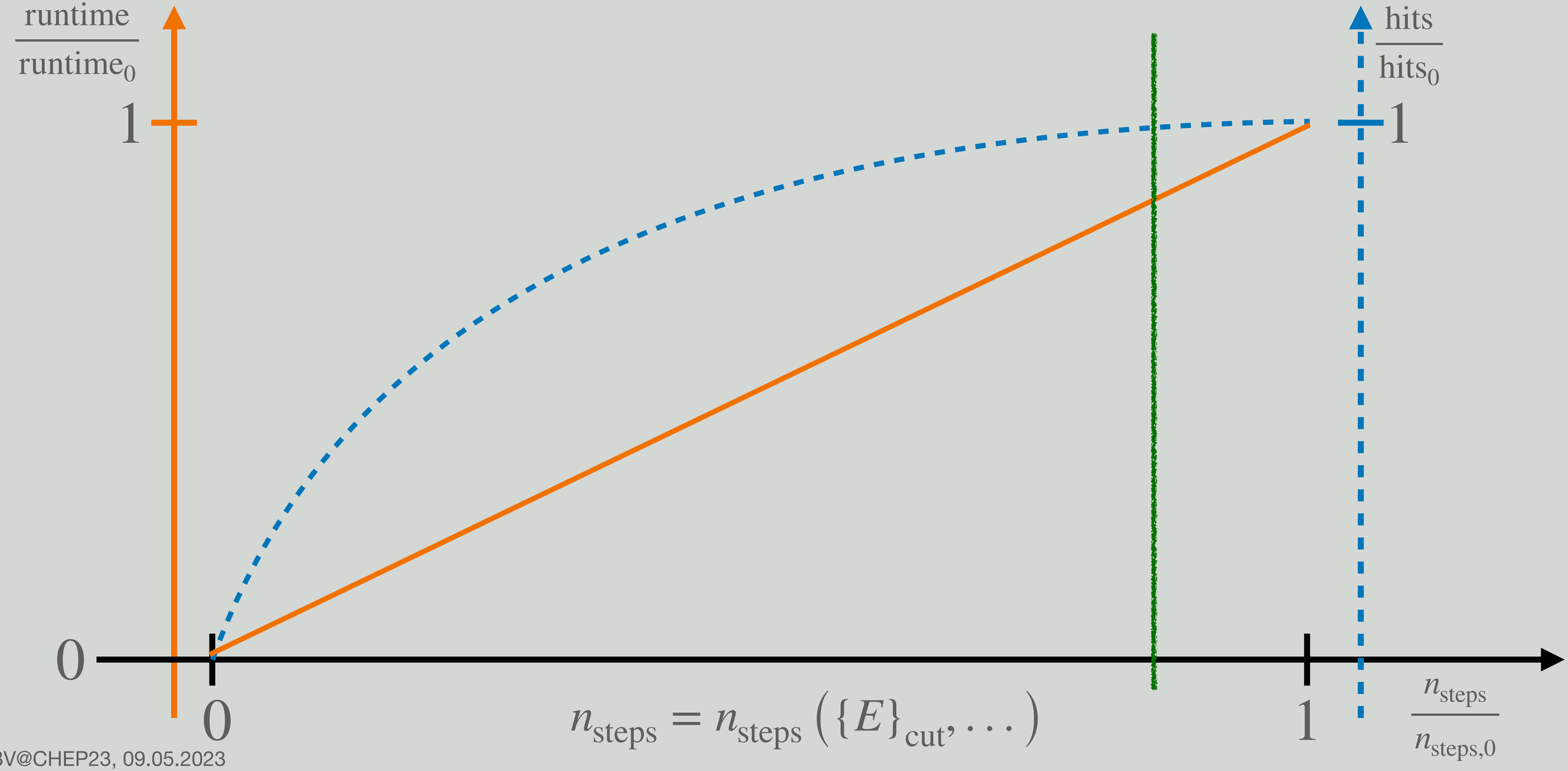
Each step takes time to be computed…



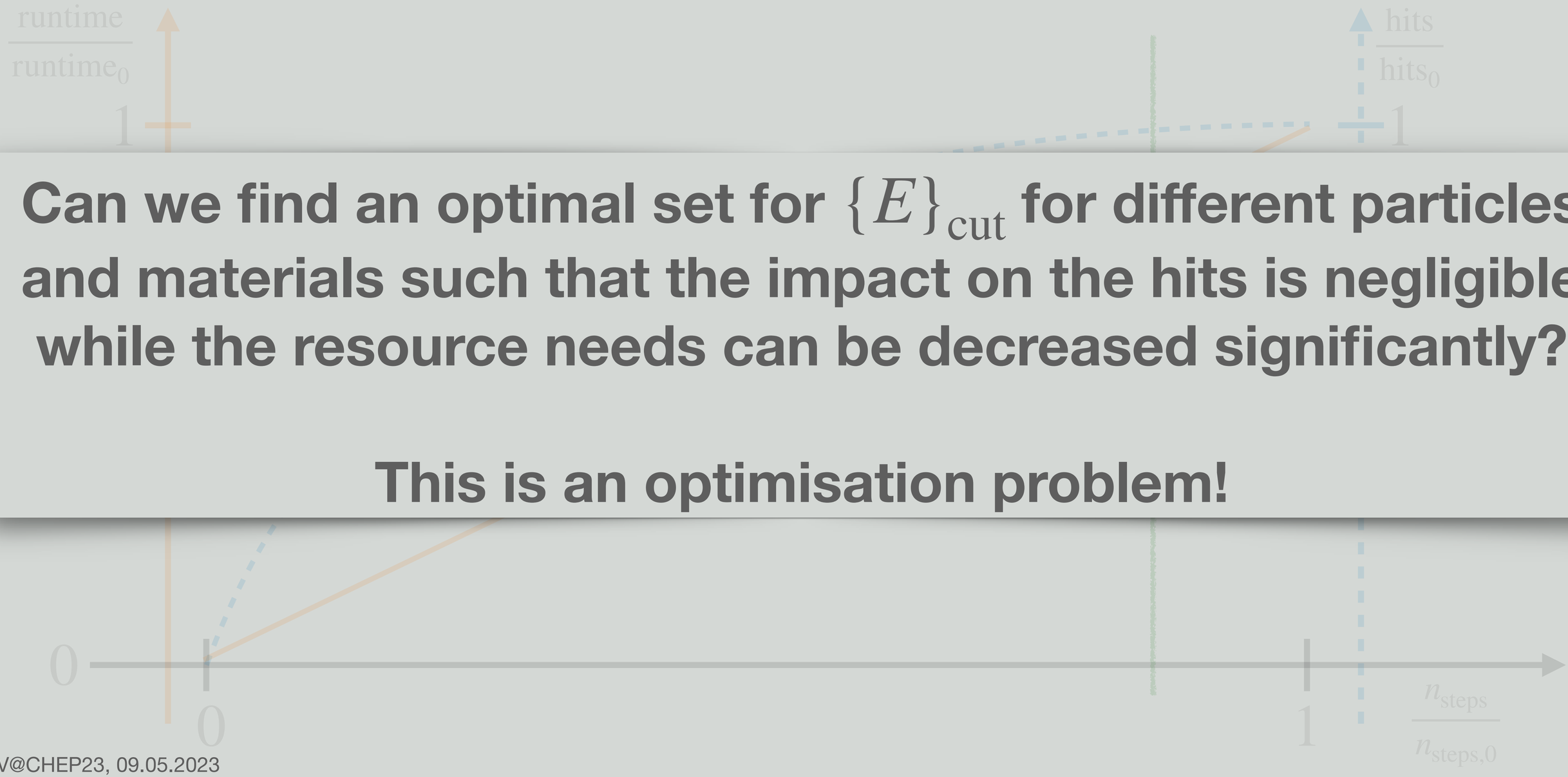
Steps without hits don’t add to the taste…



…so we might speed up the preparation!



…so we might speed up the preparation!



Can we find an optimal set for $\{E\}_{\text{cut}}$ for different particles and materials such that the impact on the hits is negligible while the resource needs can be decreased significantly?

This is an optimisation problem!

Ingredients

*On the shoulders of Virtual Monte Carlo.

100% reproducible simulations	MCStepLogger * MCReplayEngine
Optimisation framework	o2tuner
Validation and closure	ReleaseValidation
Specific definition and configuration of optimisation	o2tunerRecipes

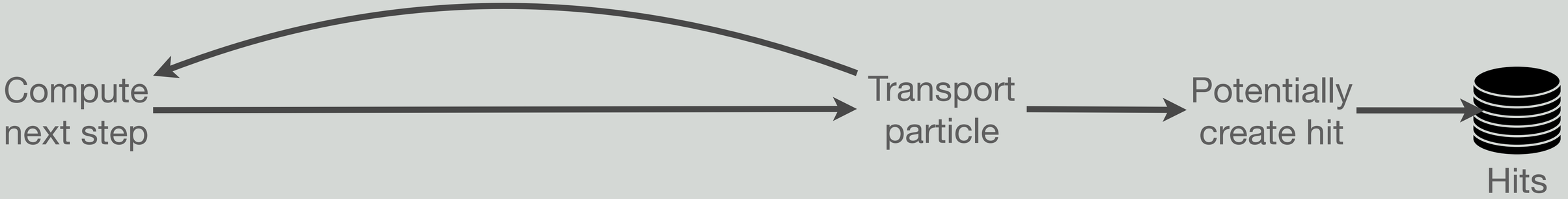
Follow the links to browse the code!

Ingredients - precooked

100% reproducible simulations

MCStepLogger

MCReplayEngine

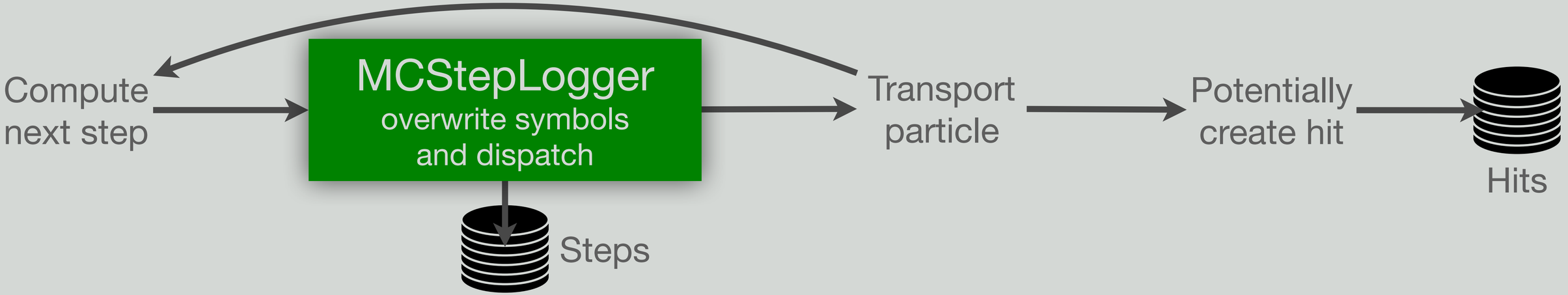


Ingredients - precooked

100% reproducible simulations

MCStepLogger

MCReplayEngine

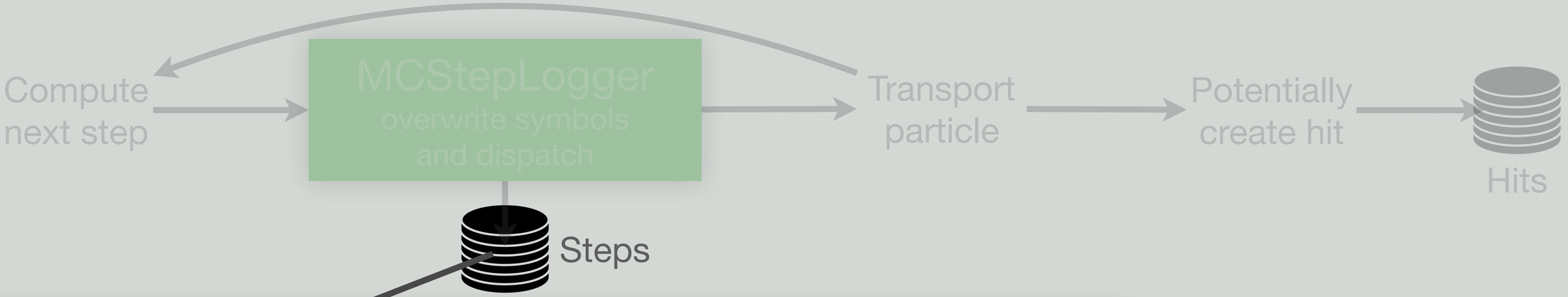


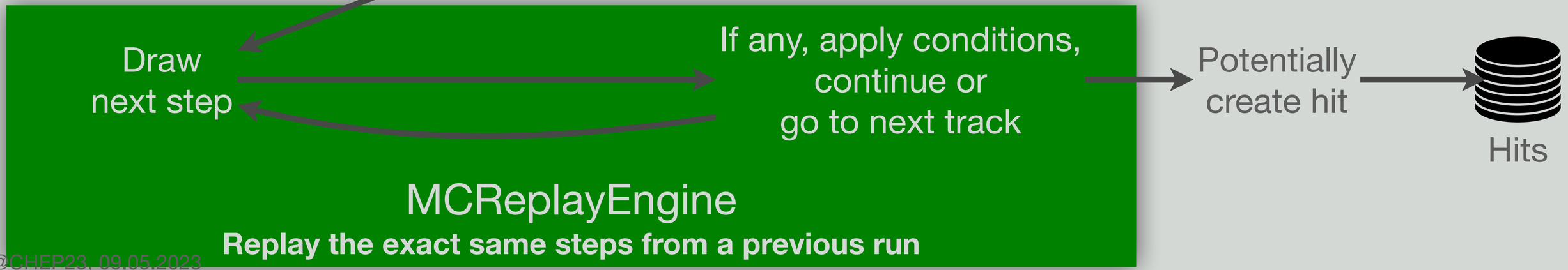
Ingredients - developed for this toolchain

100% reproducible simulations

MCStepLogger

MCReplayEngine





Ingredients - developed for this toolchain

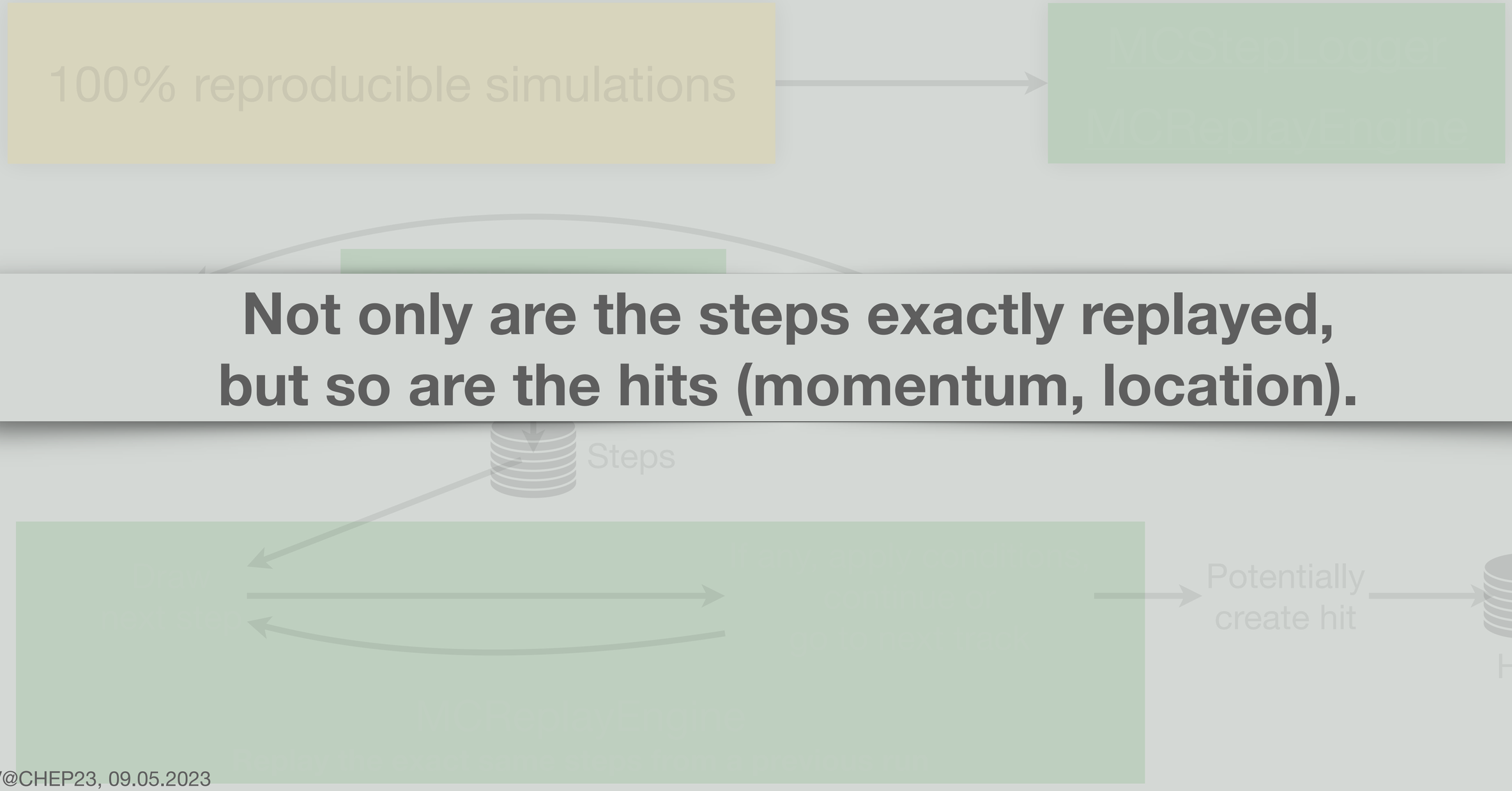
100% reproducible simulations

We can EXACTLY replay
a previously recorded full simulation!
And we can do so over and over again;
the “simulation” (no actual computation) is much faster.

Potentially create hit

Hits

Ingredients - developed for this toolchain



**Not only are the steps exactly replayed,
but so are the hits (momentum, location).**

Ingredients - precooked

- Execute user-defined recipe
  - Work through necessary steps (e.g. data preparation, optimisation, evaluation)
  - Ensure reproducibility
  - Wrap parallelisation
  - **Optimise** various applications such as ML models or **parameter-dependent executables and scripts**

Ingredients - precooked

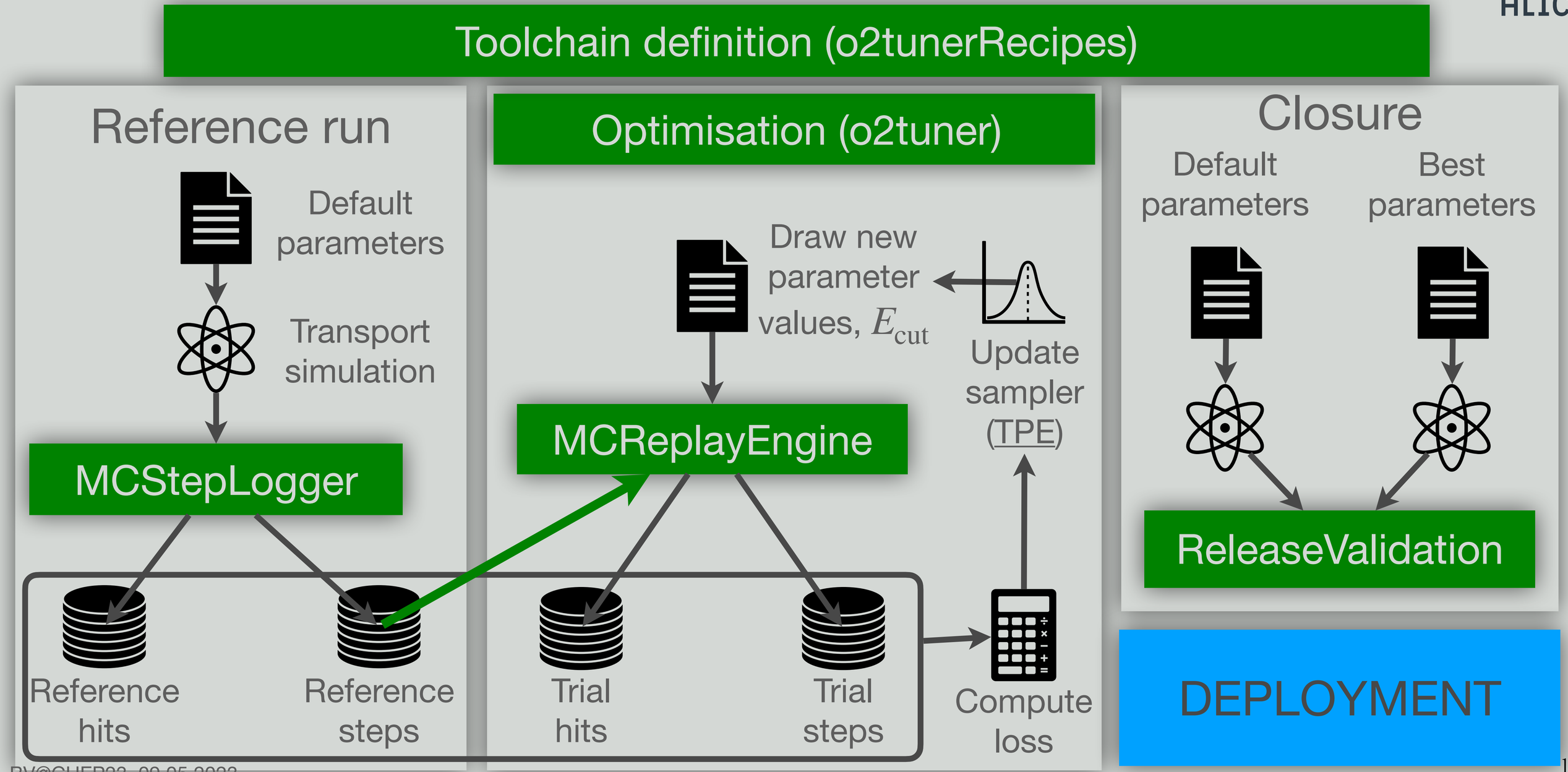
- Automatically
  - Compare various different observables
  - Compute various different metrics
  - Plot and report summaries

Ingredients - prepared by the user

Specific definition and configuration of optimisation → o2tunerRecipes

- Toolchain definition
  - Objective function and loss [required]
  - Functions for (reference) data creation [optional]
  - User-defined evaluation and closure tests [optional]
  - Optimisation configuration [optional]
    [number of trials, jobs, parameter sampling etc.]

Preparation



Prepare the loss

Steps themselves are **invisible** to the detectors. They are the results of MC calculations.

Hits are the physical energy deposits a detector **can see**. This is what we need to keep.

Keep hits while dropping steps! Defining the loss.

Prepare the loss

Steps ⟶ Hits

$$\frac{\partial L}{\partial s_{\mathrm{t}}} > 0 \qquad \frac{\partial L}{\partial h_{\mathrm{t}}^{\mathrm{d}}} < 0$$

Define a metric to be evaluated for each chosen set of drawn values.

$$L_{\mathrm{t}}\left(s_{\mathrm{ref}}, \{h\}_{\mathrm{ref}}^{\mathrm{d}}, s_{\mathrm{t}}, \{h\}_{\mathrm{t}}^{\mathrm{d}}\right) =$$

Optimisation trial t,
Number of steps s,
Detector d,
Hits h^{d} in detector,
detector-specific penalty α^{d}.

Prepare the loss

Steps ⟶ Hits

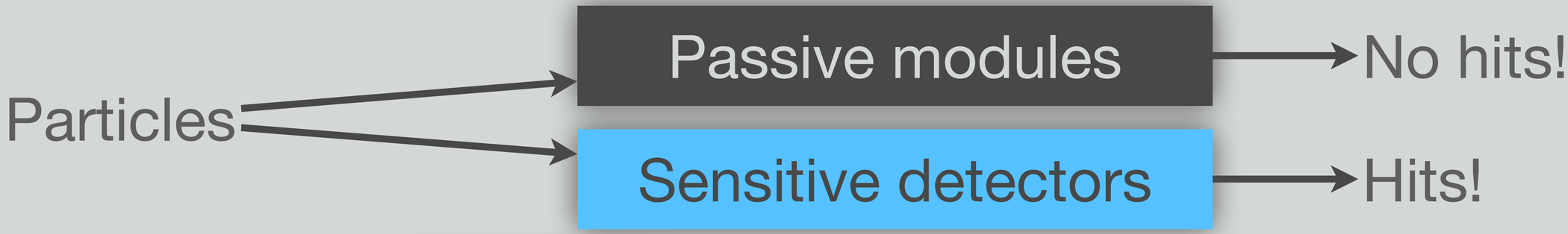
$$\frac{\partial L}{\partial s_\mathrm{t}} > 0 \qquad \frac{\partial L}{\partial h_\mathrm{t}^\mathrm{d}} < 0$$

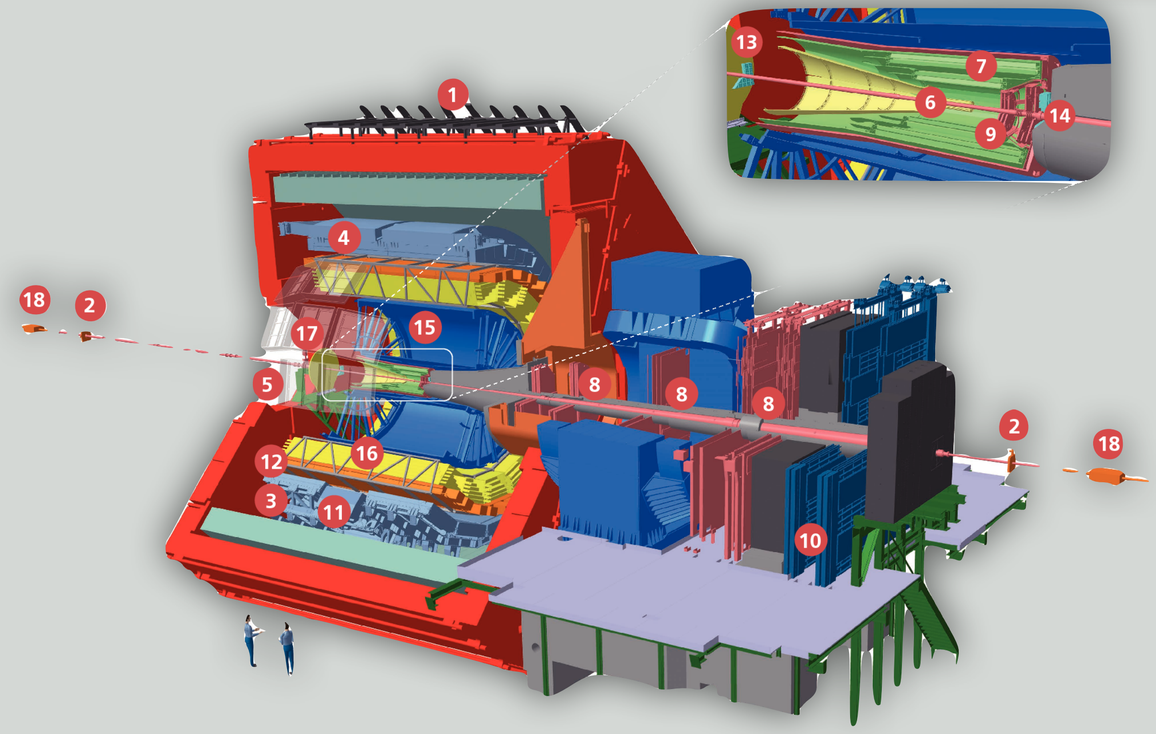
Define a metric to be evaluated for each chosen set of drawn values.

$$L_\mathrm{t}\left(s_\mathrm{ref}, \{h\}_\mathrm{ref}^\mathrm{d}, s_\mathrm{t}, \{h\}_\mathrm{t}^\mathrm{d}\right) = \frac{s_\mathrm{t}}{s_\mathrm{ref}} + \frac{1}{N^\mathrm{det}} \sum_{\mathrm{d}\in\mathrm{det}} \alpha^\mathrm{d}\left(h_\mathrm{t}^\mathrm{d}, h_\mathrm{ref}^\mathrm{d}\right) \cdot \left[1 - \frac{h_\mathrm{t}^\mathrm{d}}{h_\mathrm{ref}^\mathrm{d}}\right]$$

Optimisation trial t,
Number of steps s,
Detector d,
Hits h^d in detector,
detector-specific penalty α^d.

Final thoughts before optimising





We first target the reduction **of steps in passive modules: only leave what will eventually be seen by detectors.**

Show results for beam pipe.

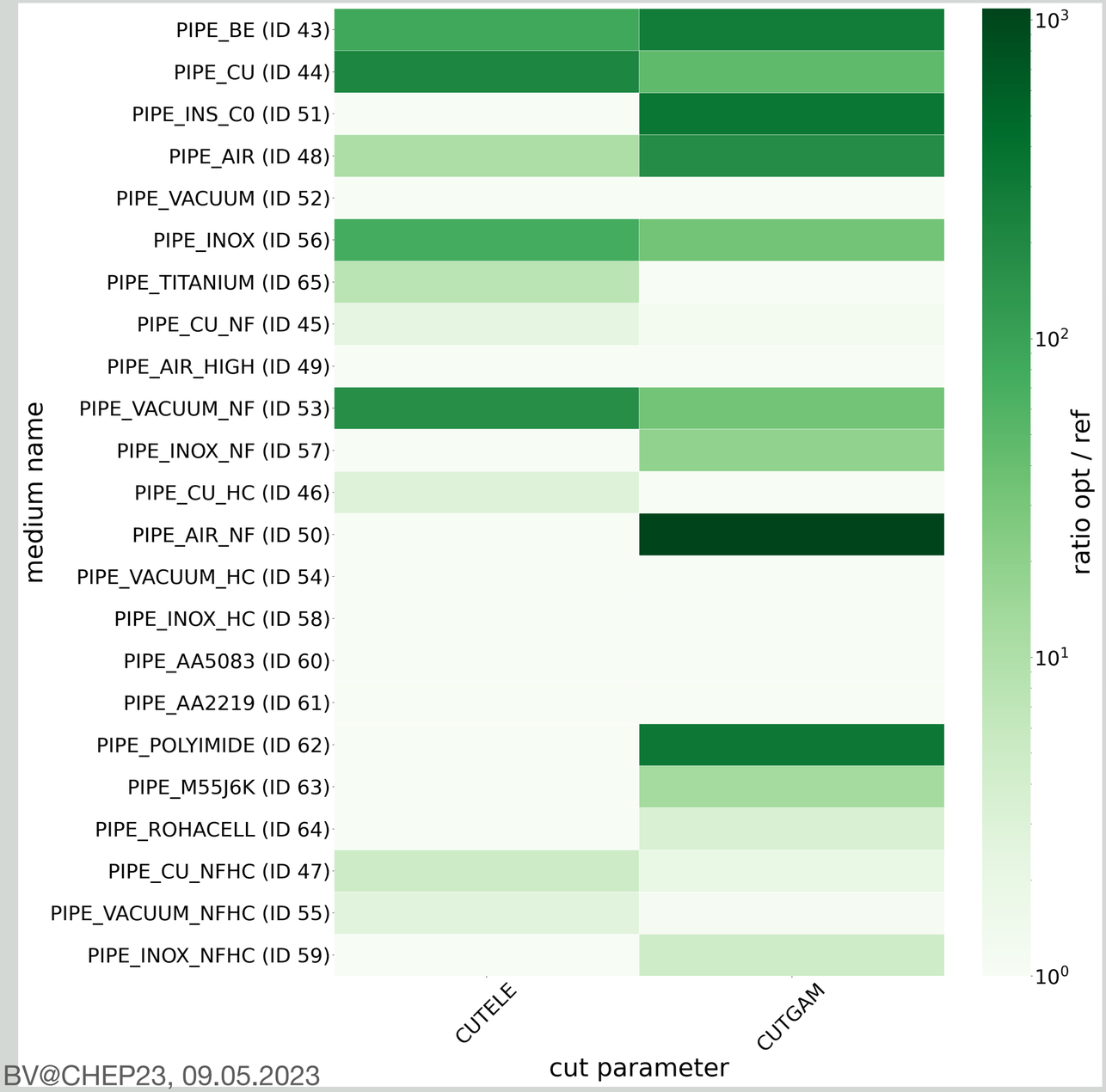
Account for hits in all detectors.

Optimisation…

boiling, waiting, stirring…

Tasting!

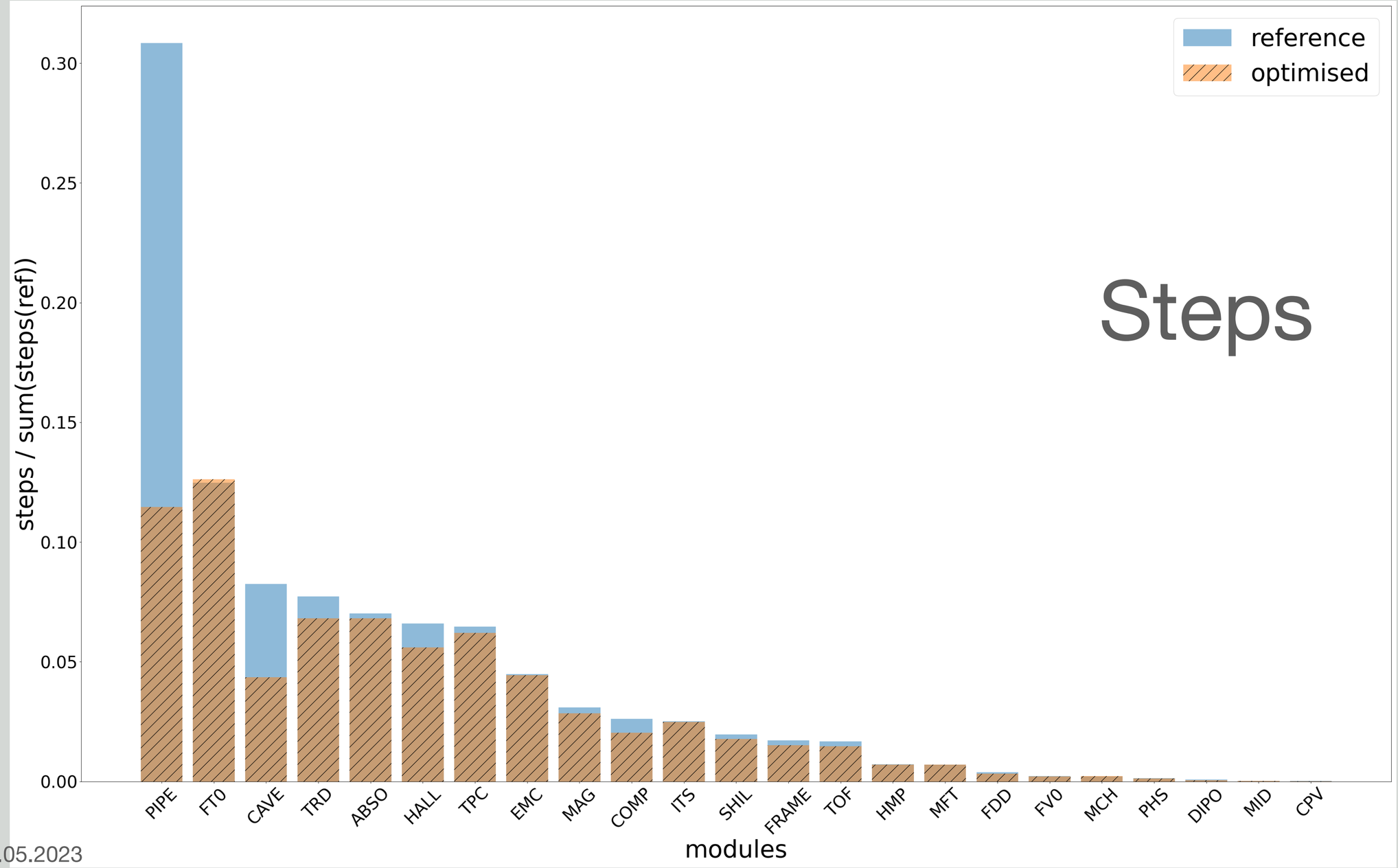
Optimised values of electromagnetic cut parameters



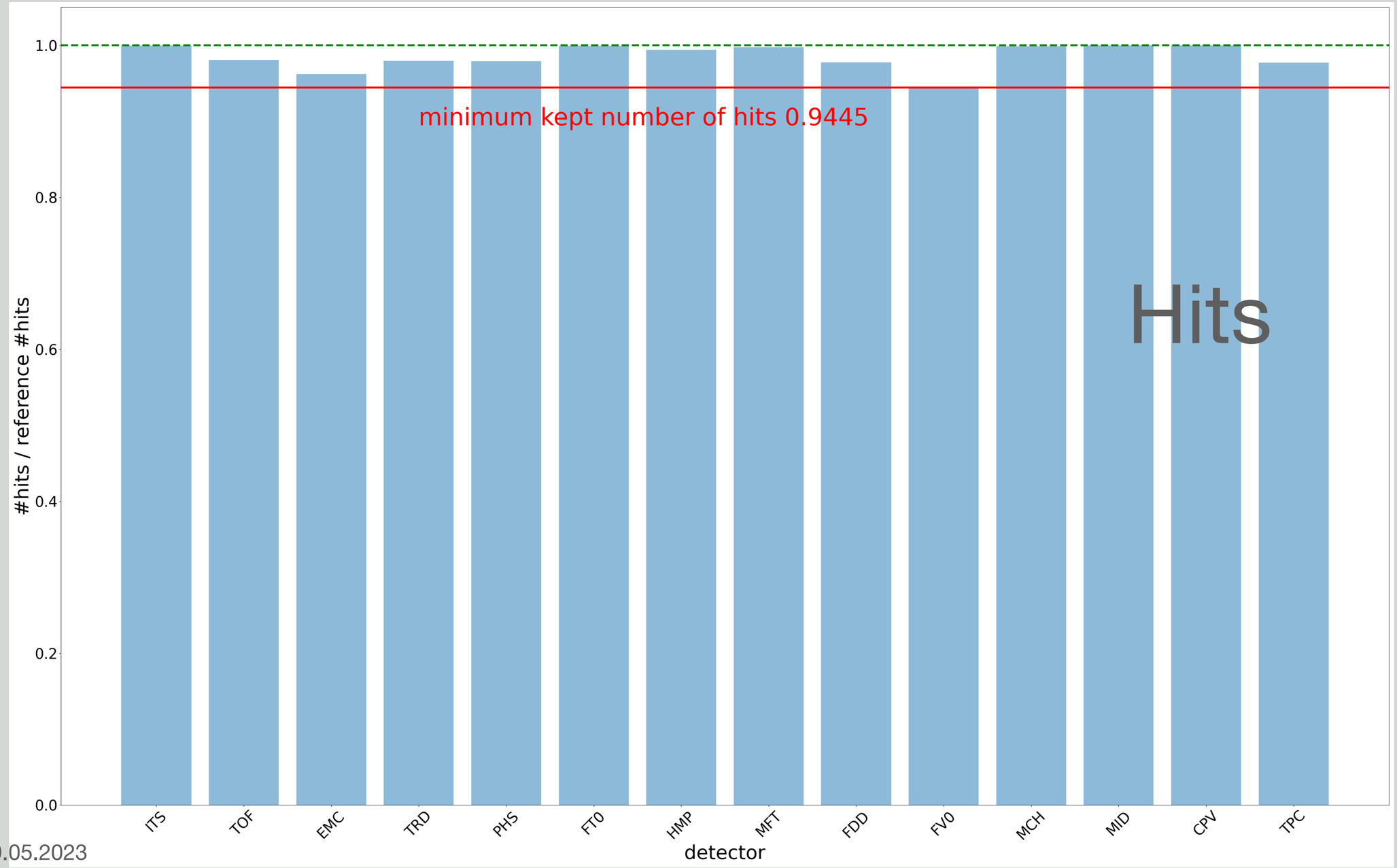
Many parameters changed by 1-3 orders of magnitude.

In this case (electromagnetic), cutting on electrons(photons) has a strong impact on the abundance of photons(electrons).

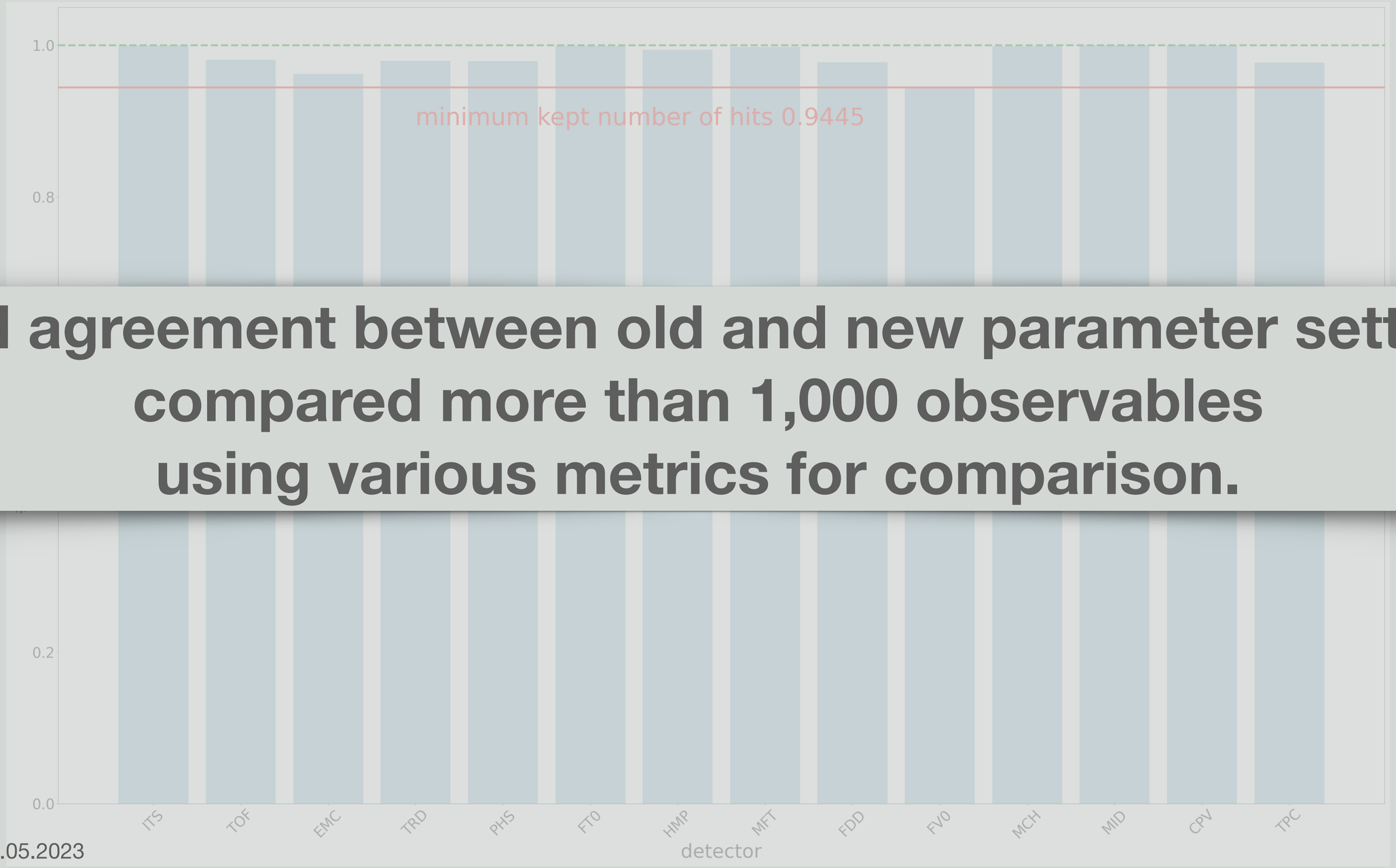
Removed more than 20% of all steps…



…while keeping almost all hits



…while keeping almost all hits



Good agreement between old and new parameter settings, compared more than 1,000 observables using various metrics for comparison.

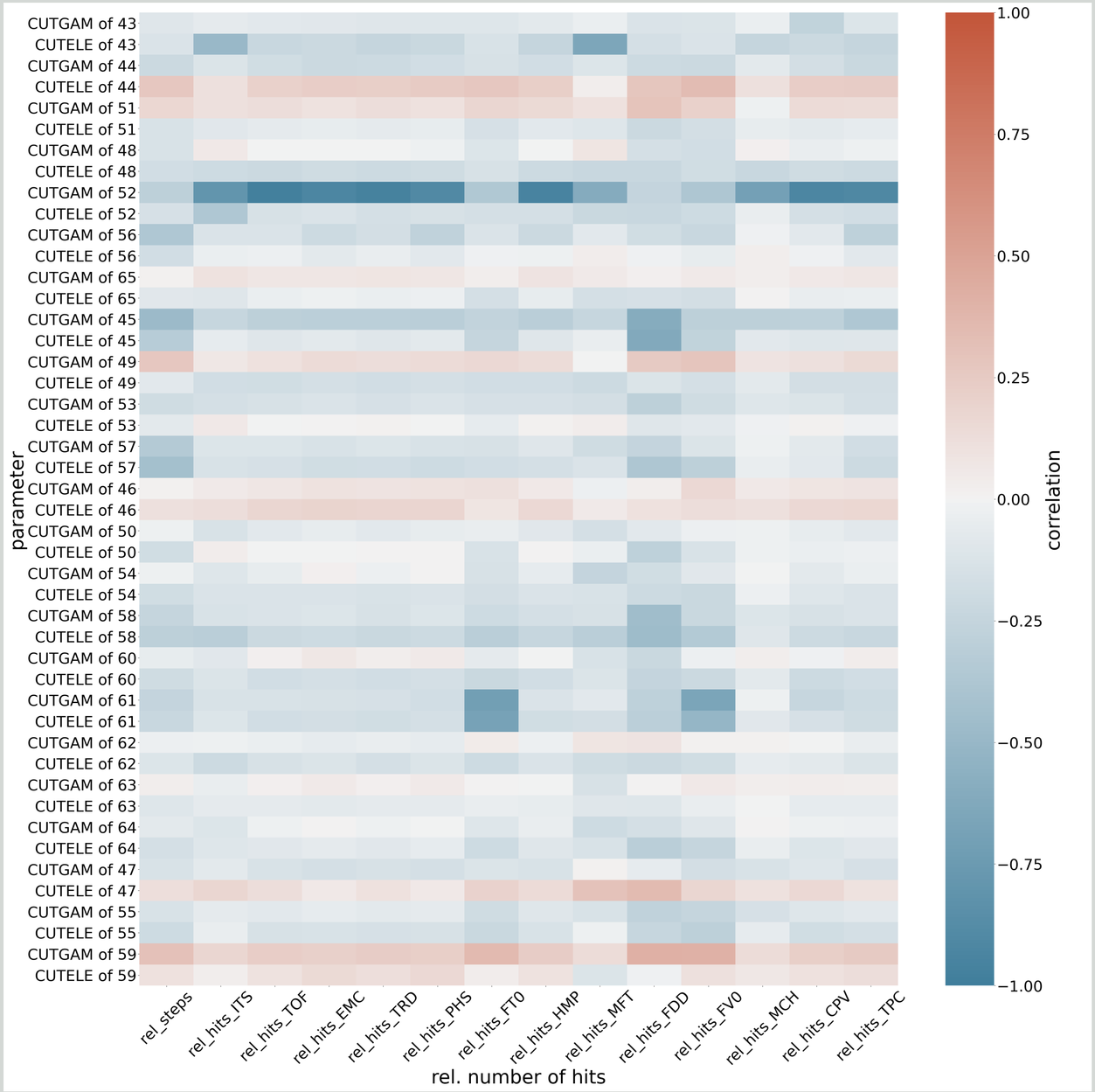
Bon Appetit!

- **First time we have a fully automatic parameter optimisation toolchain for MC detector simulation.**
- Gain 20% speed-up of full-simulation (only beam pipe, only electromagnetic parameters).
- We will target other geometry or phase-space related parameters in such optimisations.
  - Push for including all passive modules.
    Preliminary: Gain between 30% and 40%.
  - After that, target additional parameters.
- Not only applicable to transport simulation but also to other parts of the simulation chain.
  [Digitisation, reconstruction, tracking…]
- Our studies teach us more about our detector as well as our simulation algorithms.
  [Provide valuable insight towards potential fast-simulation approaches.]

Thank you very much,
enjoy dinner,
and have a great conference!

BACKUP

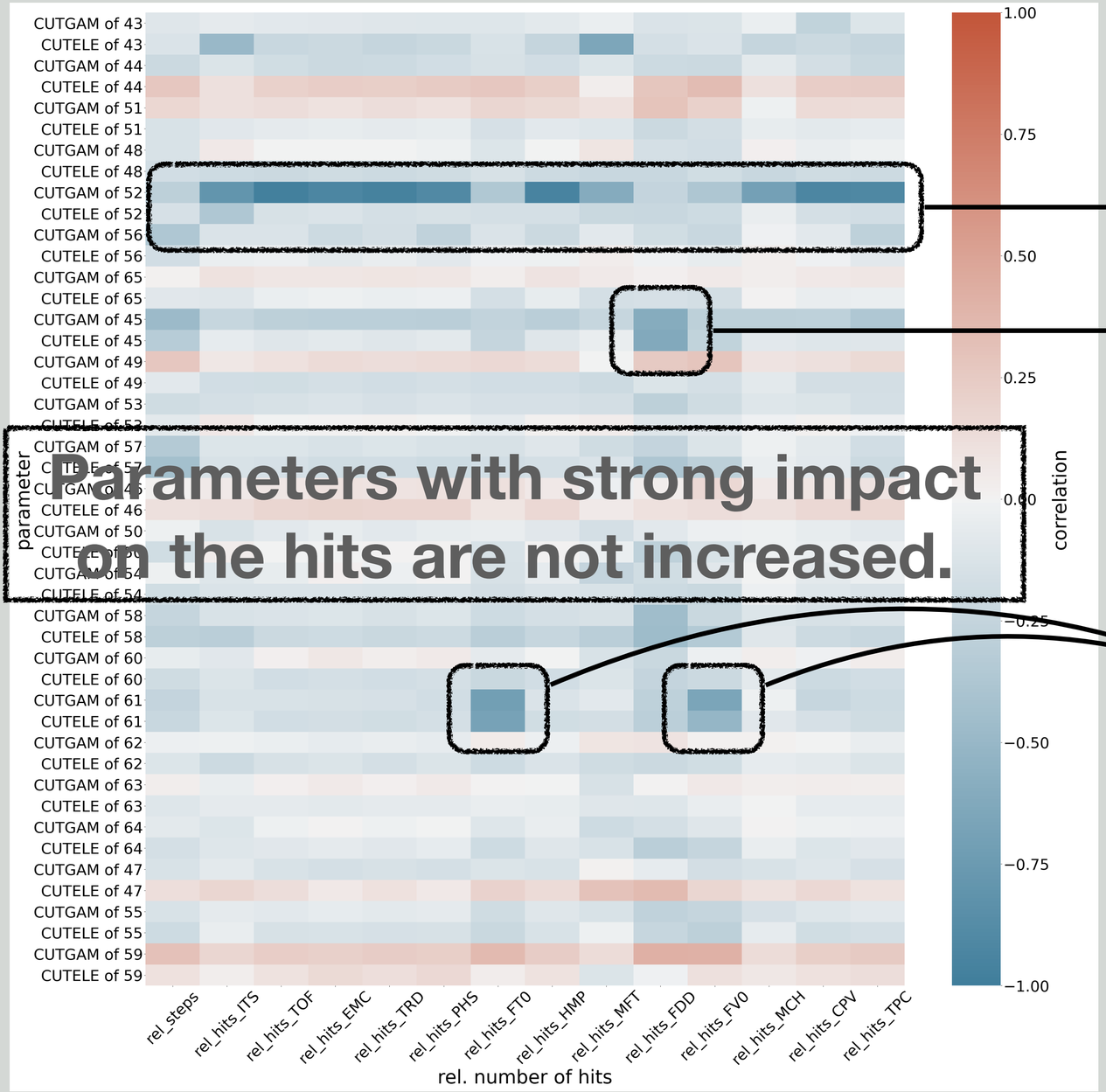
Parameters and hits go well together



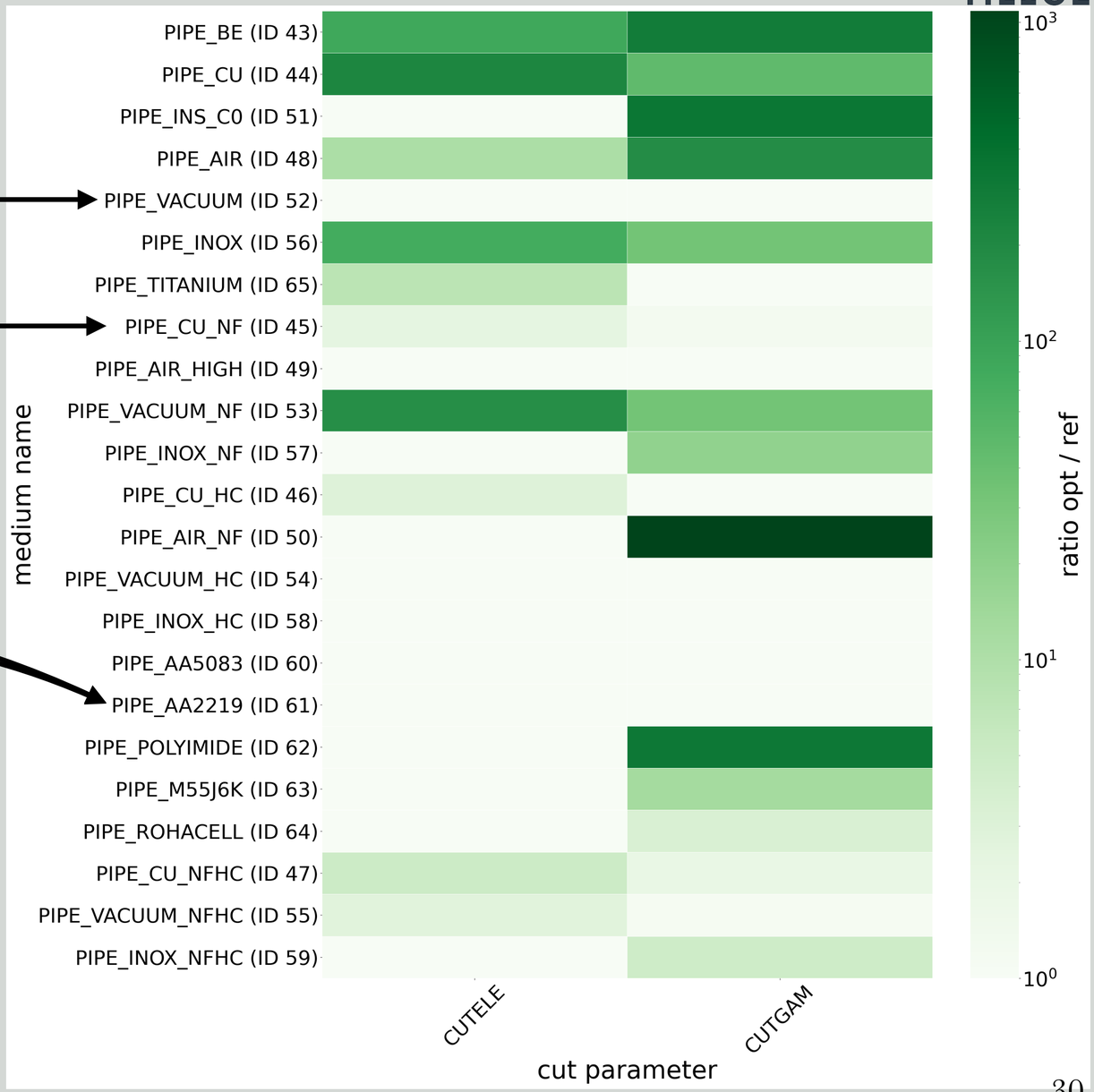
Correlation of number of kept hits with chosen cut value across all optimisation trials.

- A negative correlation basically means higher/lower cut ~ less/more hits.
- Intuitively, a positive correlation physically does not make sense. However, we are only looking at a “first order” relation between the hits and cuts. The choice of cut values is more complex!

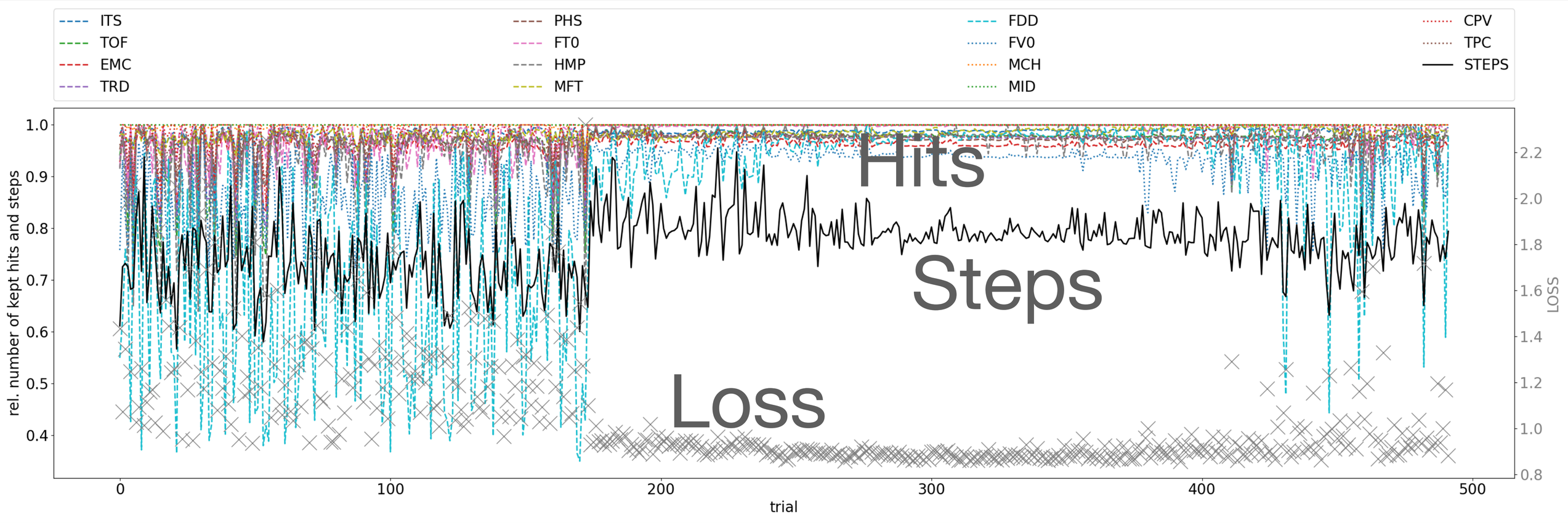
Parameters and hits go really well together



Parameters with strong impact on the hits are not increased.



Evolution of steps and hits during optimisation



“Warm-up” (drawing random parameters) for 150 trials. | **Optimising** | **Exhausted**

Example to work with o2tuner

config.yaml

```yaml
stages_user:
  hello:
    cmd: "echo Hello"

  evaluate:
      optimisations:
        - optimisation
      python:
        file: evaluate.py
        entrypoint: evaluate

stages_optimisation:
  optimisation:
    config:
      some_key: some_value
    file: optimise.py
    objective: objective
    deps:
      - hello
```

evaluate.py

```python
"""
To test the full o2tuner chain
"""


def evaluate(inspectors, config):
    """
    A dummy objective
    """
    # in this example we know that we have an inspector, otherwise we should check
    inspector = inspectors[0]
    print(inspector.get_losses())
    print(config)
    return True
```

optimise.py

```python


def objective(trial, config):
    """
    A dummy objective
    """
    x = trial.suggest_float("x", -10, 10)
    y = trial.suggest_float("y", -10, 10)
    annotate_trial(trial, "sum", x + y)
    annotate_trial(trial, "some_key", config["some_key"])
    return (x - 2)**2 + (y - 3)**2
```

```
o2tuner -w </my/work/dir> -c config.yaml [-s optimisation]
```