On Estimating Gradients of Discrete Stochastic Programs for Detector Design Optimization

CHEP 2023

Lukas Heinrich, TUM
Michael Kagan SLAC

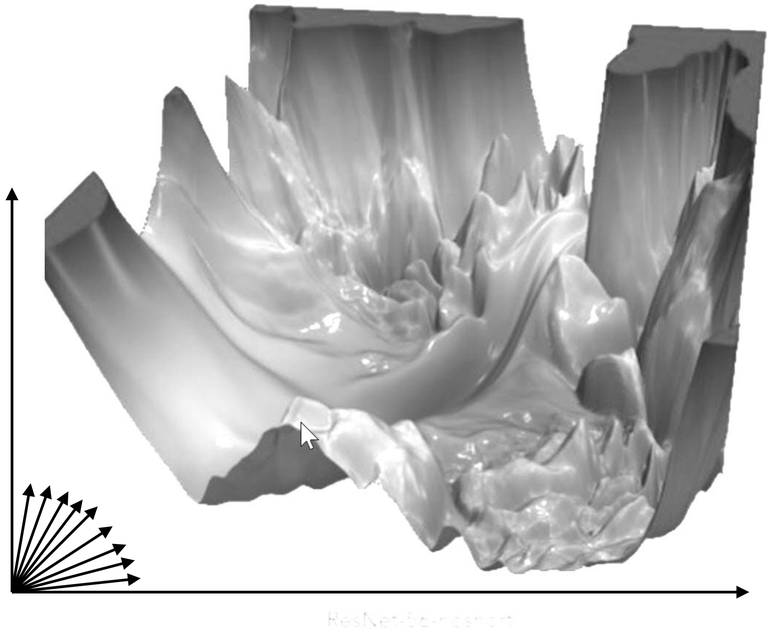
Technische Universität München

ODSL
ORIGINS Data Science Lab

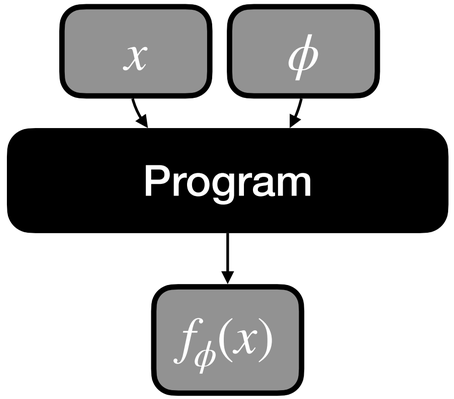
Differentiable Programming

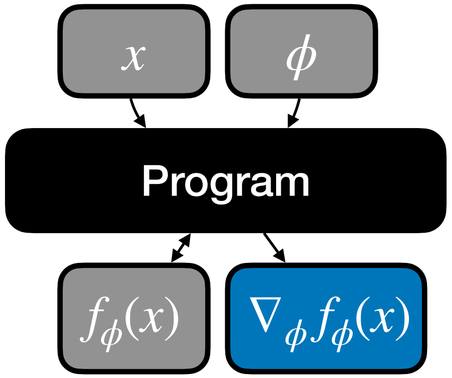
A major enabler of large-scale Machine Learning:

Ability to compute gradients of numeric programs



To deal with hyper-planes in a 14-dimensional space,
visualize a 3D space and say 'fourteen' to yourself very loudly.
-G. Hinton





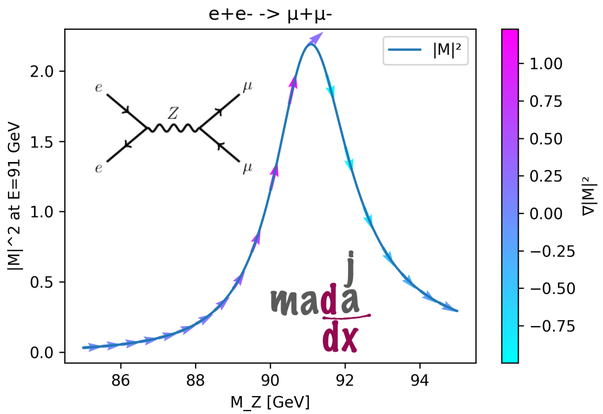
TensorFlow

PYTORCH

Gradients Beyond ML

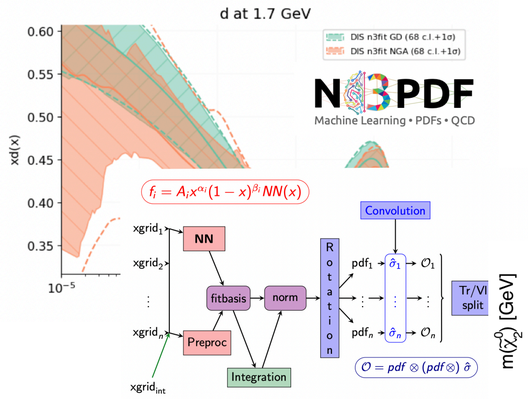
Interest in applying DP to programs that are not NNs:

powerful method to combined physics + ML



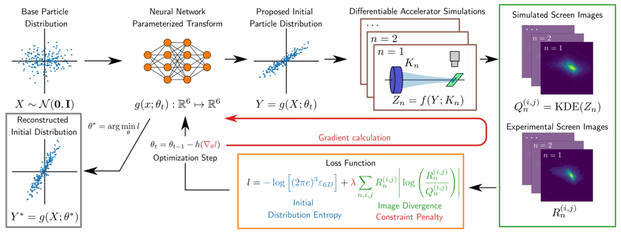
Differentiable Matrix Elements

2203.00057



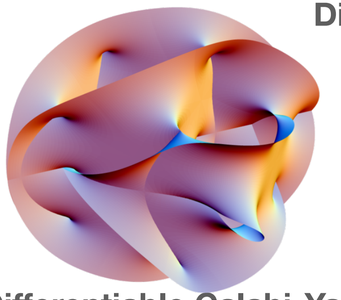
Differentiable Proton Structure

1907.05075



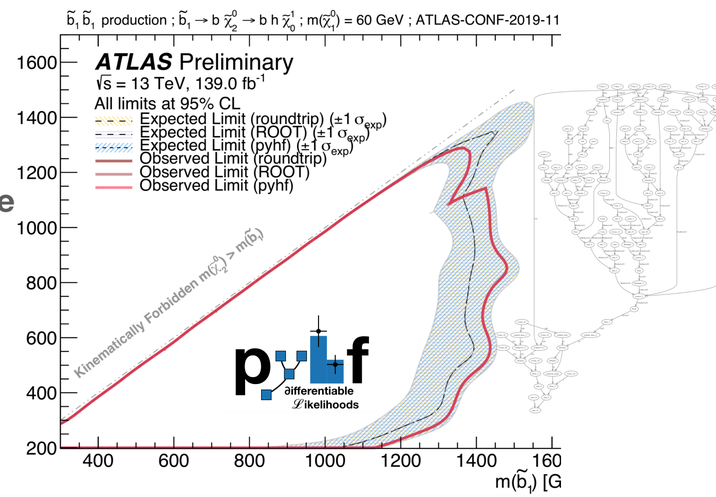
Differentiable Accelerator Simulation

2211.09077



Differentiable Calabi-Yau Manifolds

2211.12520



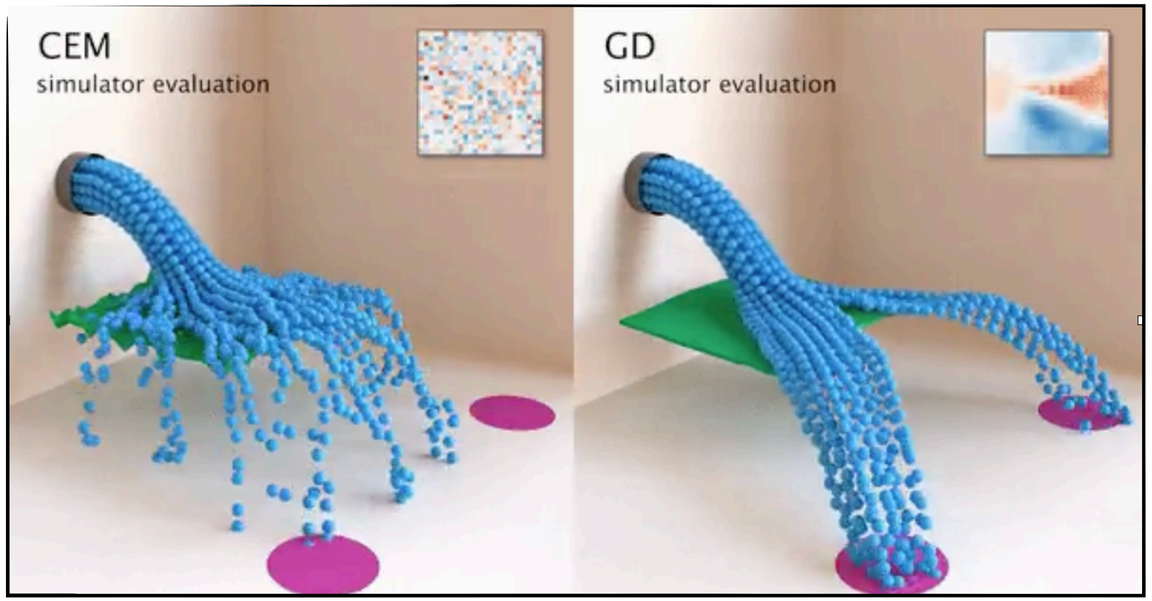
Differentiable Inference

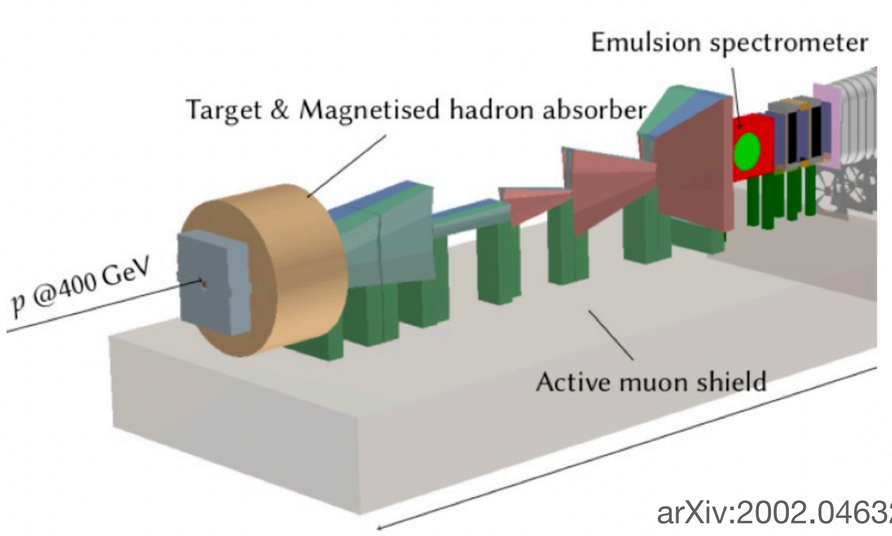
10.21105/joss.02823

Emerging Usecase: Design Optimization

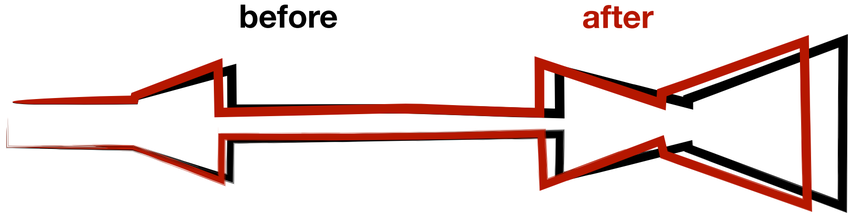
One of the most consequential high-dim. optimizations:

find a good detector design



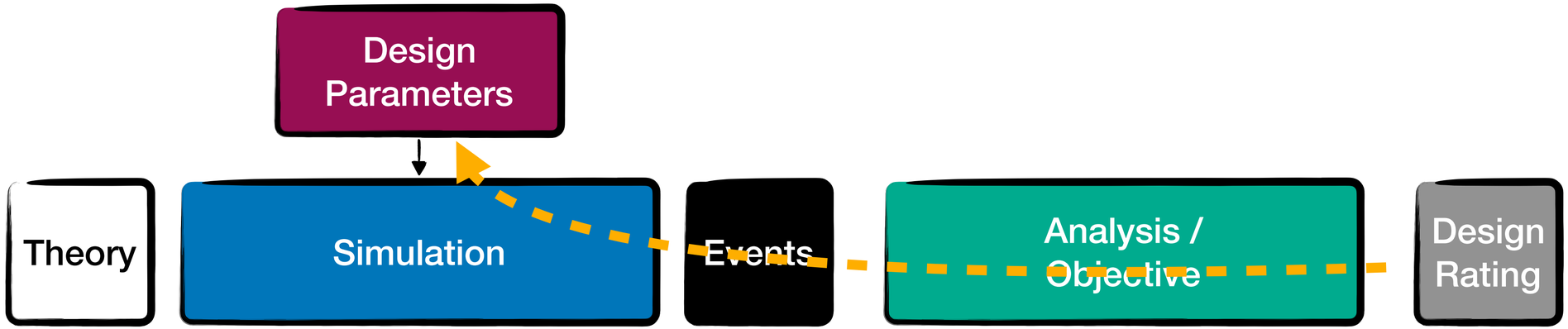


arXiv:2002.04632



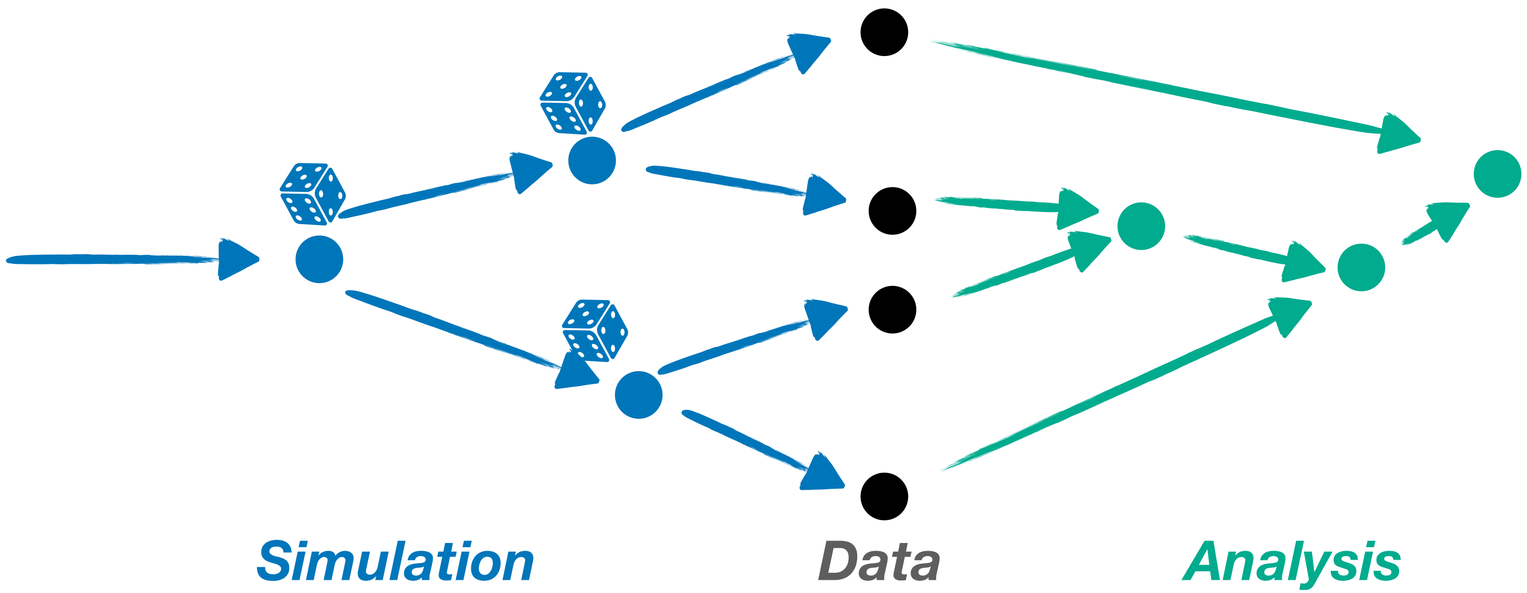
A Requirement:

For a gradient-based design optimization, one needs both:
a *differentiable simulation and reco/analysis*



A Small Problem

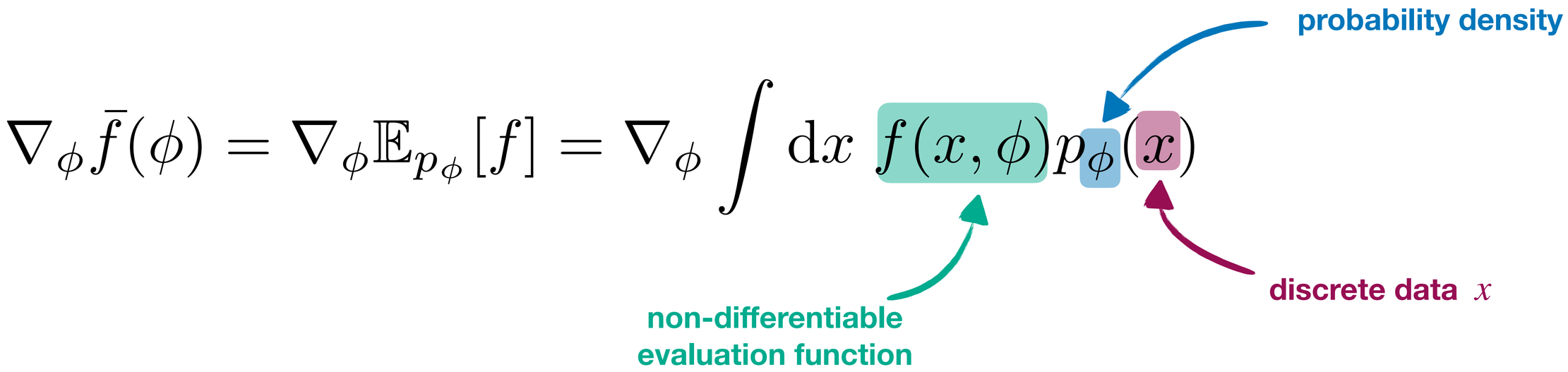
To first order, most of simulation is **stochastic branching**, while most of analysis is **iterative clustering** with many discrete choices



HEP: fundamentally incompatible with differentiable programming?

Not necessarily

The gradients we need are gradients of **expectation values** over probability distributions & observables with discrete aspects



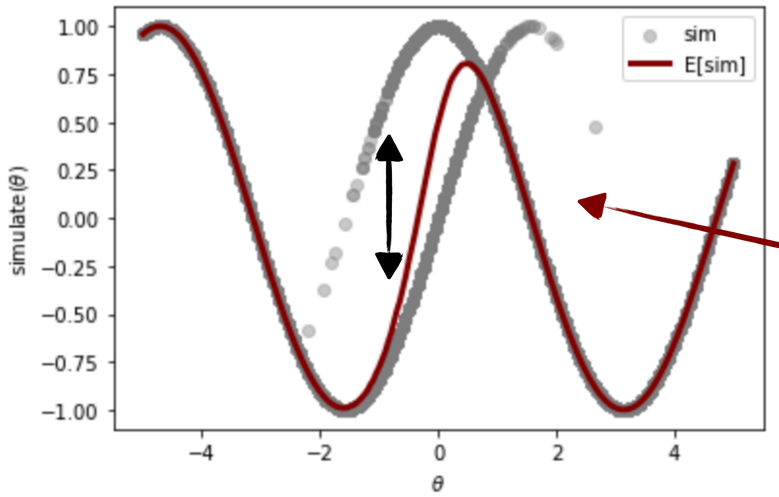
In many cases, the expectation value *<u>is</u>* differentiable

(just not via standard automatic differentiation)

Simple Examples

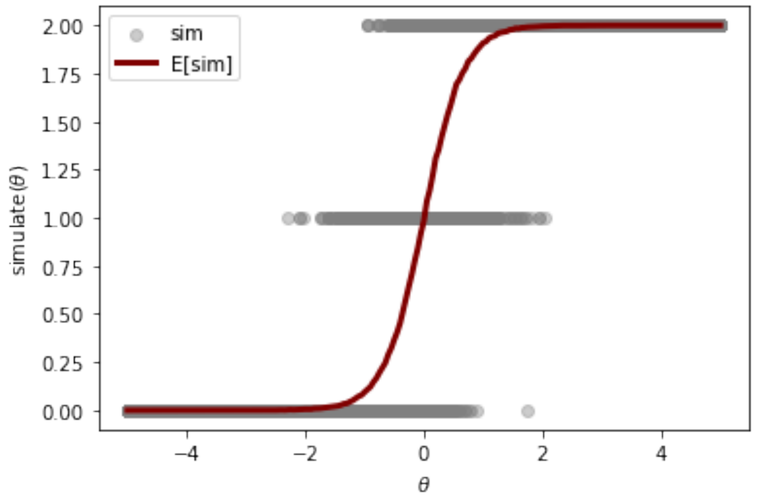
```python
def simulate(theta):
    p = sigmoid(theta)
    x = bernoulli(p) #0 or 1
    if x == 0:
        eval = sin(theta)
    else:
        eval = cos(theta)
    return eval
```

Discrete Jumps



Smooth Expectation Value

```python
def program(theta):
    p = sigmoid(theta)
    x1 = bernoulli(p) #0/1
    x2 = bernoulli(p) #0/1
    eval = x1 + x2 # 0, 1 or 2
    return eval
```

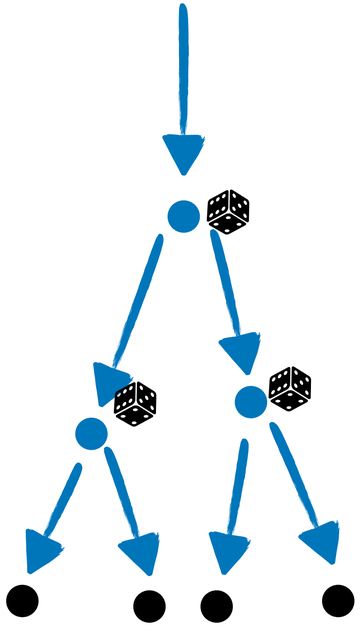


Differentiating Expectation Values

The best known algorithm for gradient estimation is used a lot in *Reinforcement Learning - score function estimation*

$$\nabla \bar{f}(\phi) = \mathbb{E}_{p_\phi}[f(x)\,\nabla_\phi \log p_\phi(x)]$$

requires tracking probabilities (and their gradients) while running code → probabilistic programming



HEP Simulator: discrete processe

Atari Games discrete actions

Differentiating Expectation Values

More recently, renewed interest in approaches based on “smoothed pertubaton analysis”

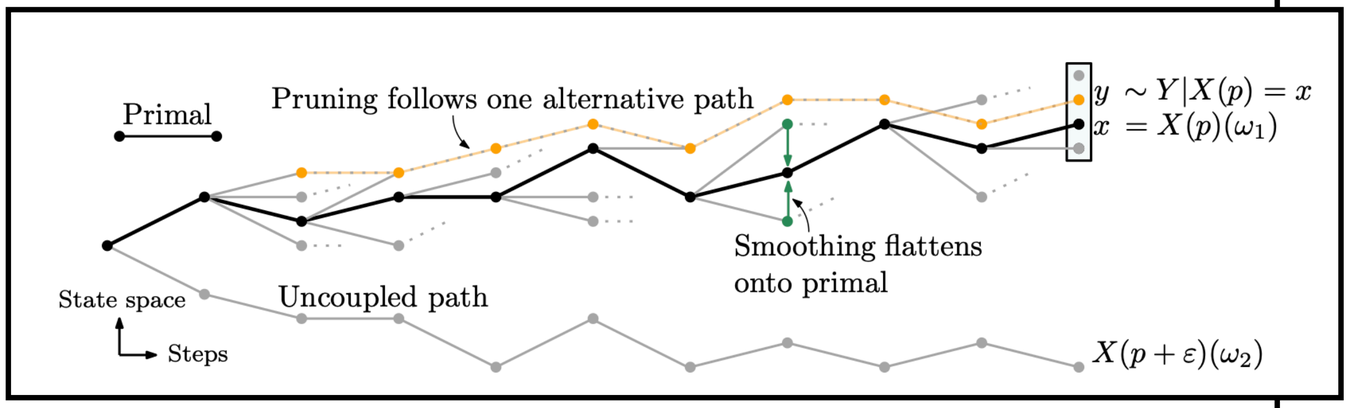
Extend fwd-mode AD to discrete-stochastic environments

$$(x, \delta) \rightarrow (x, \delta, w, y)$$

Dual Numbers in Standard AD

Quadruplets in Stochastic AD

$$\nabla_\phi \bar{f}(\phi) = \mathbb{E}_{p_\phi}[\delta + w(x - y)]$$



arxiv: 2210.08572

Automatic Differentiation of Programs with Discrete Randomness

Gaurav Arya
Massachusetts Institute of Technology, USA
aryag@mit.edu

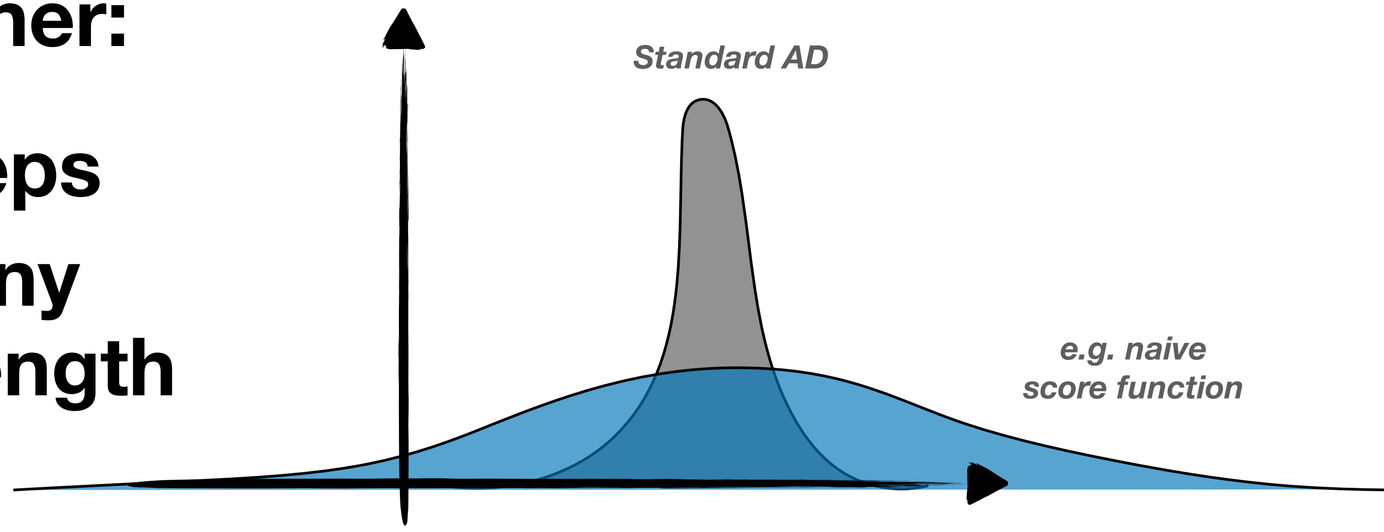
Moritz Schauer
Chalmers University of Technology, Sweden
University of Gothenburg, Sweden
smoritz@chalmers.se

Key Questions in HEP

Gradients Estimation in Stochastic Programs can suffer from very high variance → unstable optimization

HEP is in an interesting corner:

- **a lot of tiny stochastic steps**
- **branching processes, many independently, variable-length processes**



Q: Will these methods work in a HEP setting?

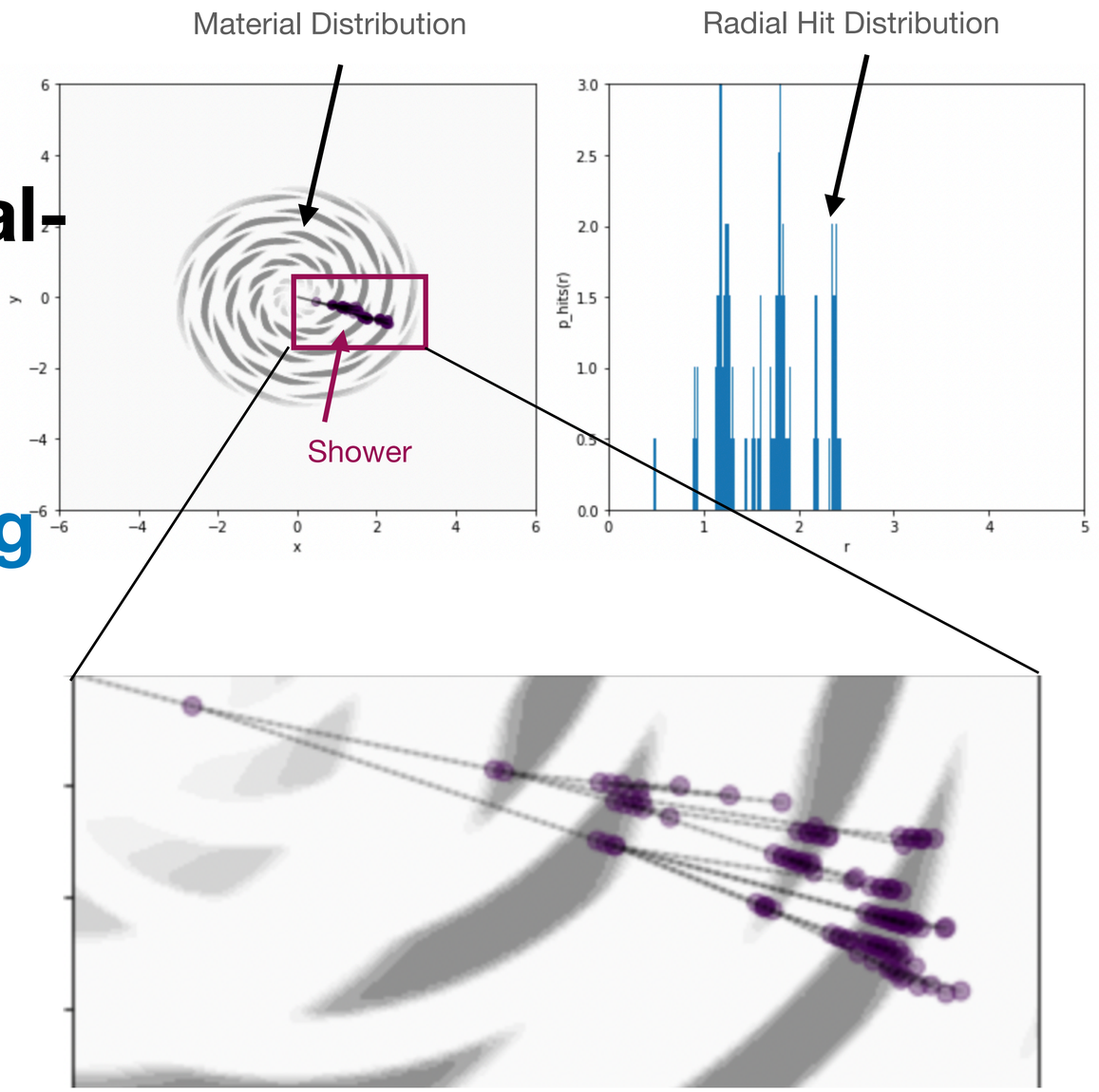
Example Application

We consider a simplified material-induced particle shower

High Density: E-loss and splitting
Low Density: linear propagation

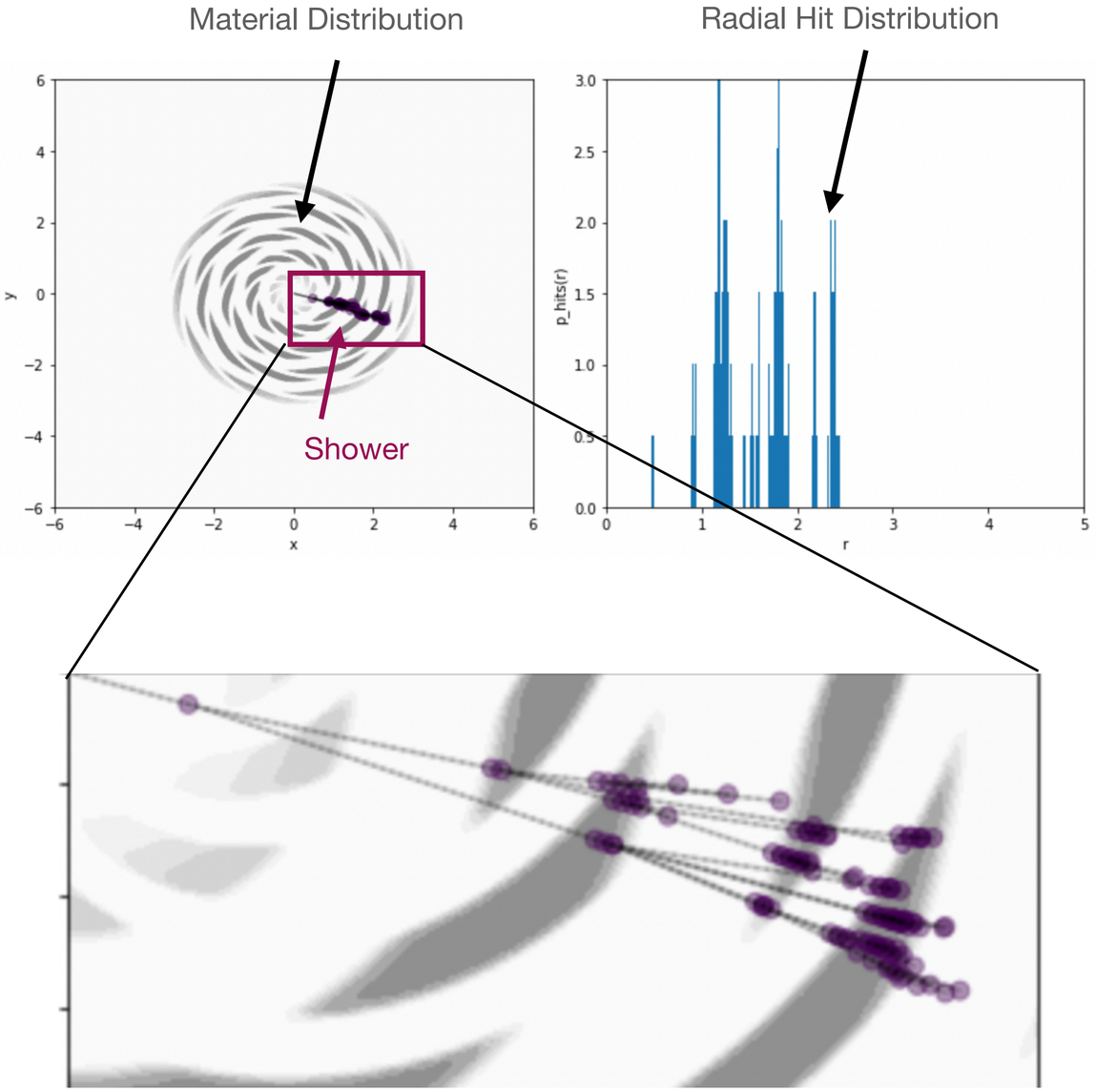
Design Parameter:
Radial Distance of Material

Design Goal: Shower Depth



Example Application

```
def simulate(t,ϕ){
    t′ = propagate(t)
    dointeract ~ p(interact|material(x,y,z| ϕ))
    if dointeract:
        hits.append([x,y,z])
        dosplit ~ p(split|t′)
        if dosplit:
            t1,t2 = split(t′)
            simulate(t1), simulate(t2)
        else:
            E = (1-α) E // energy loss
            simulate(t′)
}
```



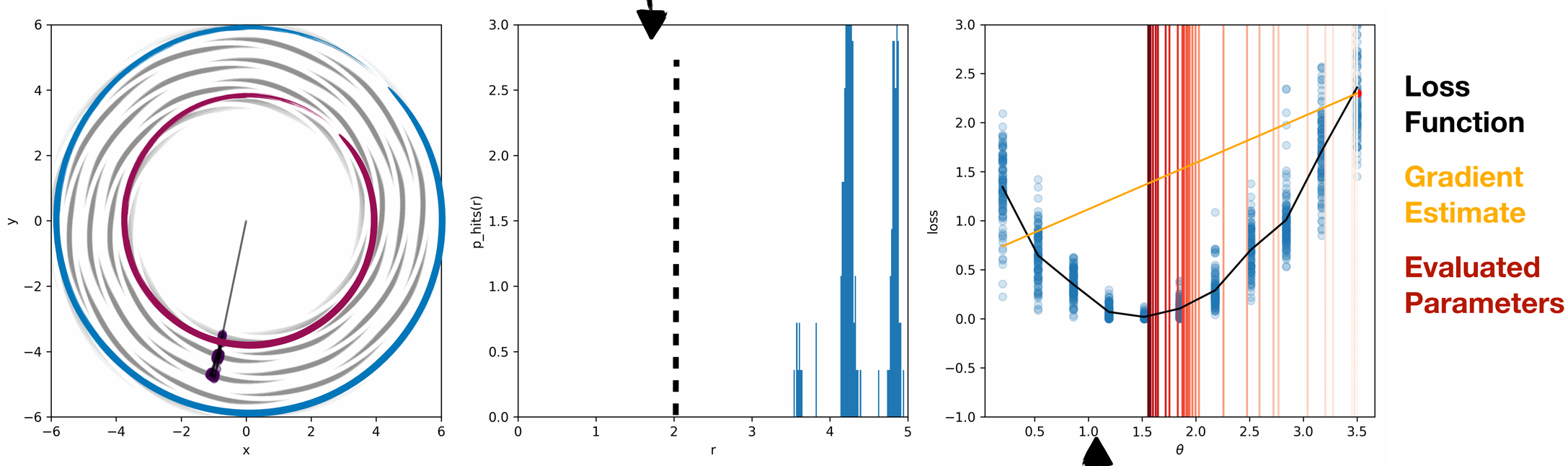
Optimizing Material Distribution

At least for this toy model, optimizing material for hits properties works

Objective: Mean Hit Radius → 2.0

initial

final



Loss Function

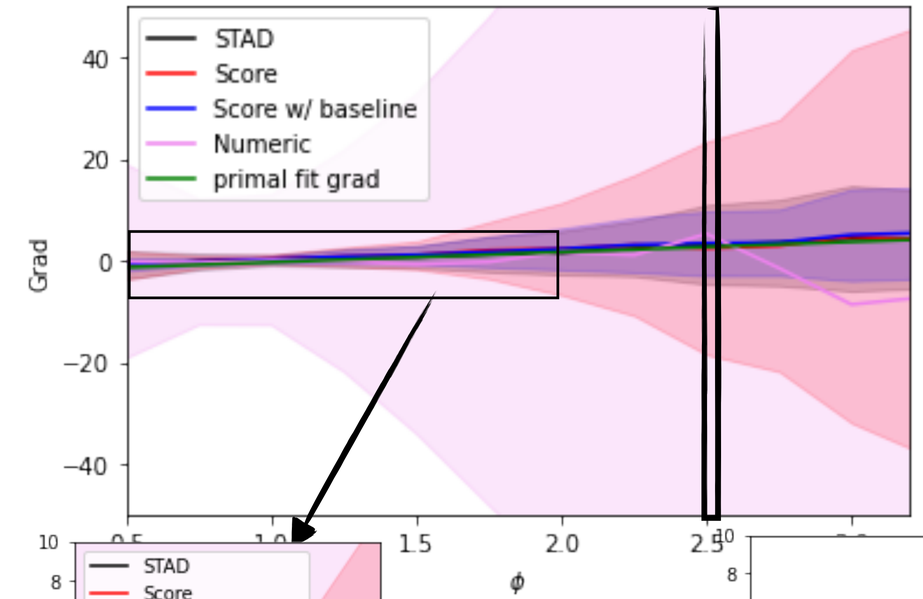
Gradient Estimate

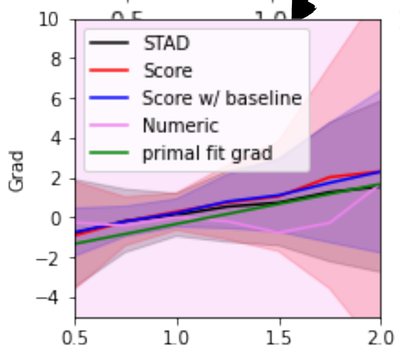
Evaluated Parameters

We do see noisy gradients, but:
as long as direction is OK (in expectation), **minimization works**

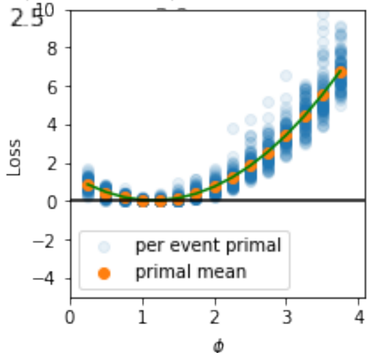
Comparison

Both score function and Stochastic AD gradient estimates massively reduce variance and optimization is possible

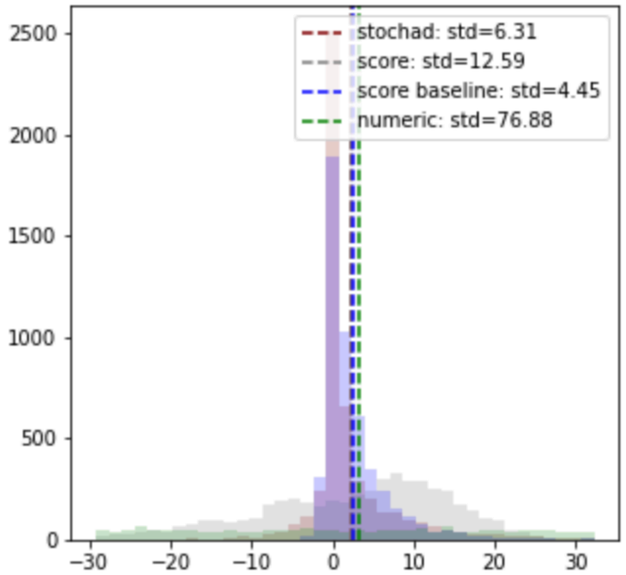




$= \nabla_\phi$



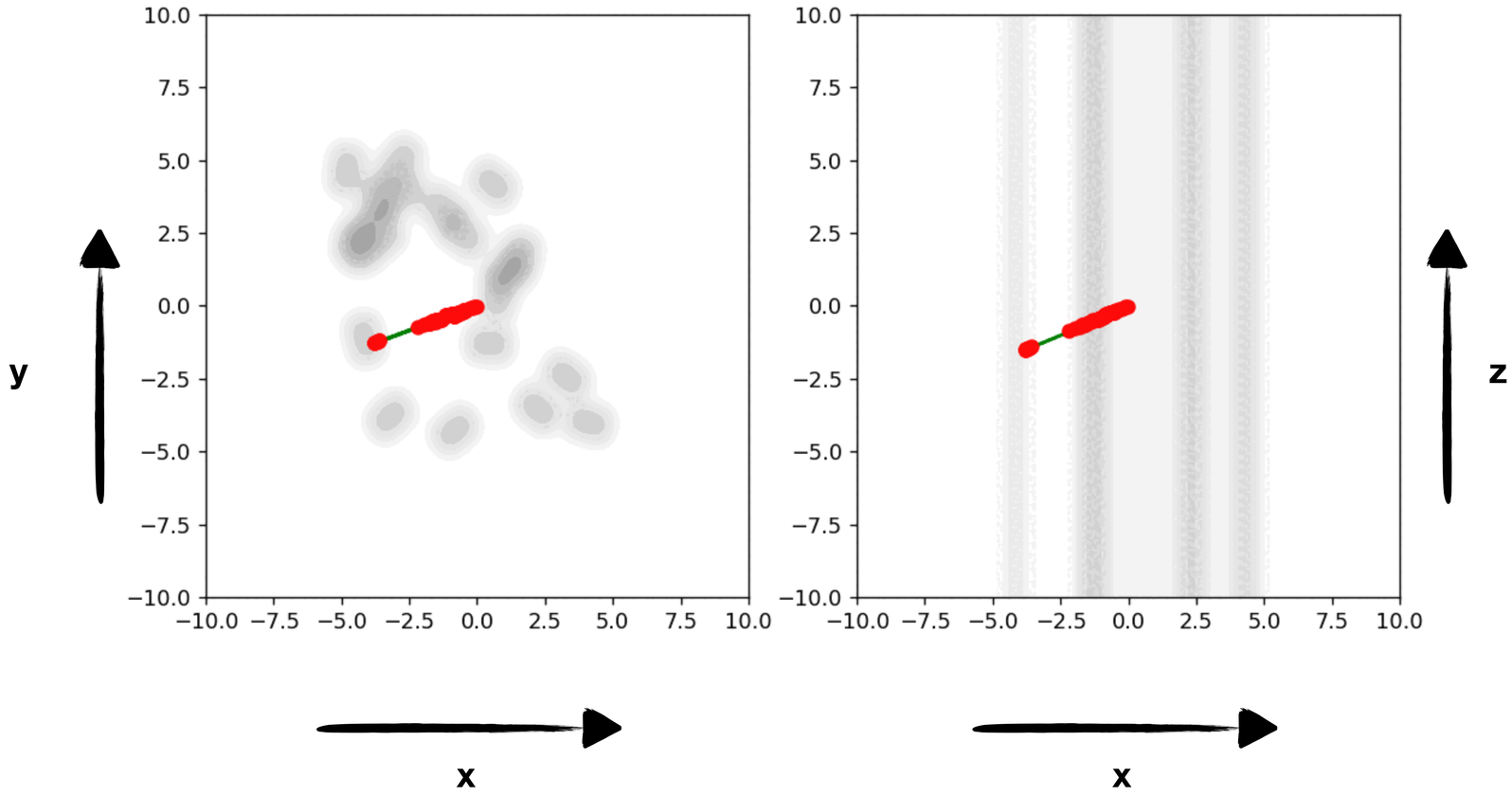
$\nabla_\phi L|_{\phi=2.5}$



A fun experiment:

Alternatively: 20 free-floating blocks of detectors.

Self-assembly into a circular shell to contain showers



Next Steps & Outlook

Takeaway message:

The presence of ***discrete stochastic randomness*** makes ***differentiable programming*** difficult …

But not impossible. With appropriate techniques, we **can** differentiate the seemingly undifferentiable

But, may require integration of more complex programming constructs than just AD.