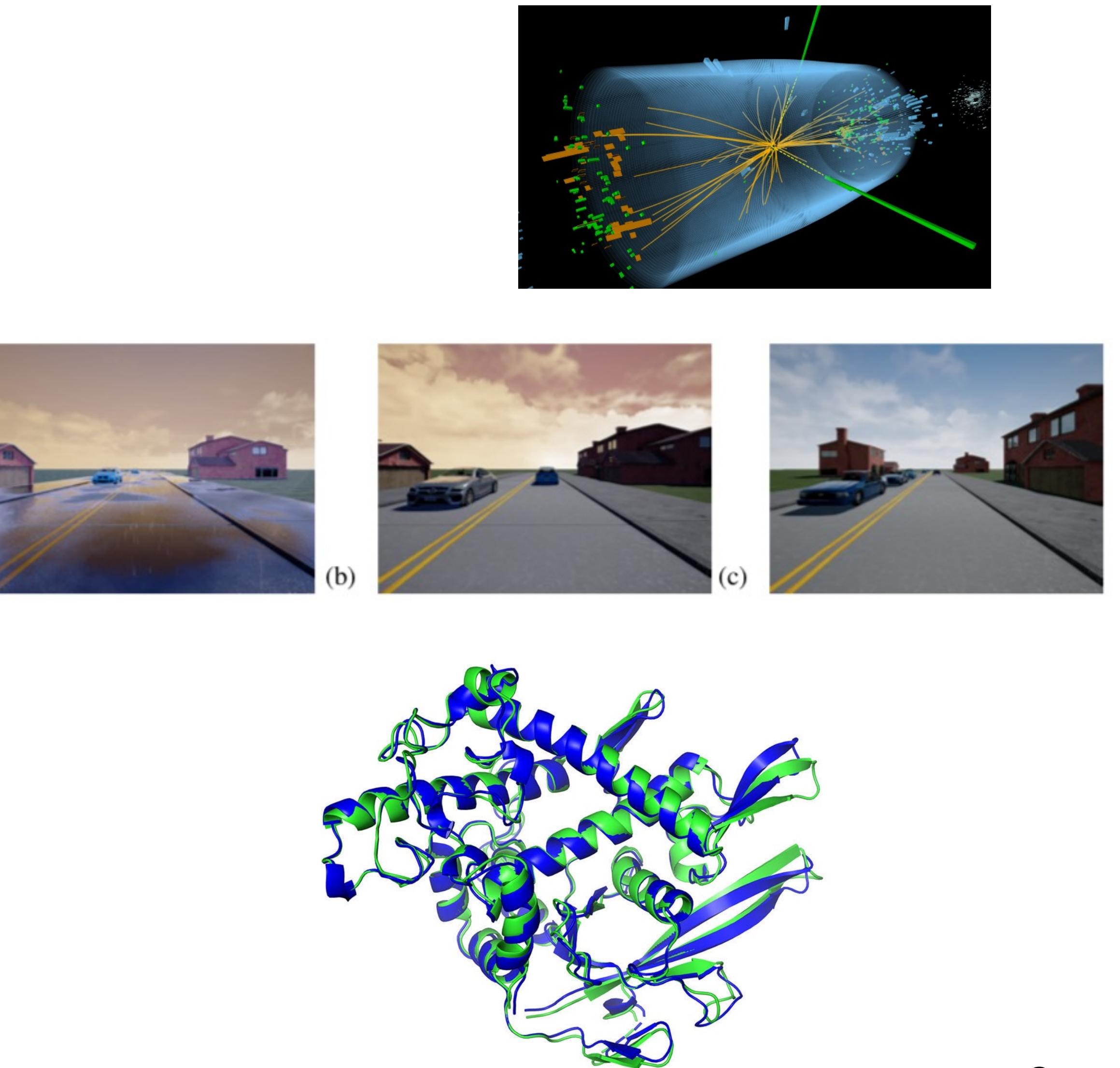


Black-box optimisation with Local Generative Surrogates

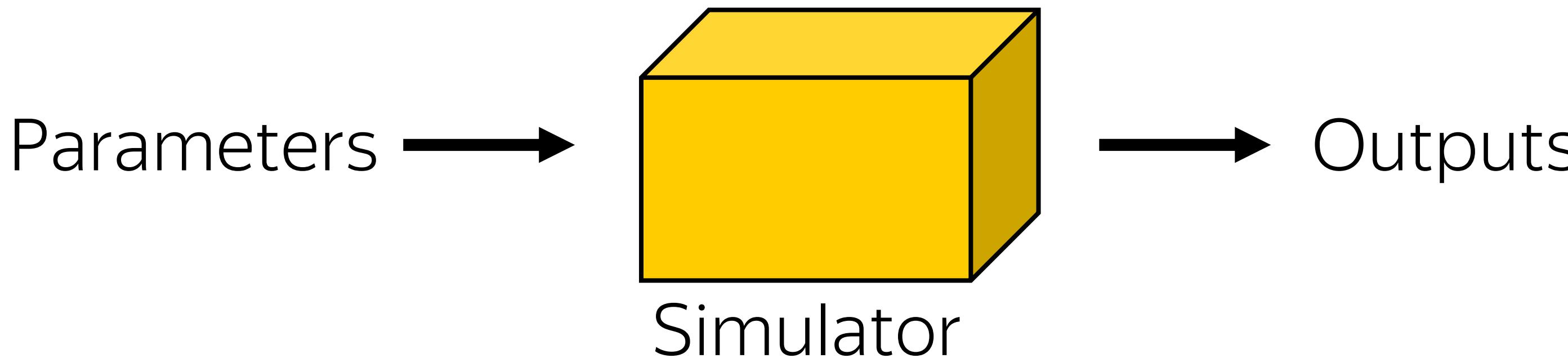
April 2023
Sergey Shirobokov
(ShareChat Research)

Simulators

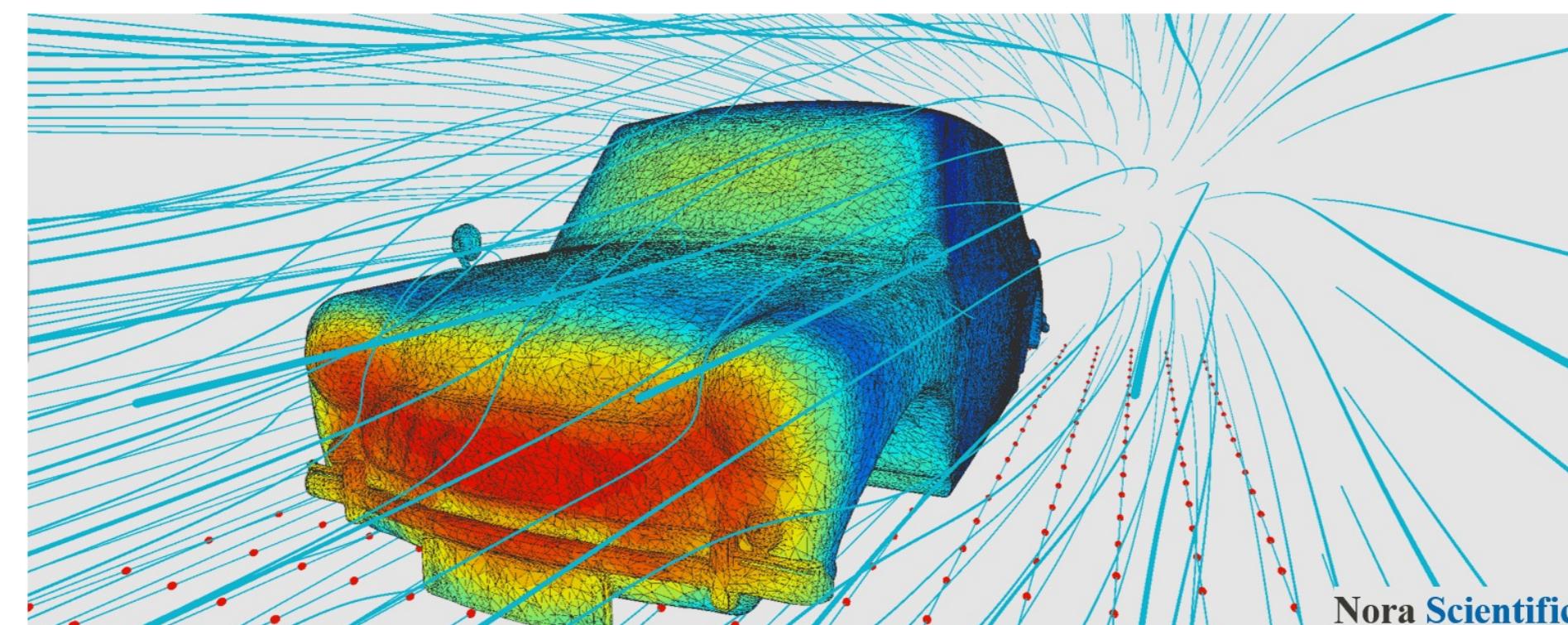
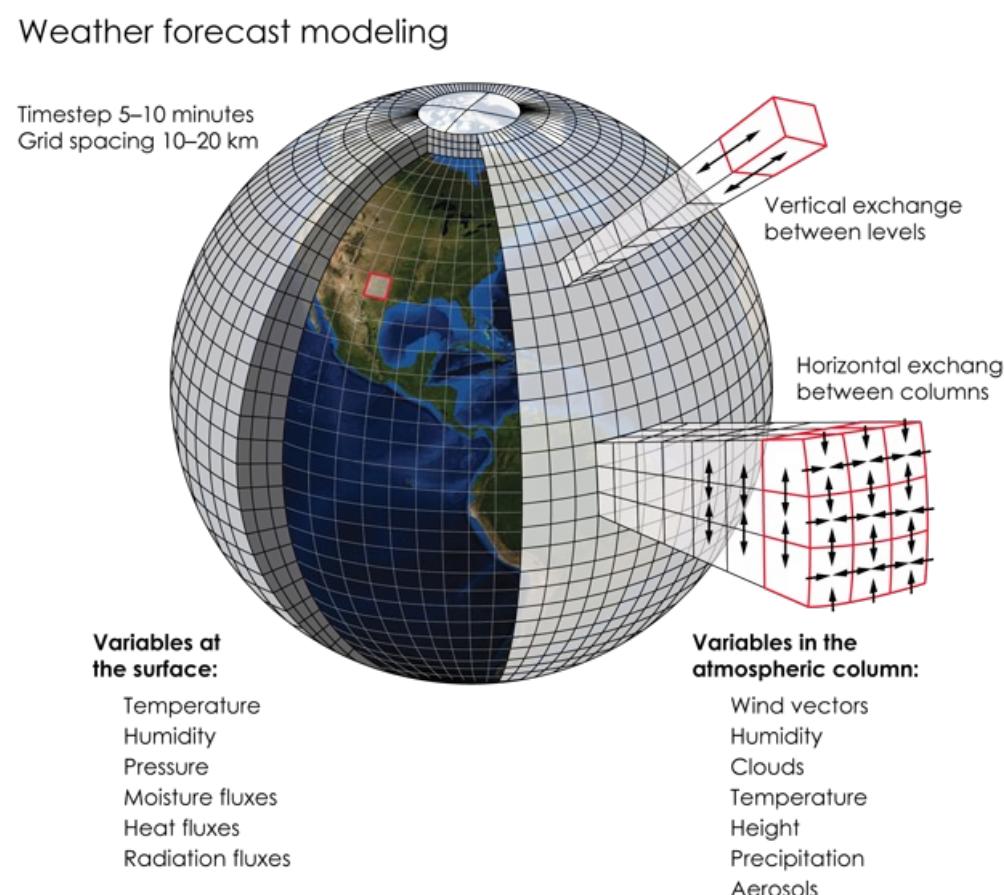
- High energy physics
- Drug discovery
- Weather predictions
- Vehicle control
- Hardware
- Material science



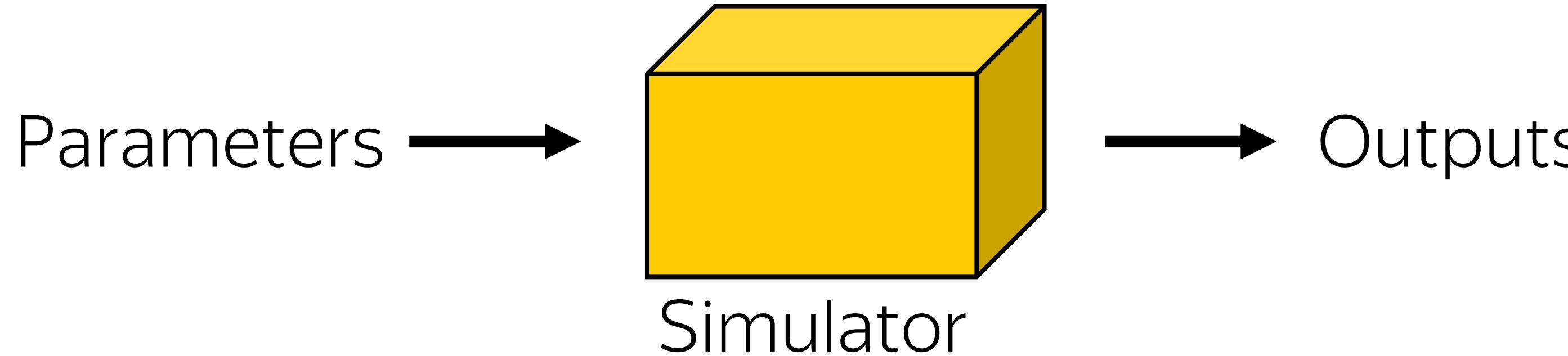
Simulators



- Prediction →
- Simulation of a (stochastic) system
 - Output samples from a distribution

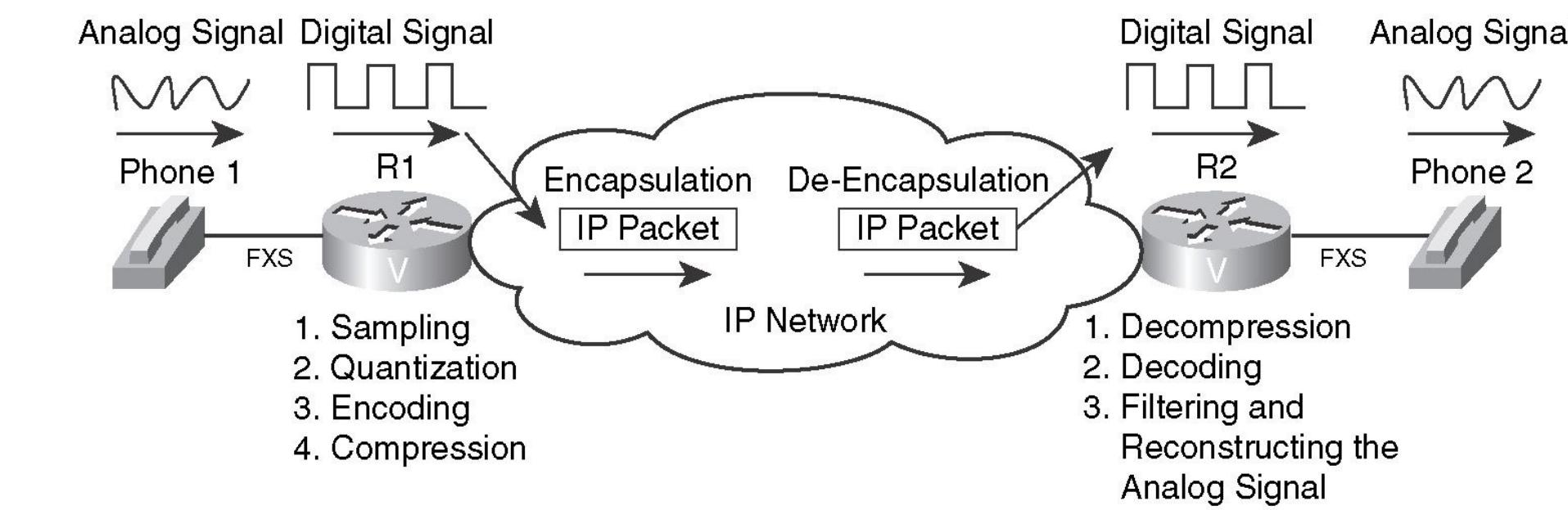
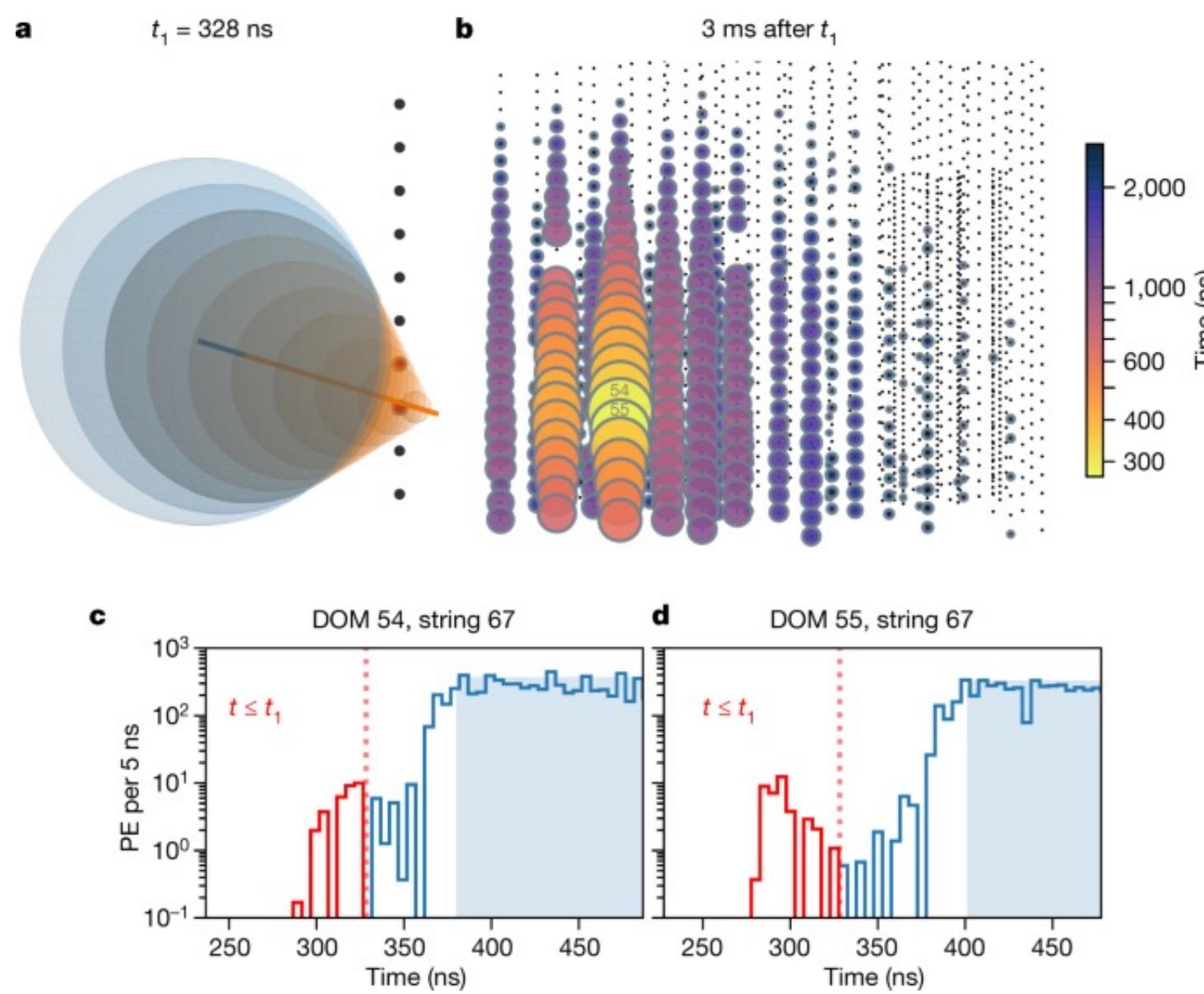
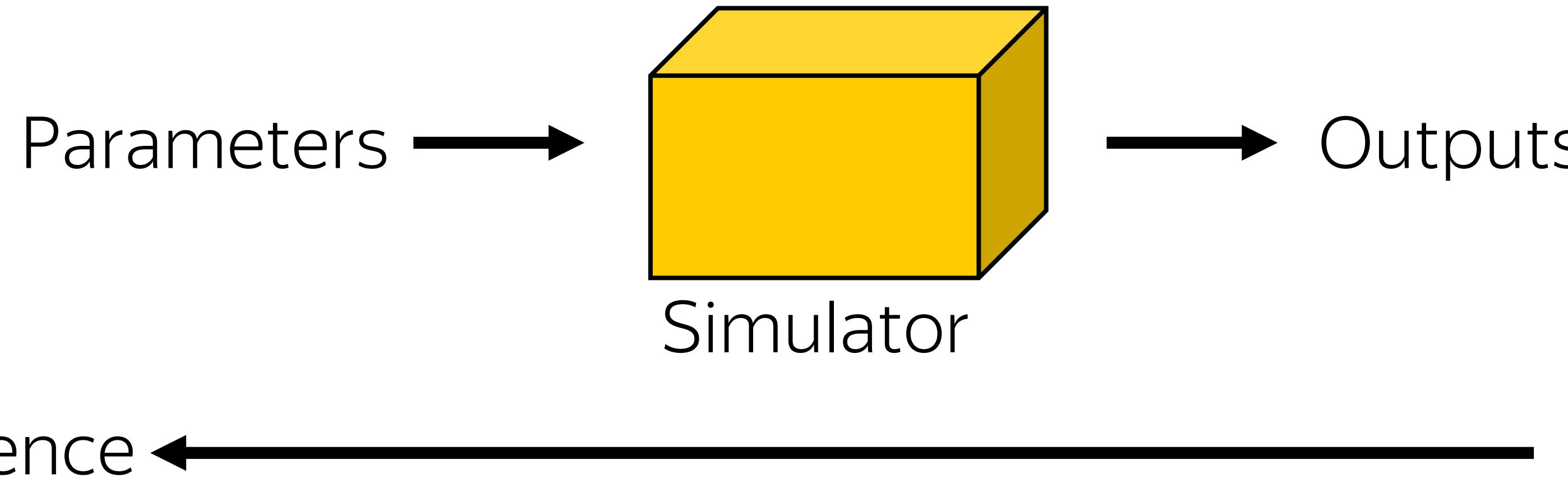


Simulators

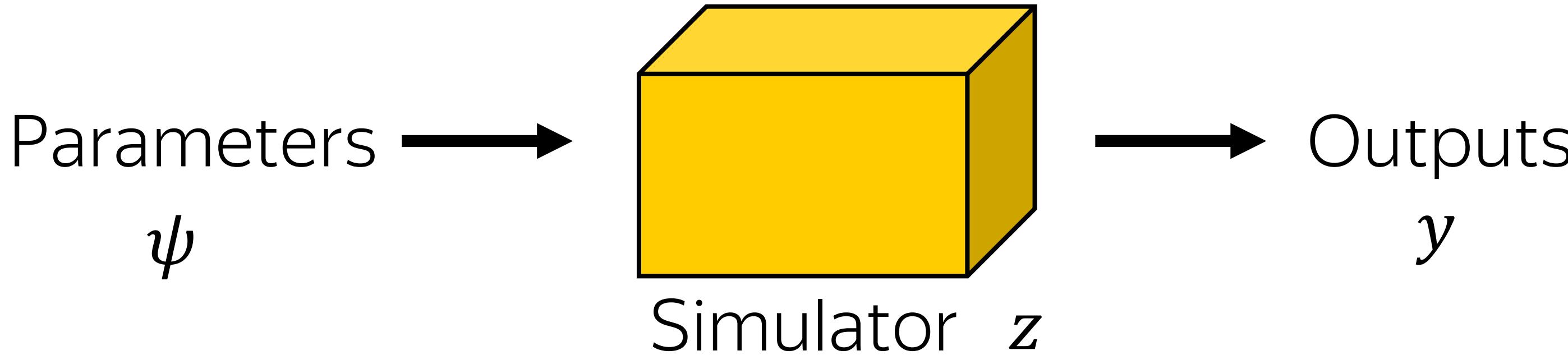


- Prediction →
- Simulation of a (stochastic) system
 - Output samples from a distribution
- Inference ←
- Find parameters that explain observed data
 - Perform optimisation w.r.t to the parameters
 - Solve inverse problem

Simulators



Simulator-based optimisation



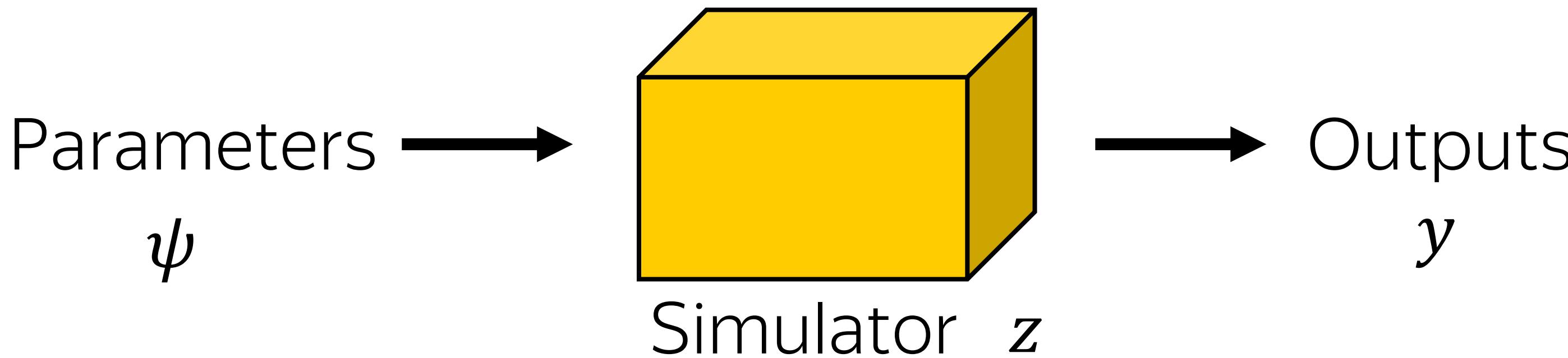
Prediction $\xrightarrow{y, z, \psi \sim p(y, z, \psi)}$

Observe: $y \sim p(y|\psi) = \int p(y, z|\psi) dz$

Inference $\xleftarrow{\text{Likelihood: } p(y|\psi) = \int p(y, z|\psi) dz}$

Posterior: $p(\psi|y) = \frac{p(y|\psi) p(\psi)}{p(y)}$

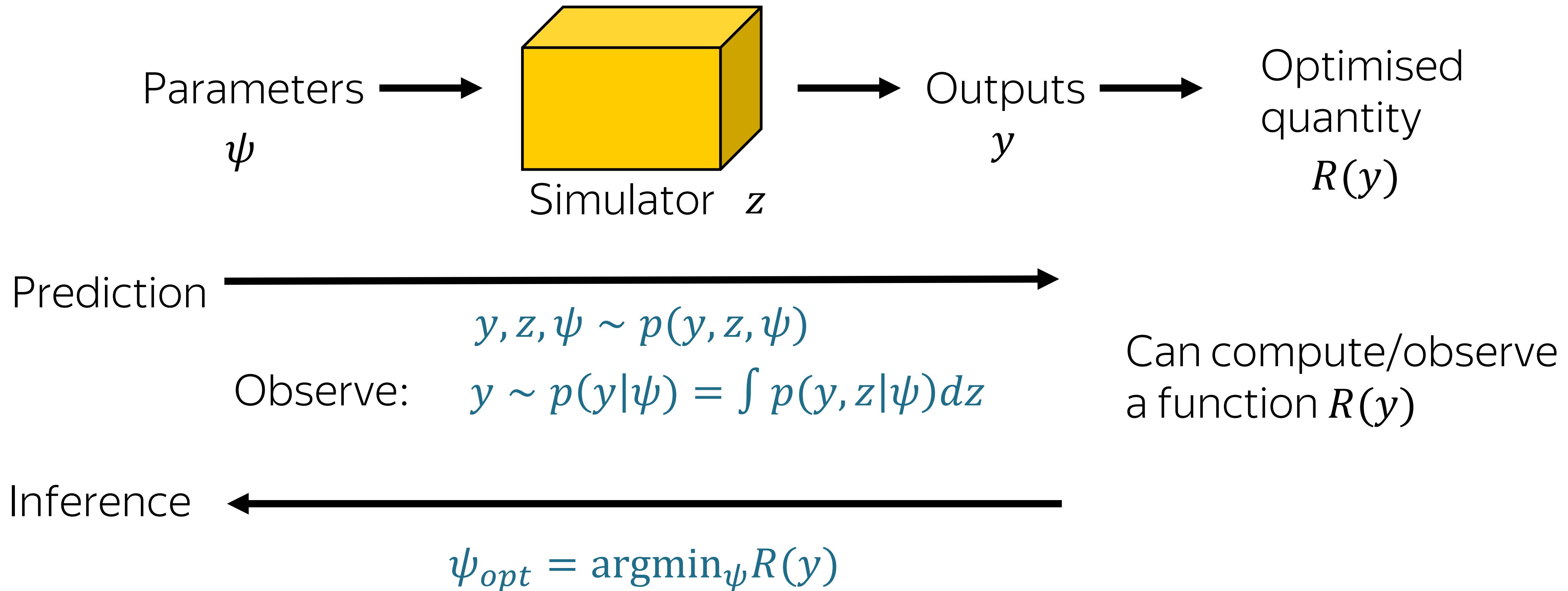
Simulation-based Inference problem



- Observe: $y_{det} \sim p(y_{det}|\psi_{true})$
- Can generate: $y_{sim} \sim p(y_{sim}|\psi)$
- Find:
 - MLE estimate ψ_{MLE} of ψ_{true} → requires $p(y|\psi)$
 - Posterior: $p(\psi|y_{det})$ → requires $p(y|\psi)$
- A variety of methods exists to solve this problem
ABC, Neural networks-based solutions

$$p(\psi|y) = \frac{p(y|\psi) p(\psi)}{p(y)}$$

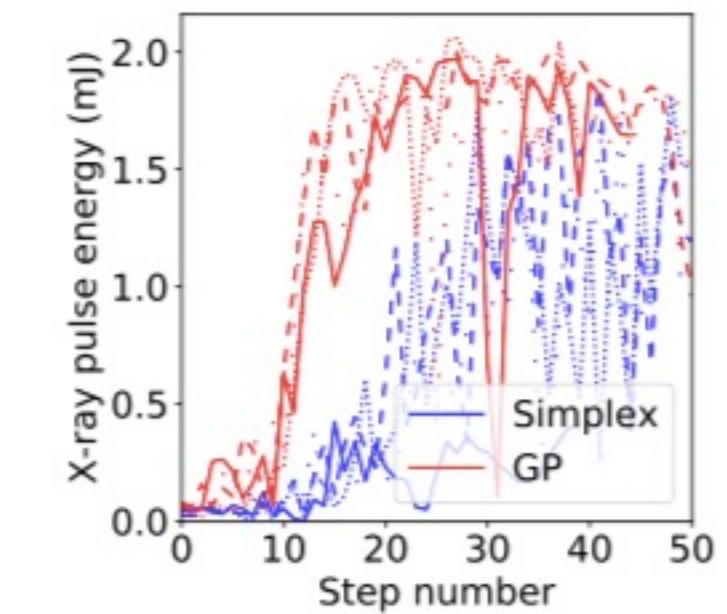
Simulator optimisation problem



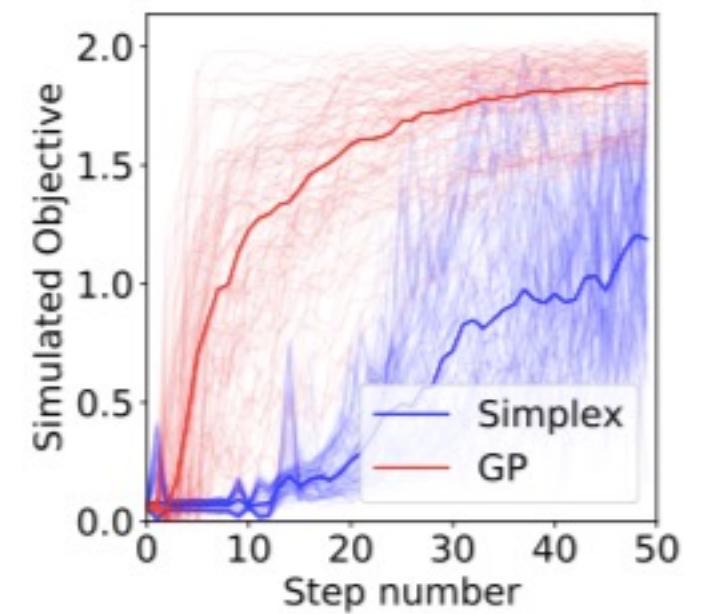
- Not interested in MLE or a posterior

Simulators optimisation

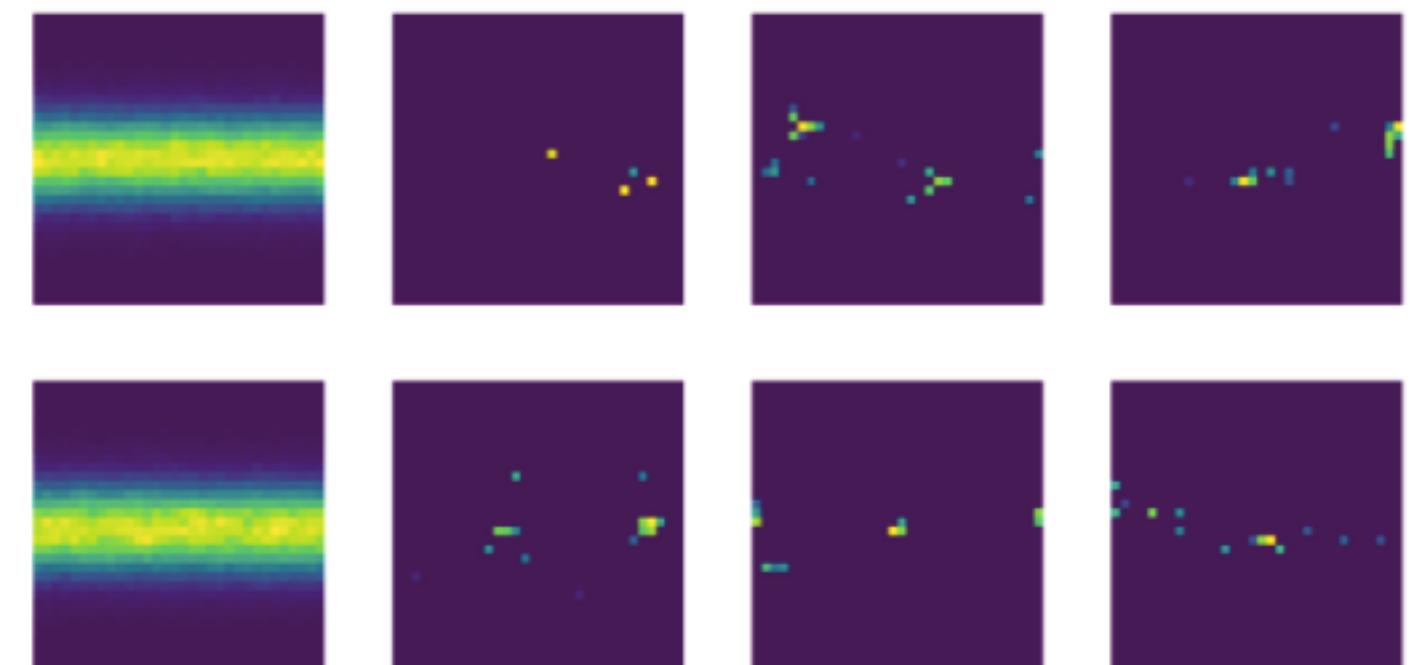
- Traffic scenes generation
- Control of the accelerator
- High Energy Physics
- Hardware



(c) Live 12 quadrupole optimization



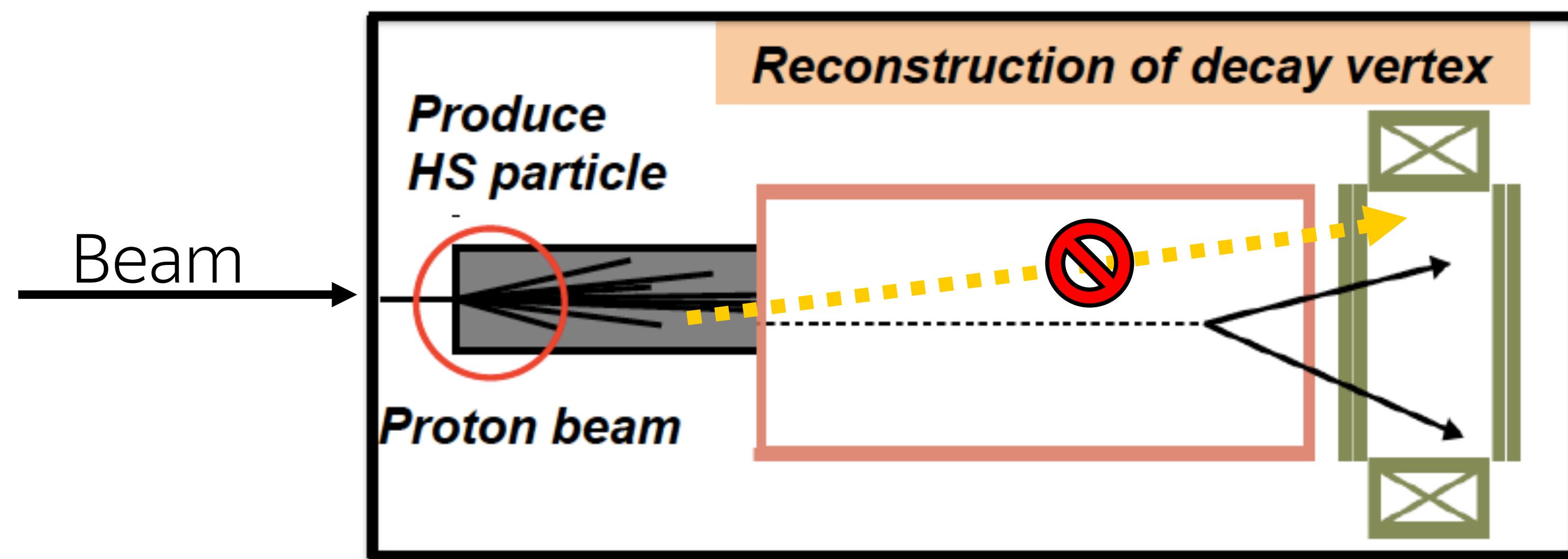
(d) Simulated 12 quadrupole optimization



Examples: [1810.02513](#), [1610.06151](#), [1909.05963](#), [1707.07113](#)

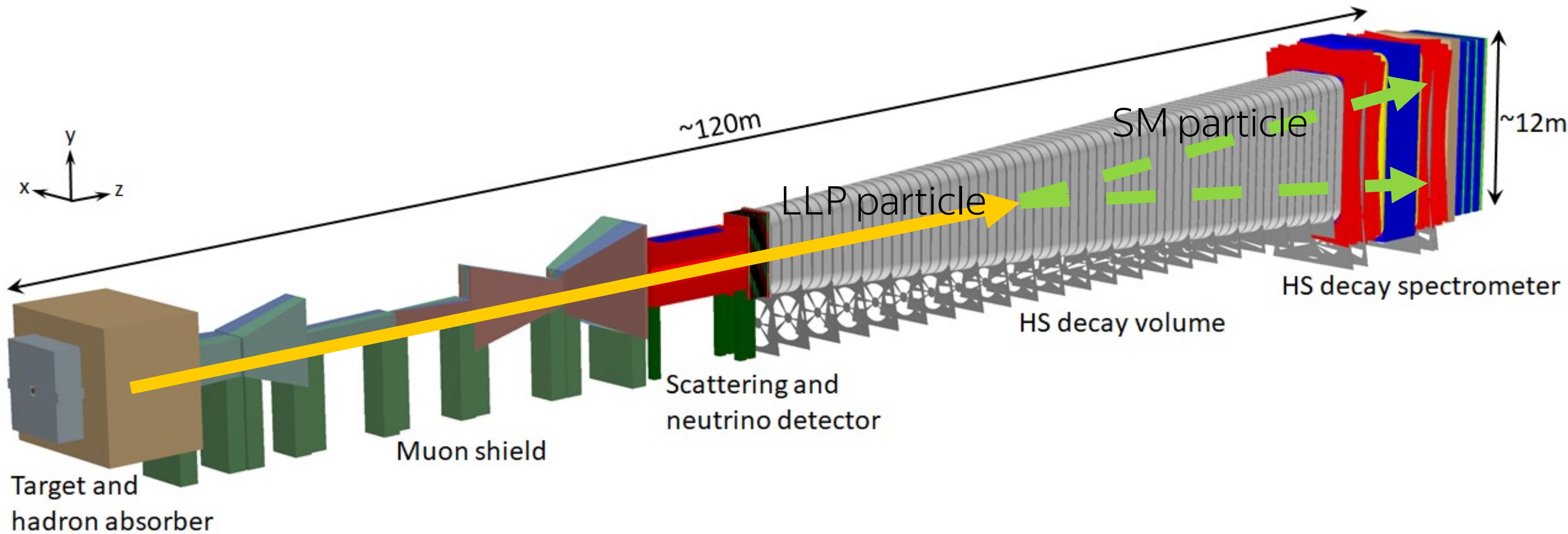
Motivation. HEP experiment

- › Particle experiment at CERN
- › 400 GeV/c proton beam
- › Beam dump
- › Hidden or LLP particles(signal), i.e. Dark Matter
- › Background(muons)
- › Goal: Detect signal, remove background



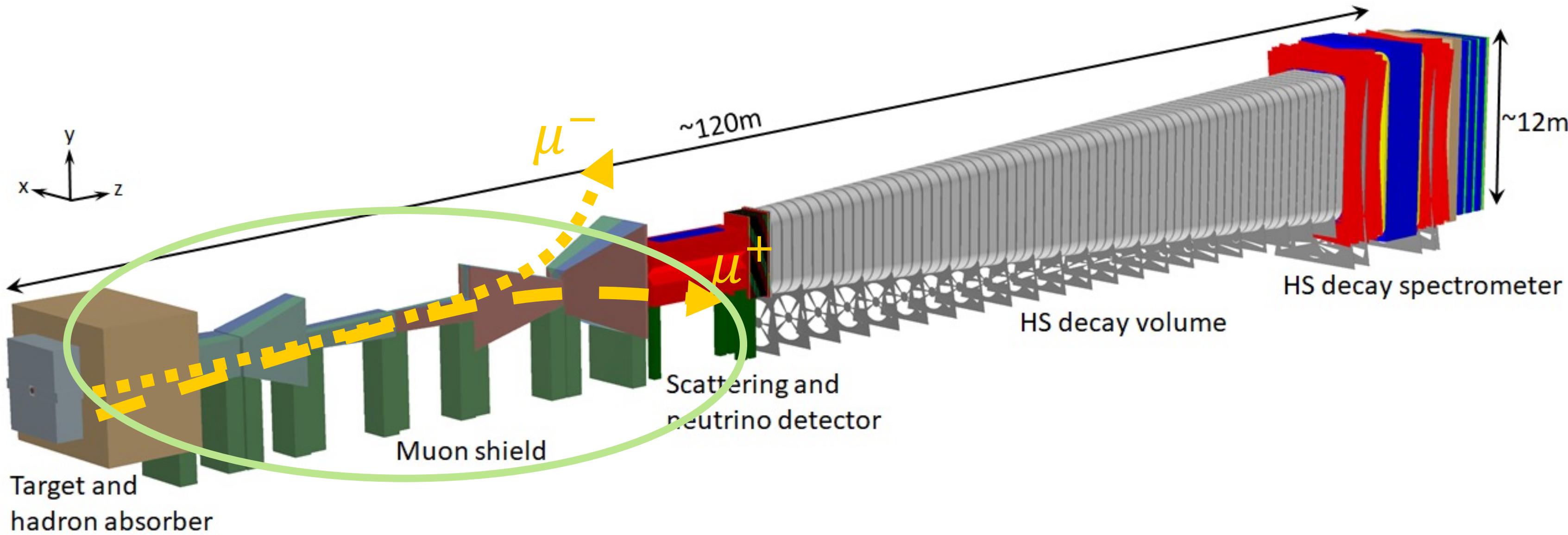
SHiP experiment

SHiP Technical proposal
1504.04956



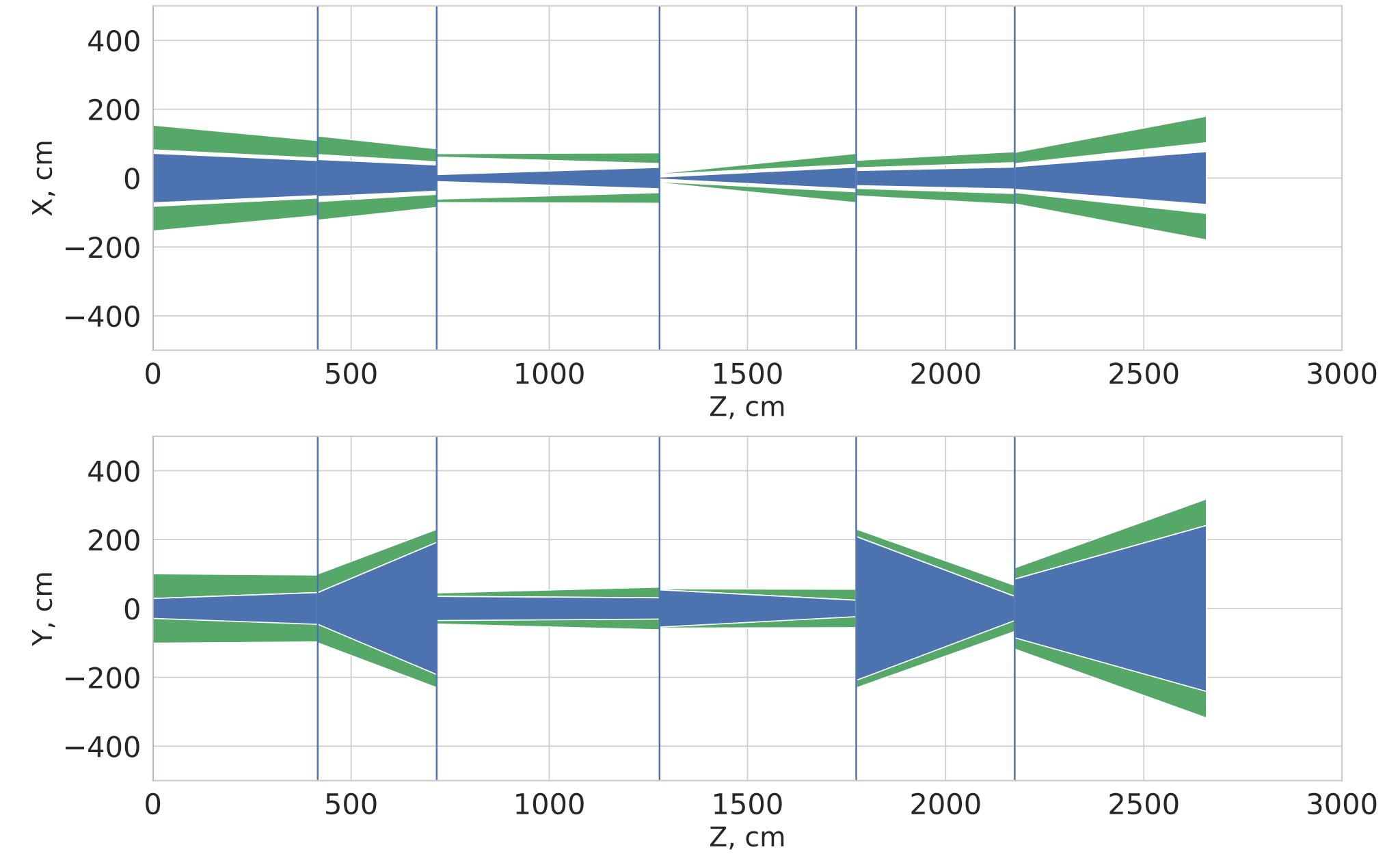
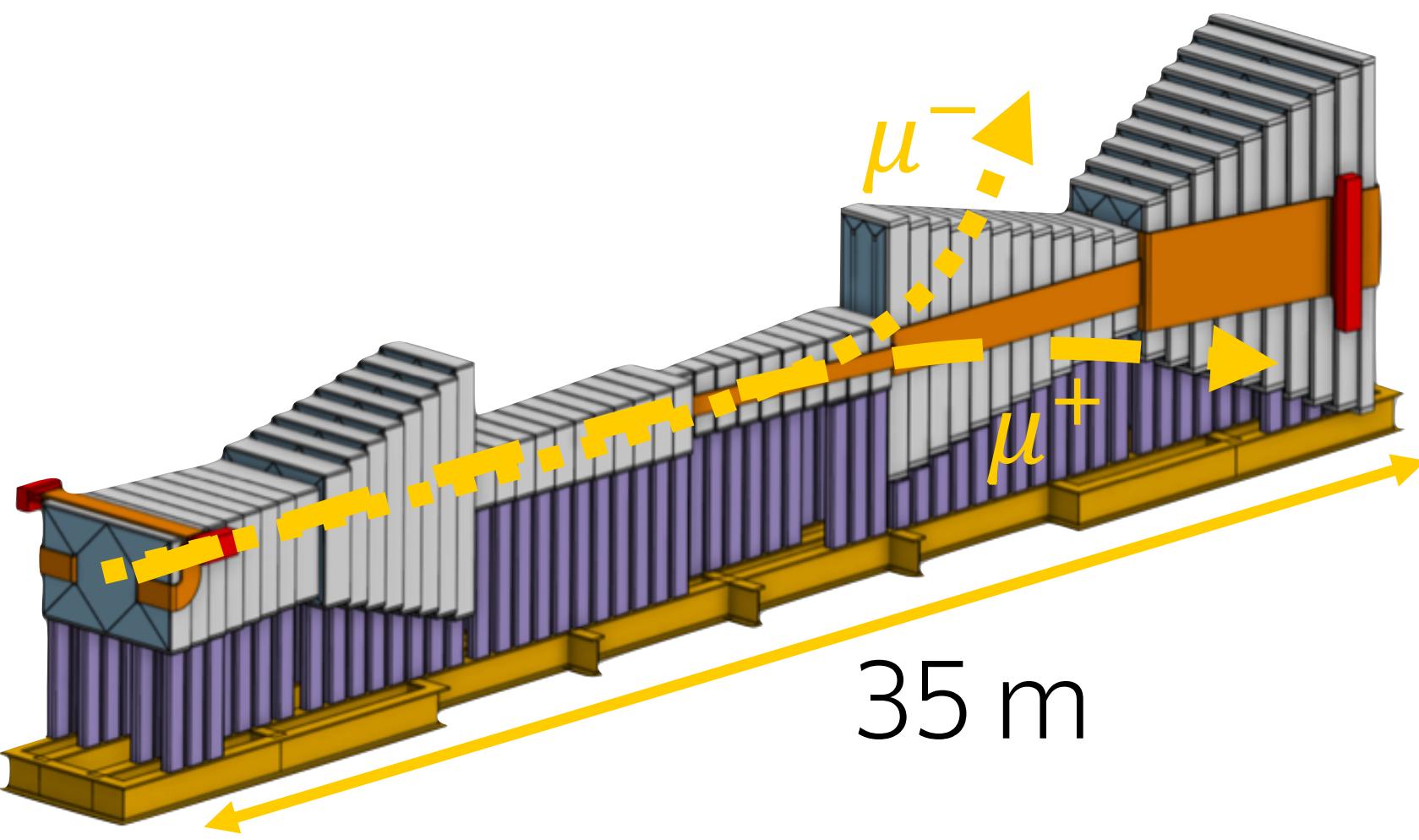
- Can detect very weakly coupled long-lived particles via decay or DM via scattering
- Planned zero background experiment

SHiP background



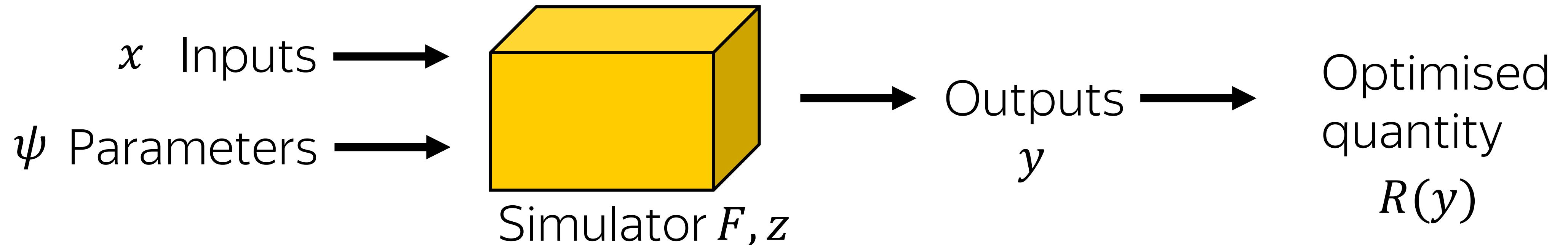
- 10^{11} muons/spill is produced inside the target
- Reduce background rate by six orders of magnitude
- Optimise the shield for the best physics performance and cost

SHiP muon shield



- Optimise parameters to reduce number of muons in the detector
- GEANT4 simulates muons propagation through the shield
- Shield is characterised by 42 parameters
- Very slow to simulate data points → hard to generate large samples*

Simulator optimisation problem



- x – muon kinematics
- ψ – geometry of the shield
- F – is the GEANT4 simulator
- y – observations: output of the simulator
- $R(y)$ - objective function

Want to find:

$$\psi_{opt} = \operatorname{argmin}_{\psi} E_y [R(y)]$$

Observe samples:

$$y = F(x, \psi, z)$$

Stochastic black-box simulator

- F – random variable:

$$y = F(x, \psi, z) \leftrightarrow y \sim p(y|x; \psi)$$

z – latent variable

- F – black-box

$p(y|x; \psi)$ is not known

Can only sample from p

- How to optimise?

$$\psi_{opt} = \operatorname{argmin}_{\psi} E_y [R(y)]$$

Stochastic black-box optimisation

$$\operatorname{argmin}_{\psi} E_y[R(y)] = \operatorname{argmin}_{\psi} \int R(y) p(y|x; \psi) q(x) dx dy$$



Intractable

How can the optimisation be performed in such case?

- Bypass the estimation of $p(y|x; \psi)$
- Compute $\nabla_{\psi} E_y[R(y)] \rightarrow \nabla_{\psi} p(y|x; \psi)$

Existing black-box optimisation methods

Gradient based:

- Numerical differentiation
- Score function estimation (REINFORCE)

Alternatives:

- Bayesian optimisation
- Evolutionary strategies

Proposed by us:

- Local Generative Surrogate optimisation

Gradient descent

- Goal: $\text{Argmin}_{\psi} f(\psi)$.
- Solution: $\nabla_{\psi} f(\psi) = 0$,
- Method: gradient descent

$$\psi_{t+1} = \psi_t - \text{const} * \nabla_{\psi} f(\psi)$$

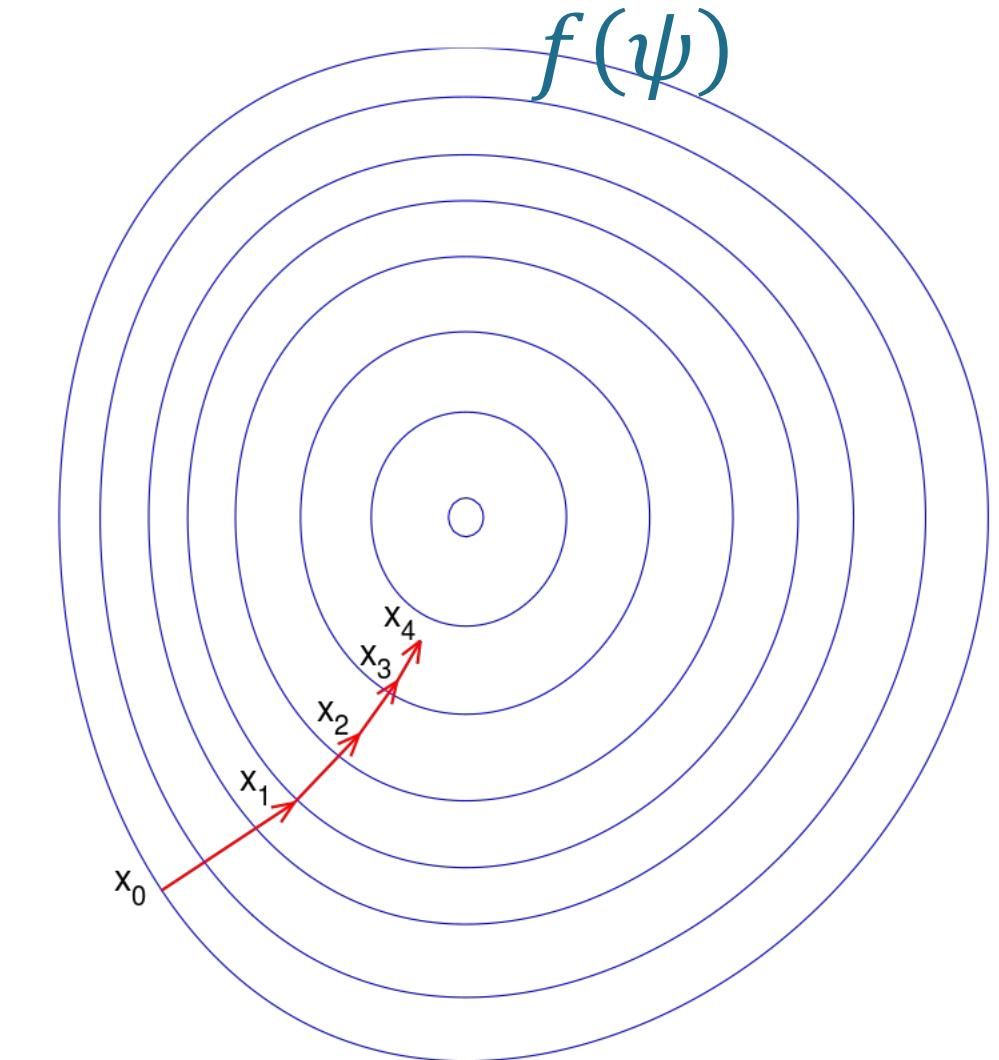
- Problem: $\nabla_{\psi} f(\psi)$ can NOT be computed

Algorithm 1 Gradient descent

Require: Initial point ψ_0 , learning rate α

- 1: **while** ψ has not converged **do**
 - 2: $\psi_{i+1} = \psi_i - \alpha \nabla_{\psi} f(\psi)$
 - 3: **end while**
-

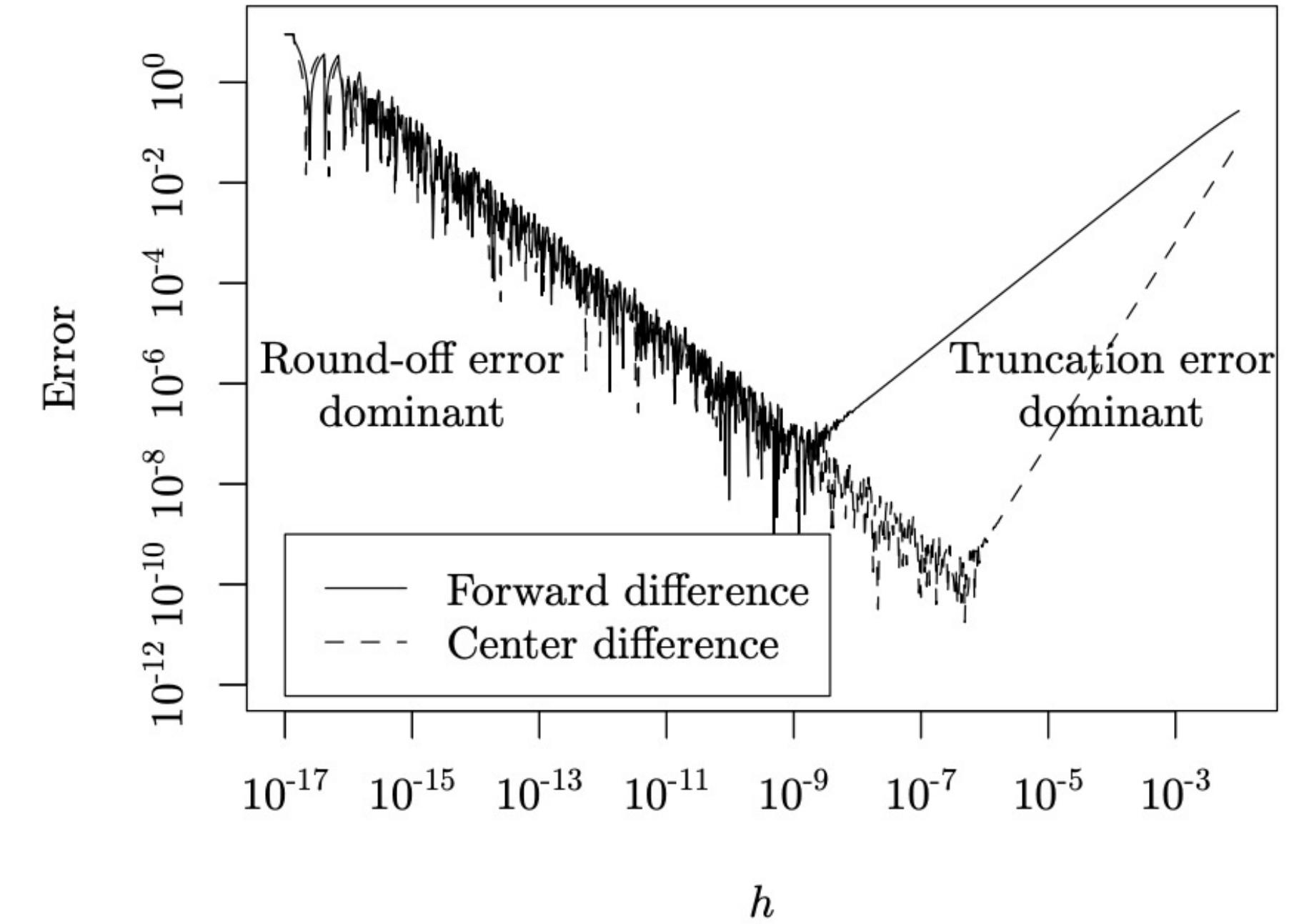
How to estimate $\nabla_{\psi} f(\psi)$?



Numerical differentiation

$$\nabla_{\psi} f(\psi) \approx \frac{f(\psi+h) - f(\psi)}{h}, h - \text{step size}$$

- May have numerical instabilities
- Require $O(d)$ evaluation of f . $\psi \in \mathbb{R}^d$
- Can be challenging to apply with stochastic functions
- Perform linear interpolation



Score function estimator

- Often called as REINFORCE gradient estimation
- Remember our objective function:

$$f(\psi) = E_y[R(y_\psi)] = \int R(y) p(y|x; \psi) q(x) dx dy,$$

- Introduce distribution over ψ : $\psi \sim p(\psi|\mu)$
- $f(\psi)$ is now stochastic. We want to optimise $E_{\psi \sim p(\psi|\mu)} [f(\psi)]$

$$\nabla_\psi f(\psi) \approx \nabla_\mu E_{\psi \sim p(\psi|\mu)} [f(\psi)]$$

- How can we compute $\nabla_\mu E_{\psi \sim p(\psi|\mu)} [f(\psi)]$?

Score function estimator

- Remember that: $\nabla_x \log(f(x)) = \frac{\nabla_x f(x)}{f(x)}$ \rightarrow $\nabla_x f(x) = f(x) \nabla_x \log(f(x))$

$$\begin{aligned}\nabla_\mu E_{\psi \sim p(\psi|\mu)}[f(\psi)] &= \nabla_\mu \int f(\psi) \underline{p(\psi|\mu)} d\psi = \int f(\psi) \underline{\nabla_\mu p(\psi|\mu)} d\psi \\ &= \int f(\psi) \underline{\underline{p(\psi|\mu)}} \nabla_\mu \log(p(\psi|\mu)) d\psi \\ &= E_{\psi \sim p(\psi|\mu)}[f(\psi) \nabla_\mu \log(p(\psi|\mu))]\end{aligned}$$

- We have an estimate of the gradient!

$$\nabla_\psi f(\psi) \approx E_{\psi \sim p(\psi|\mu)}[f(\psi) \nabla_\mu \log(p(\psi|\mu))]$$

Score function estimator

- Trick:

$$\nabla_{\psi} f(\psi) \approx E_{\psi \sim p(\psi|\mu)} [f(\psi) \nabla_{\mu} \log(p(\psi|\mu))]$$

Can evaluate

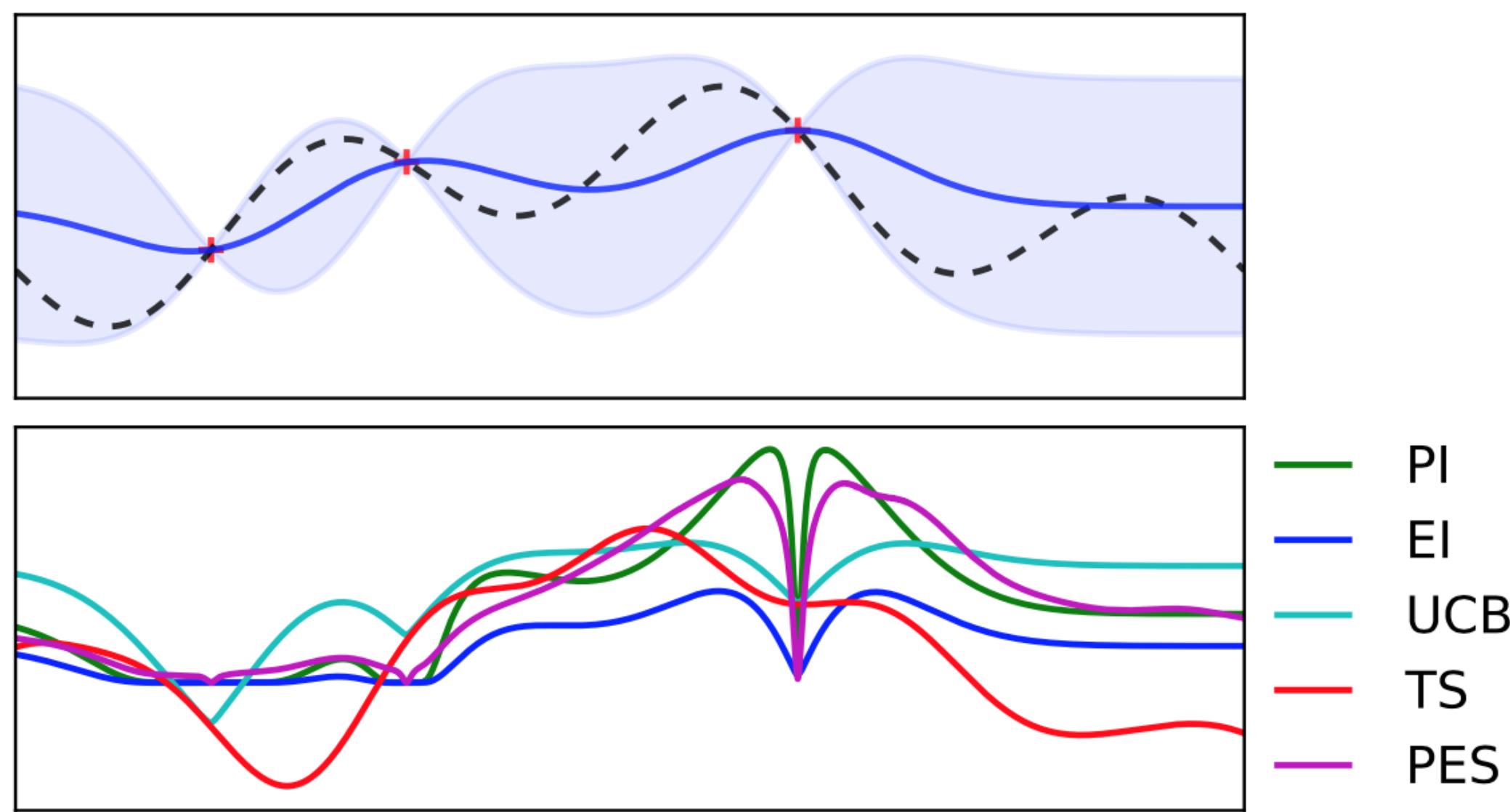
Can compute

- Have high variance[1]
- Require prior distribution over ψ
- Techniques developed to reduce variance[2,3]

But: Fast to compute

[1]<https://doi.org/10.1007/BF00992696>. [2]1711.00123, [3]1810.02513

Bayesian optimisation with Gaussian Processes



- Goal: $\operatorname{argmin}_{\psi} f(\psi)$
- Approximate $f(\psi)$ with surrogate model (GP) $\rightarrow \mu(\psi)$ and $\sigma(\psi)$
- Choose acquisition function $\beta(\psi)$: Set exploration/exploitation of the space
- Evaluates $\beta(\psi) \rightarrow$ new point ψ' to probe
- Benefits:
 - Can potentially find global minima
 - Work with non-differentiable functions
- Drawbacks:
 - Scales as $O(n^3 + n^2d)$, n – size of the training set
 - Suffer from curse of dimensionality

Evolutionary strategies

Algorithm 3 General purpose evolutionary algorithm

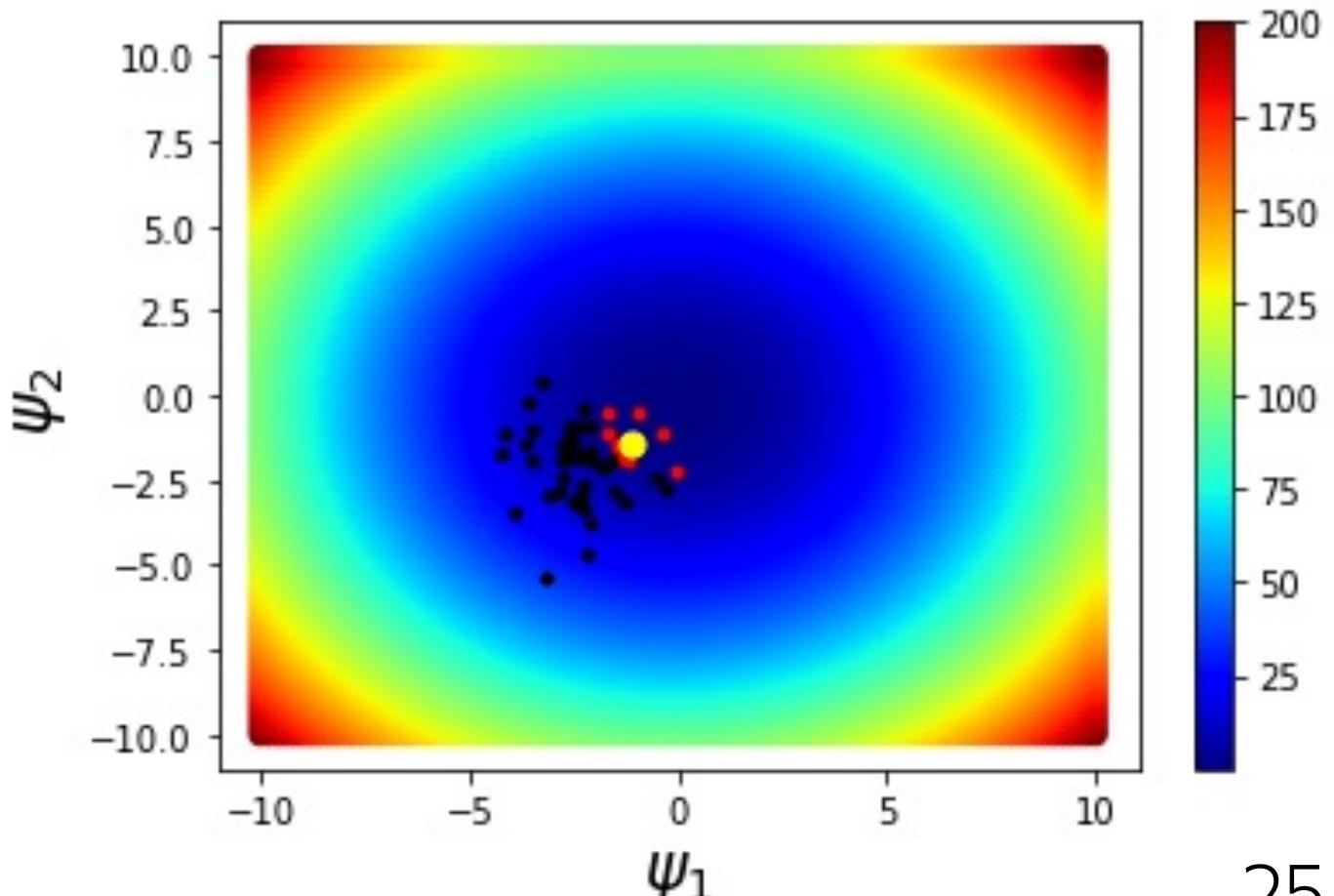
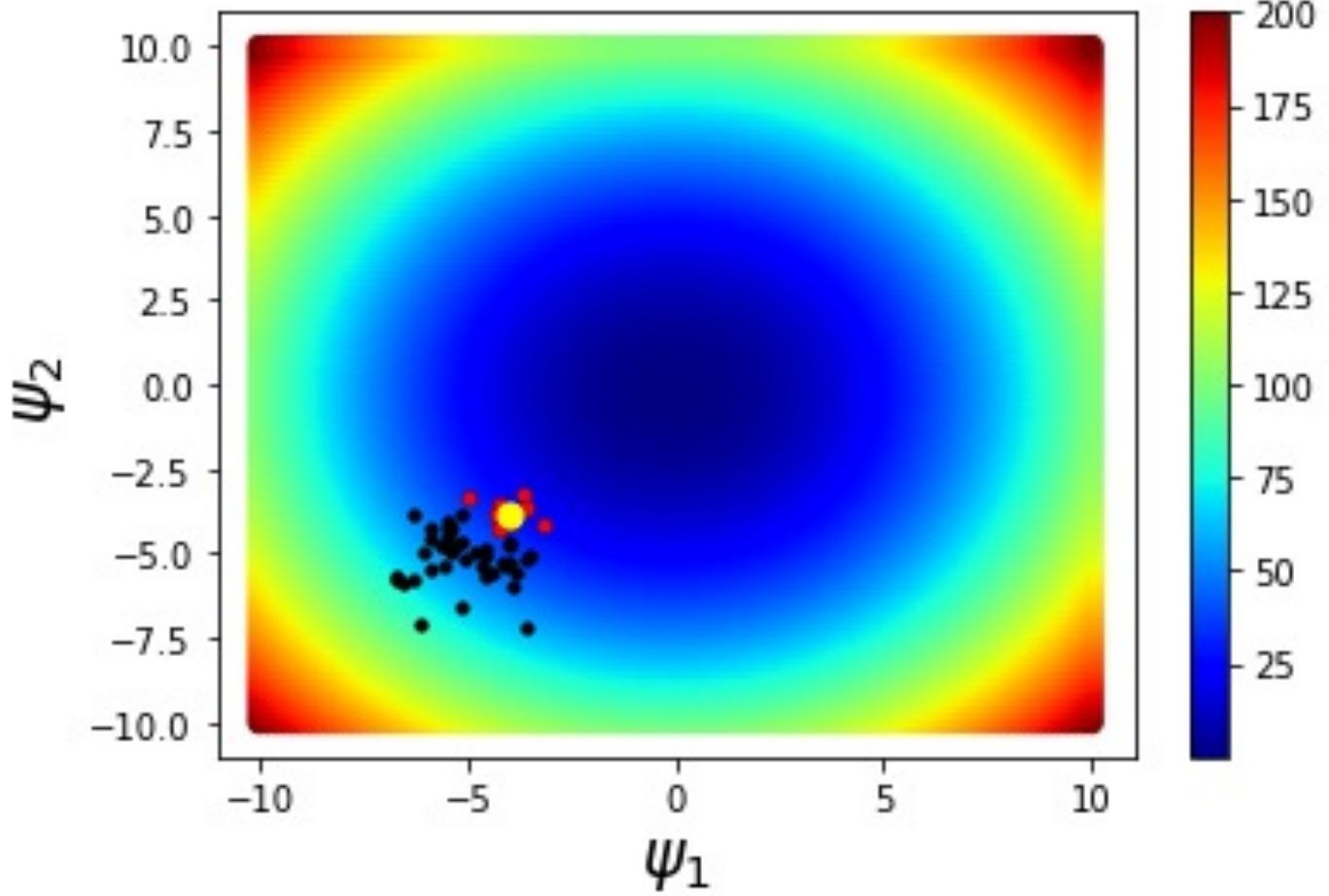
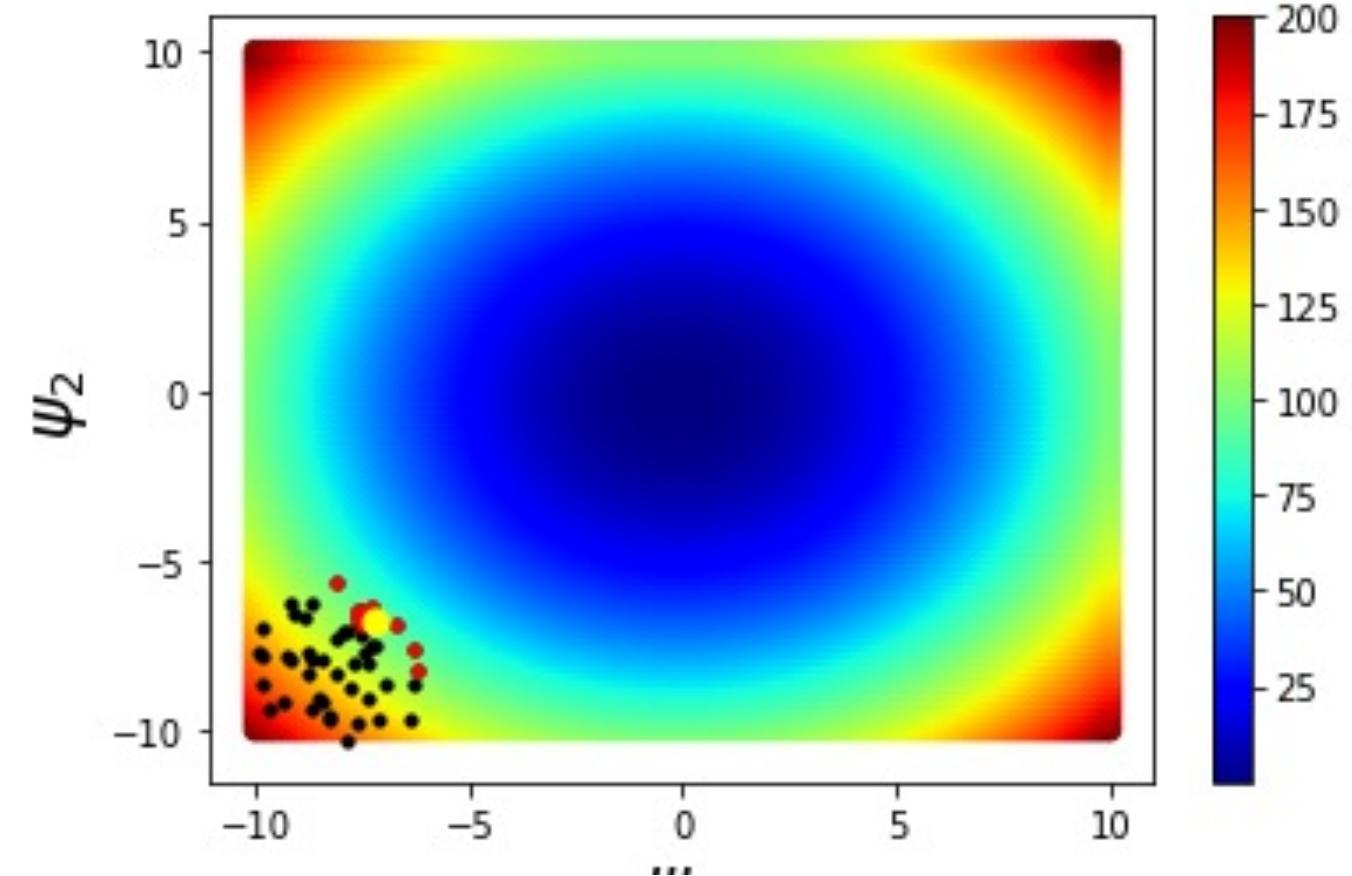
Require: Number K best samples

- 1: Generate initial population $D = \{\psi_t\}$ randomly
 - 2: **while** computationally feasible **do**
 - 3: Compute $f(\psi)$ for each point in D
 - 4: Select K best points ψ corresponding to K minimal values of $f(\psi)$
 - 5: Breed the K best selected points
 - 6: Replace the least fit samples from D with K breed points
 - 7: **end while**
-

Evolutionary strategies

- Simple case of Gaussian ES:
 - Set $\theta = (\mu)$, $p_\theta(\psi) = N(\mu, \sigma^2 I)$
 - Sample M values of ψ_i , compute $f(\psi_i)$
 - Select best K values by sorting $f(\psi_i)$
 - Update μ using selected ψ_i
- Usually requires large number of sample M .
- Modifications such as CMA-ES[1] or Guided ES[2] might utilise surrogate gradient information.

[1] <http://www.cmap.polytechnique.fr/~nikolaus.hansen/cmaartic.pdf>, [2] <https://arxiv.org/abs/1806.10230>



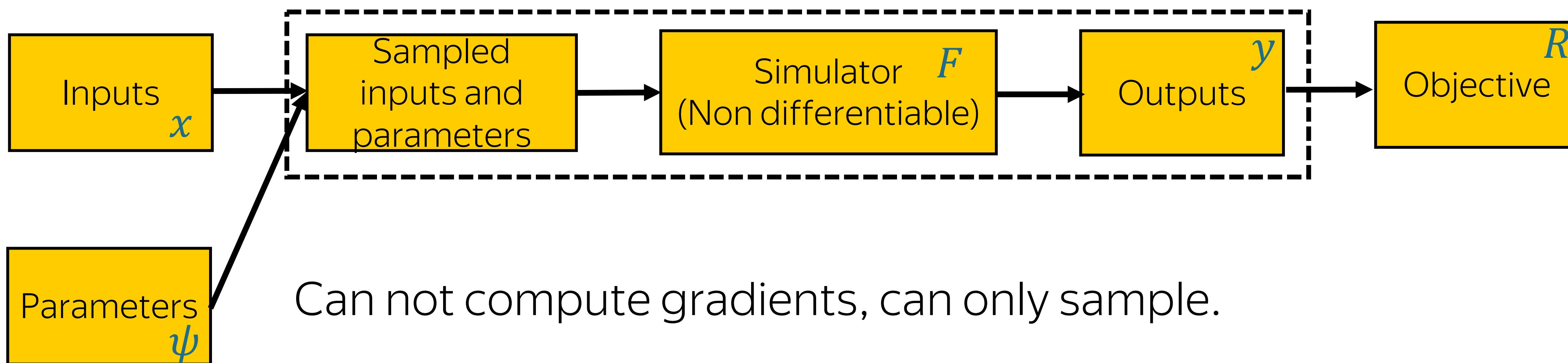
Why to create something new?

- Require frequent simulator calls → computationally expensive
- Require prior distribution or search region
- May not scale well to high dimensions
- Have high variance
- Estimate only first order gradients

We try to solve some of those issues with our method.

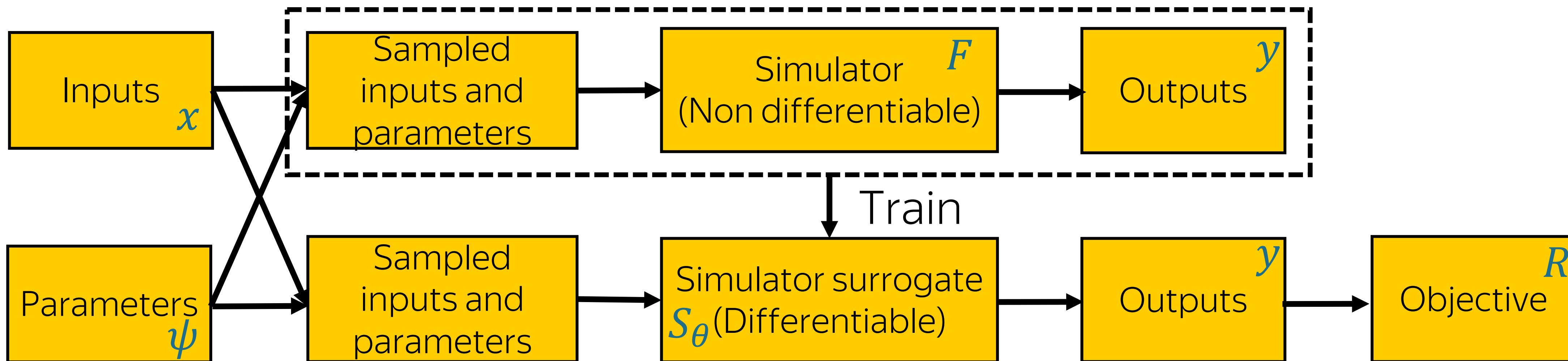
Generative surrogates

$$E_y[R(y_\psi)] \approx \frac{1}{N} \sum_{i=1}^N R(F(x_i; \psi_i))$$



Generative surrogates

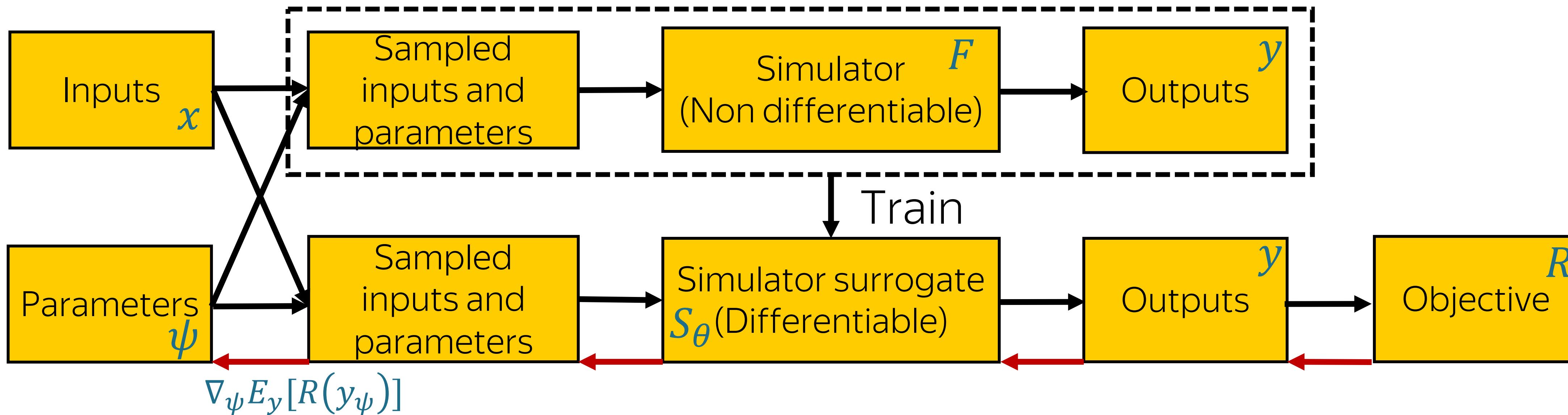
$$E_y[R(y_\psi)] \approx \frac{1}{N} \sum_{i=1}^N R(F(x_i; \psi_i))$$



$$E_y[R(y_\psi)] \approx \frac{1}{N} \sum_{i=1}^N R(S(z_i, x_i; \psi_i))$$

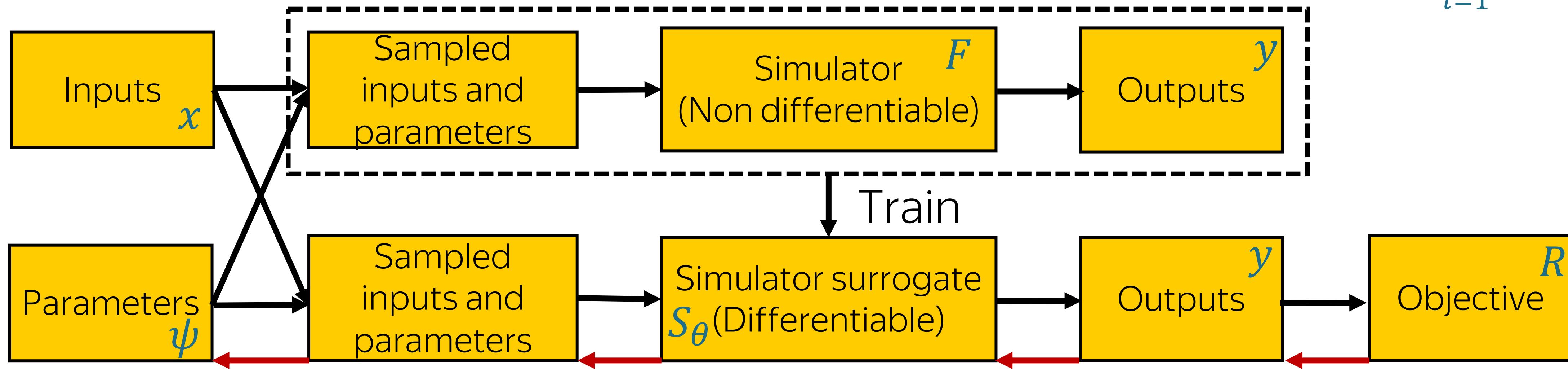
Generative surrogates

$$\nabla_{\psi} E_y[R(y_{\psi})] \approx \frac{1}{N} \sum_{i=1}^N \boxed{\nabla_{\psi} R(F(x_i; \psi_i))} \approx \frac{1}{N} \sum_{i=1}^N \boxed{\nabla_{\psi} R(S(z_i, x_i; \psi_i))}$$



Generative surrogates

$$\sum_{i=1}^N \nabla_\psi R(S_\theta(z_i, x_i; \psi))$$

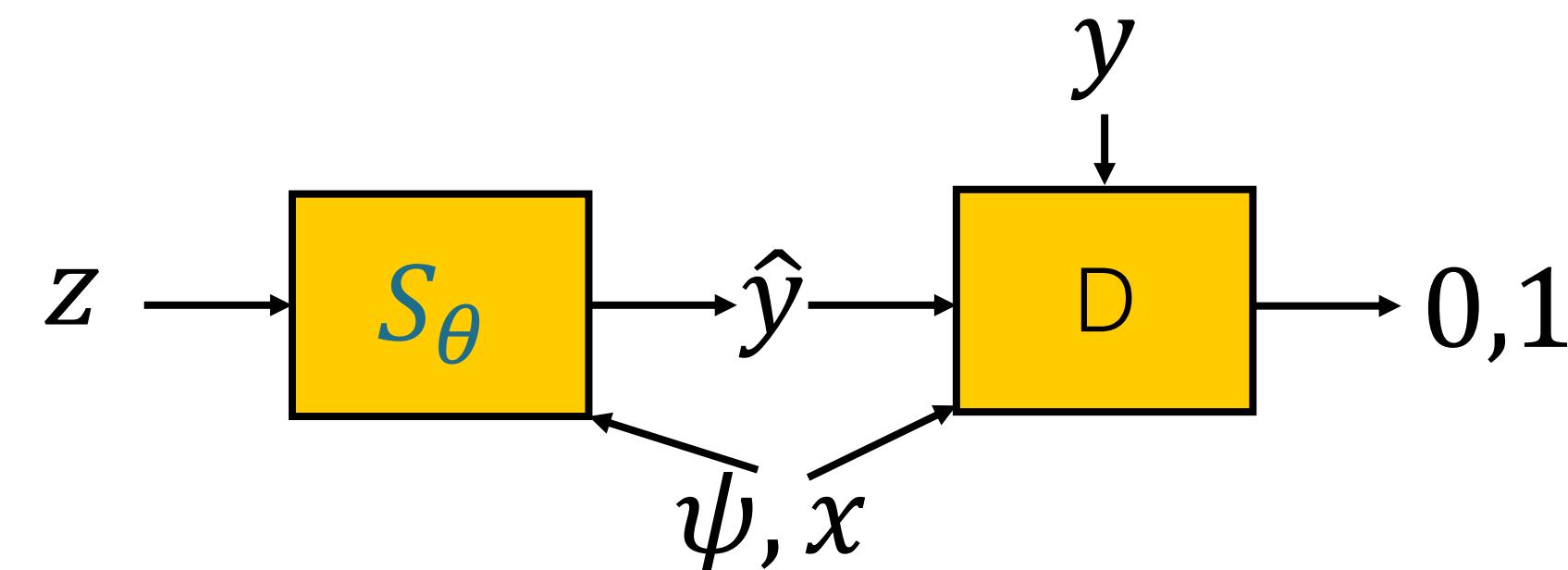


- S_θ any conditional deep generative model: GAN, NF, VAE, ...
- Once trained produce differentiable samples:

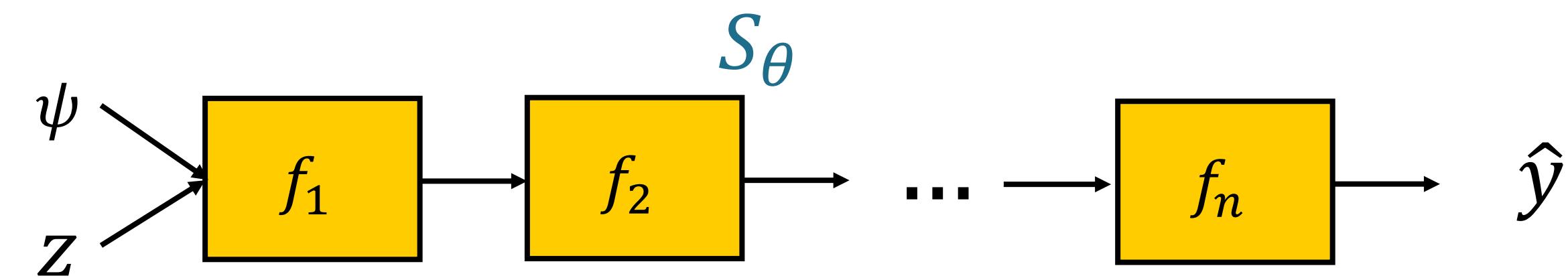
$$\nabla_\psi E_y[R(y)] \sim \sum \frac{\partial R}{\partial y_i} \times \boxed{\frac{\partial y_i}{\partial \psi}} = \sum \frac{\partial R}{\partial y_i} \times \boxed{\frac{\partial S_\theta}{\partial \psi}}$$

Conditional Generative Networks

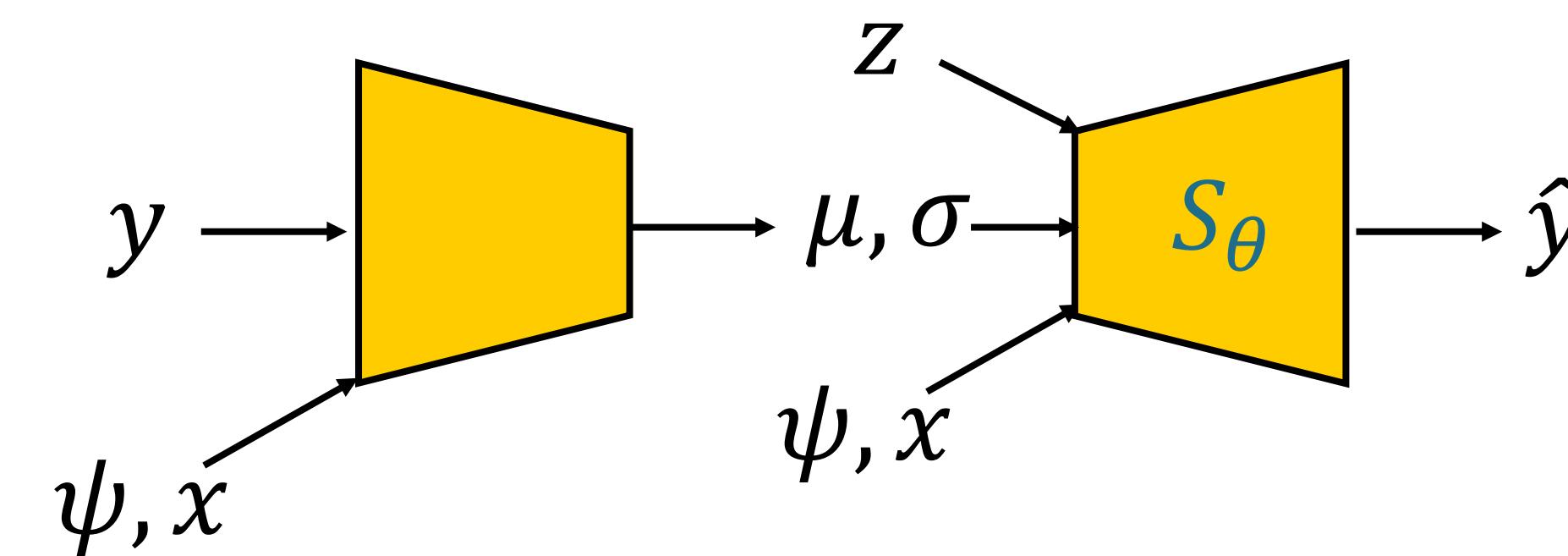
- Generative Adversarial Networks



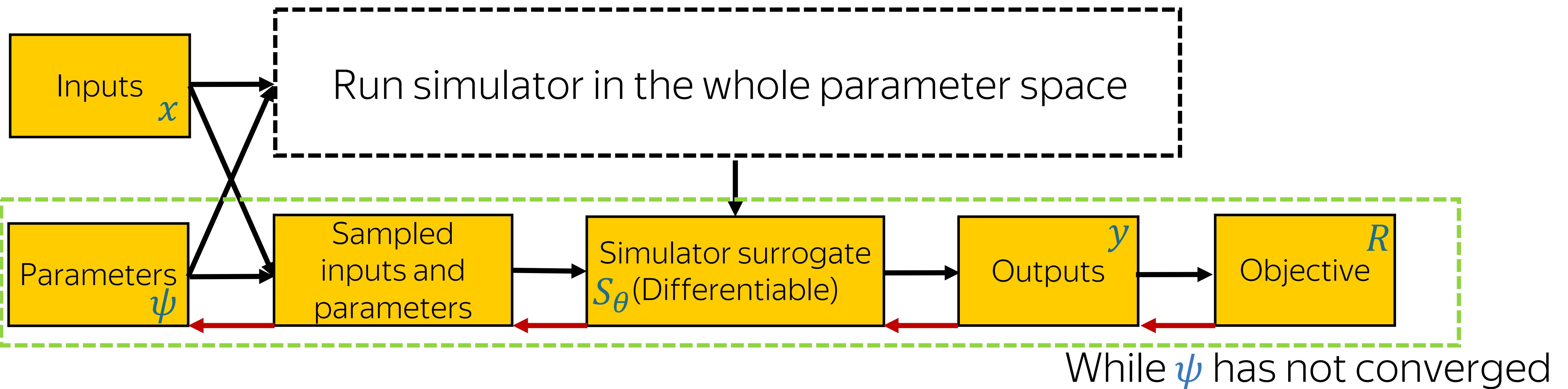
- Normalising flows



- Variational Autoencoders



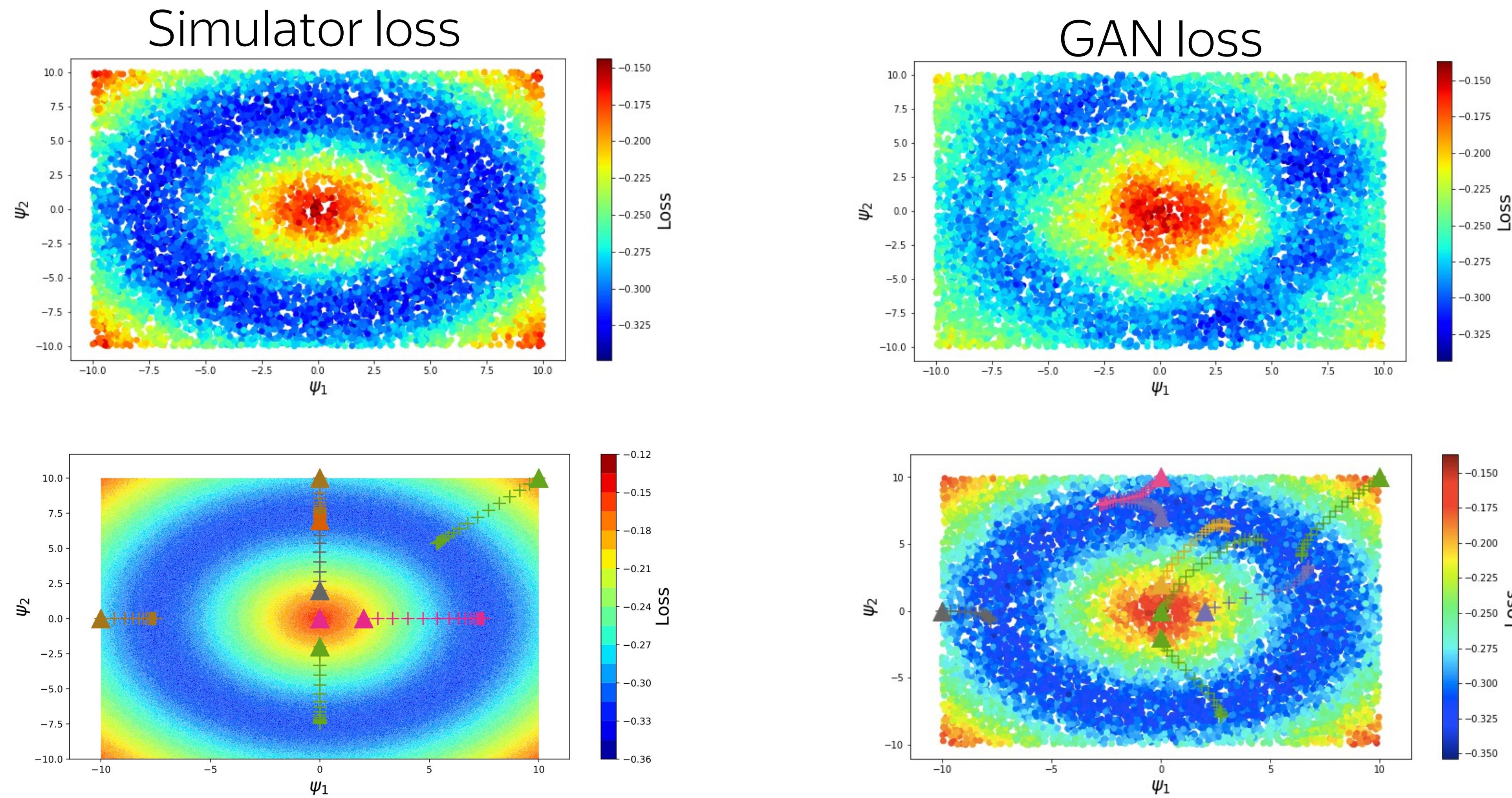
Generative surrogates: training



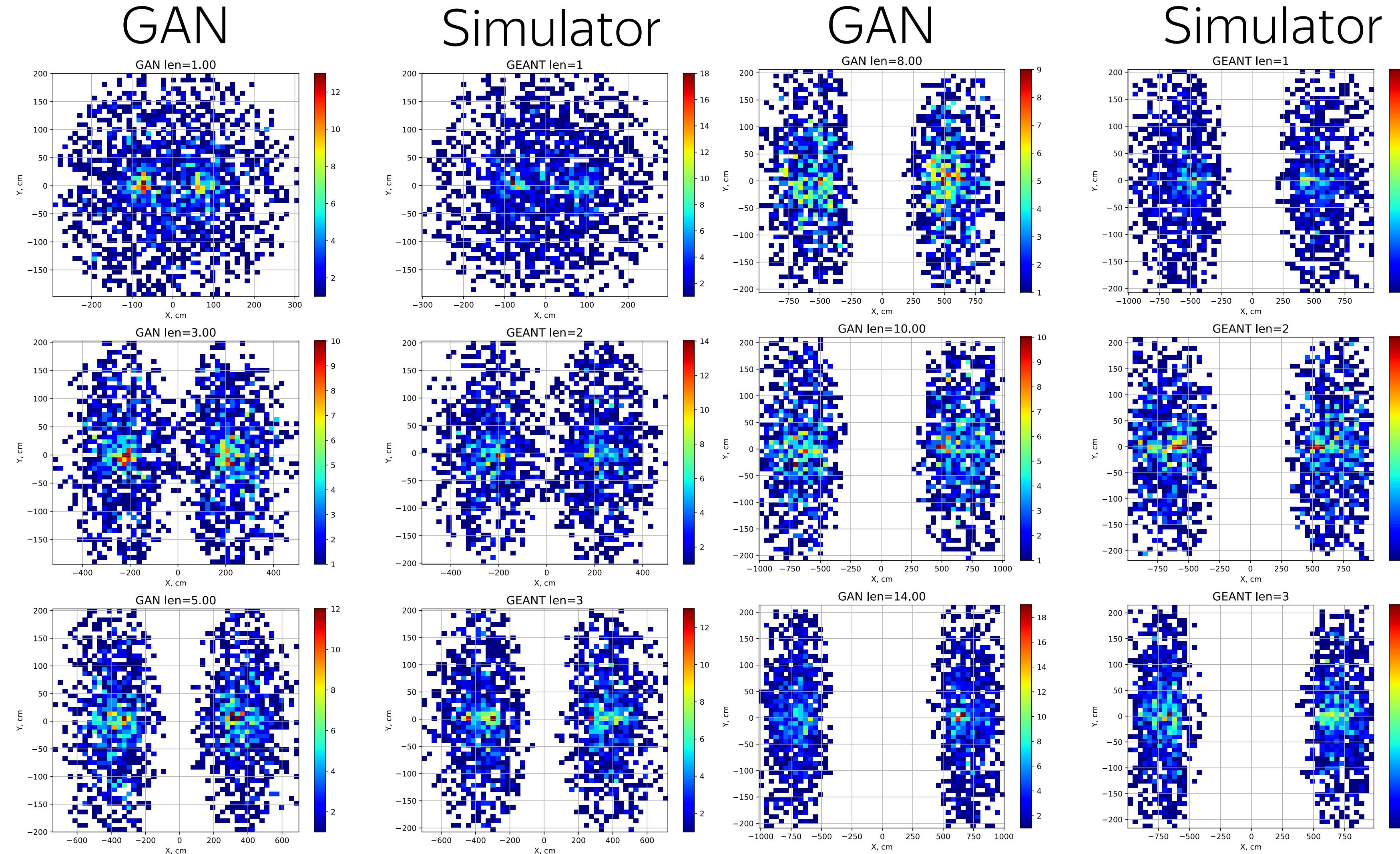
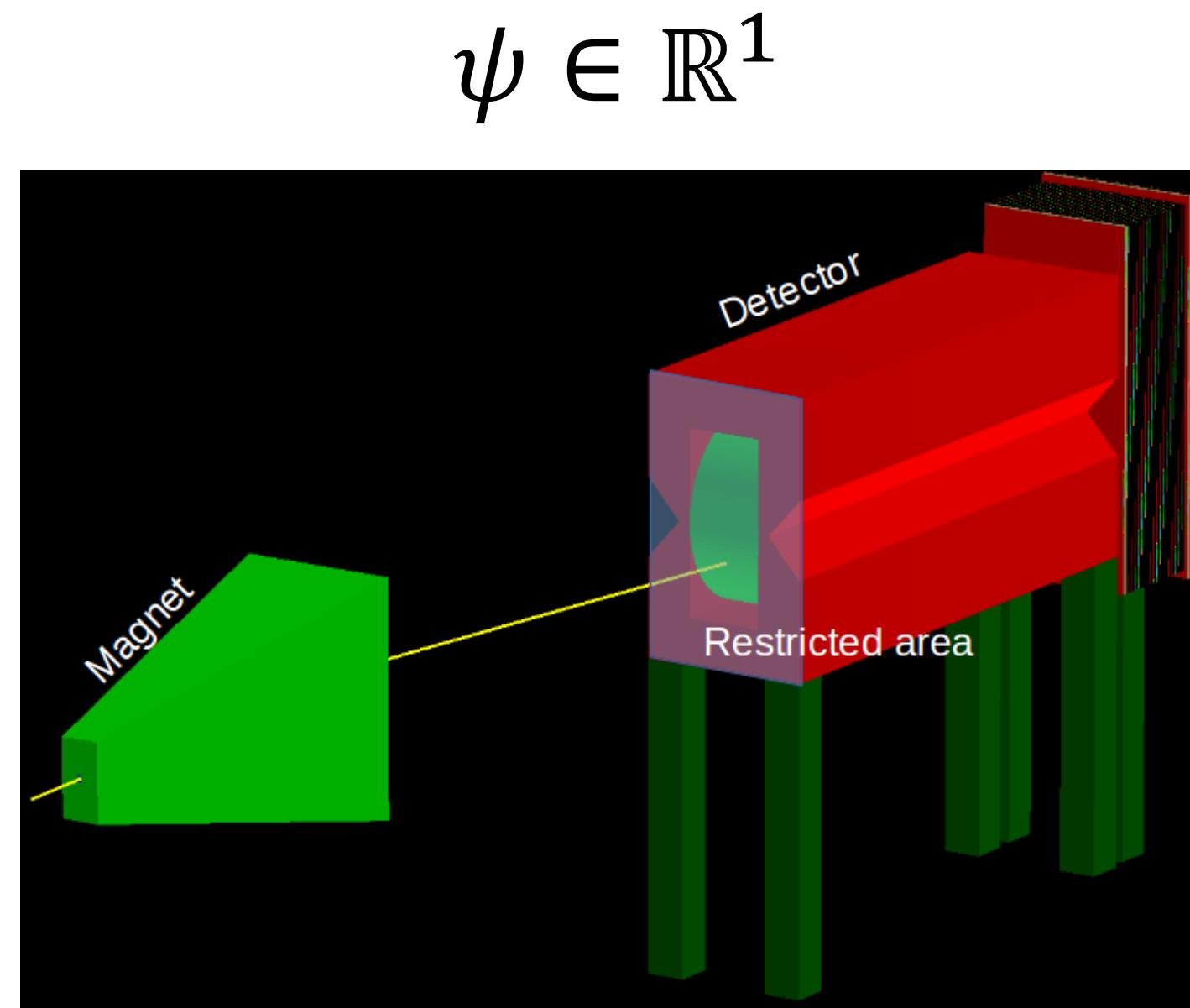
- Generate ψ on the grid
- S_θ is trained *once* in the *whole* space of parameters ψ

$$\psi_{t+1} = \psi_t - \text{const} * \frac{1}{N} \sum_{i=1}^N \nabla_\psi R(S(z_i, x_i; \psi_i))$$

Generative surrogates: toy example



Generative surrogates: physics toy example

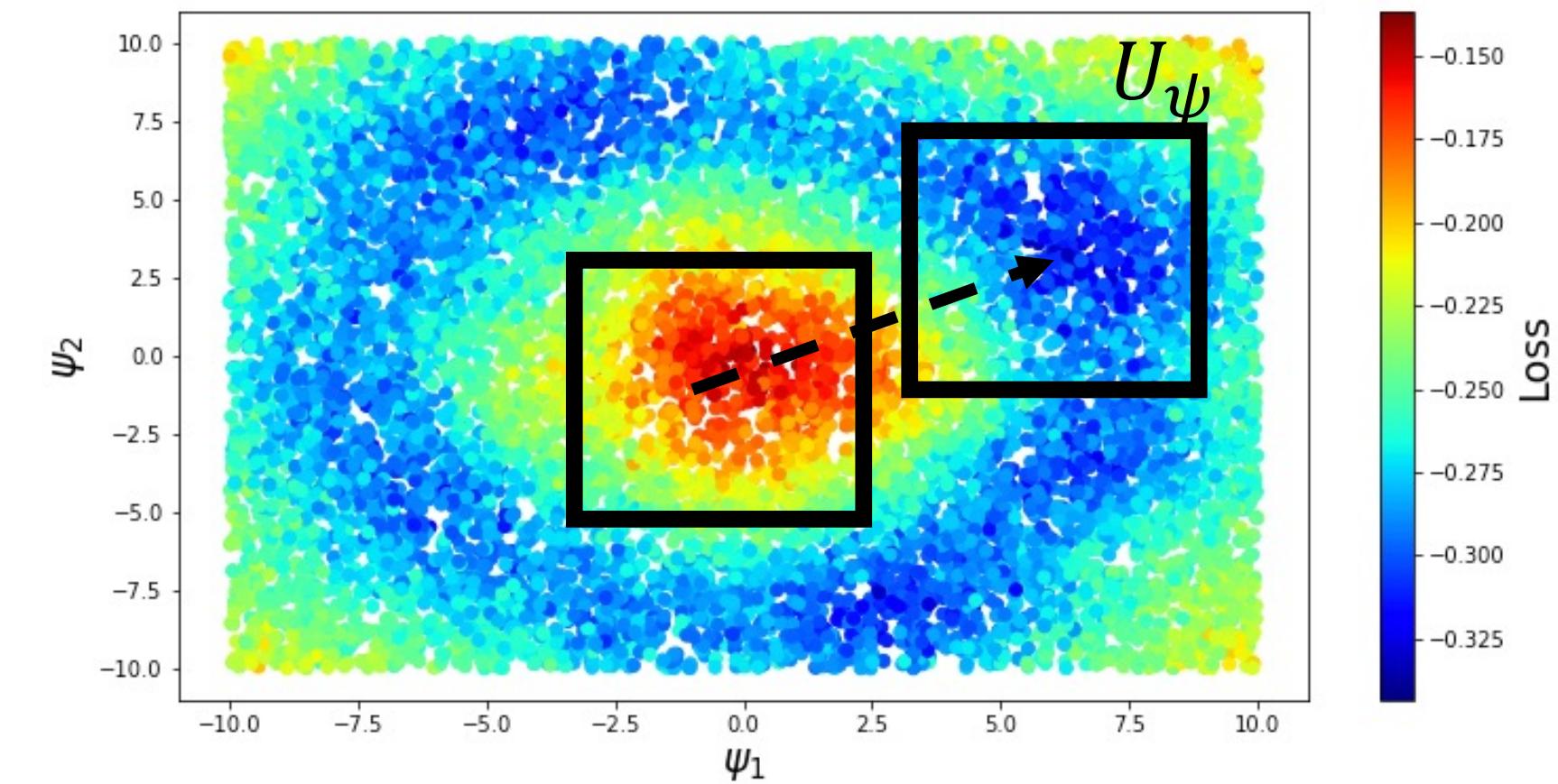


There is no locality...

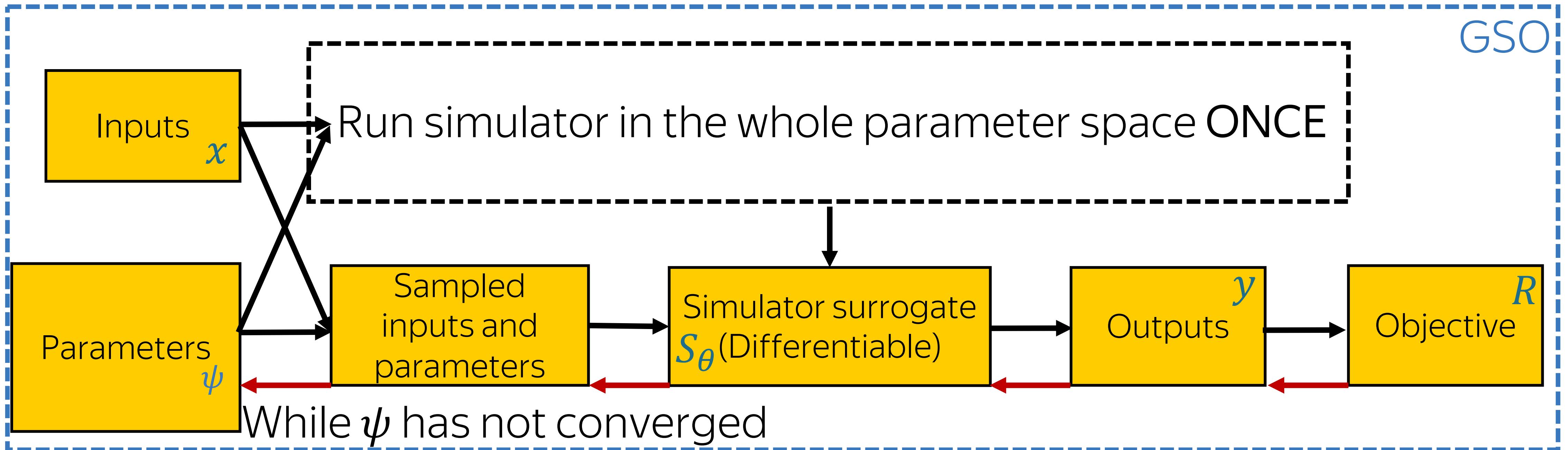
S_θ is trained in the whole ψ space:

- Curse of dimensionality
- Number of samples scales exponentially with dimension d : $\left(\frac{L}{\Delta}\right)^d$
- Generative models do not extrapolate well

Need to impose locality in the ψ space

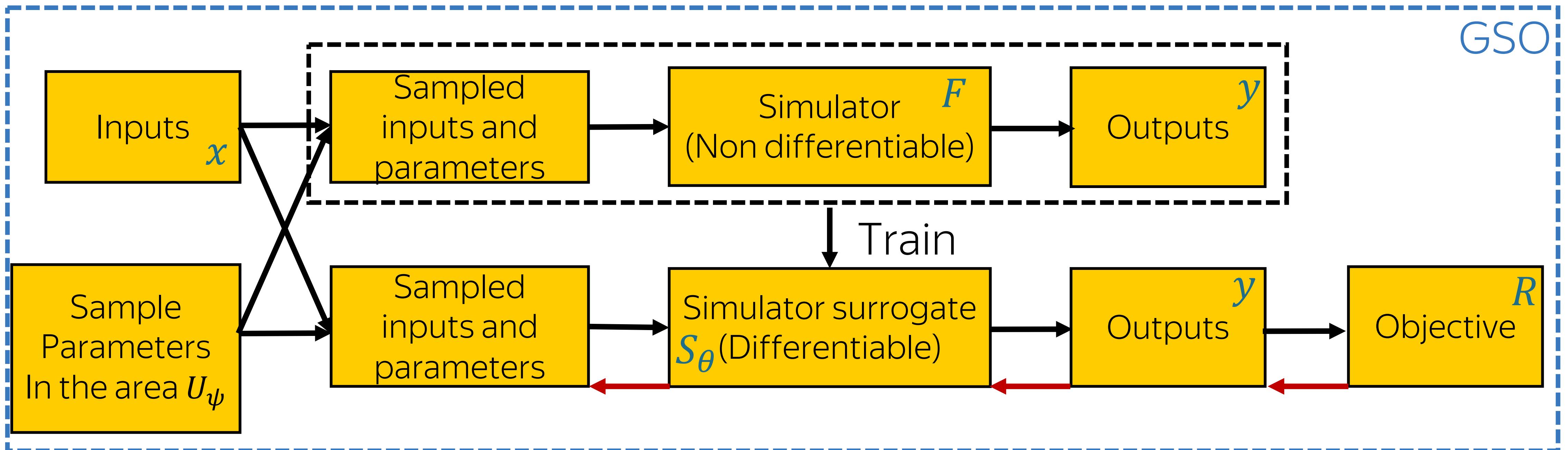


Local Generative Surrogates (L-GSO)



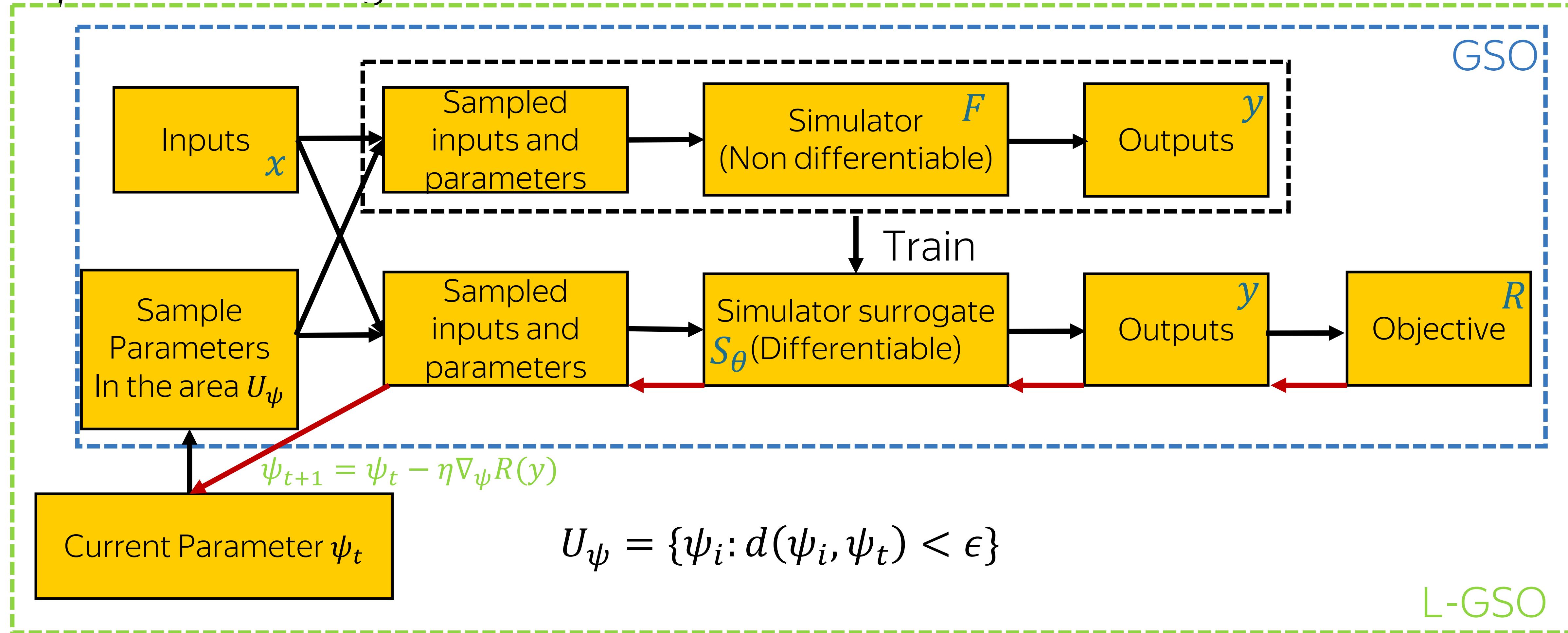
Local Generative Surrogates (L-GSO)

While ψ has not converged do:

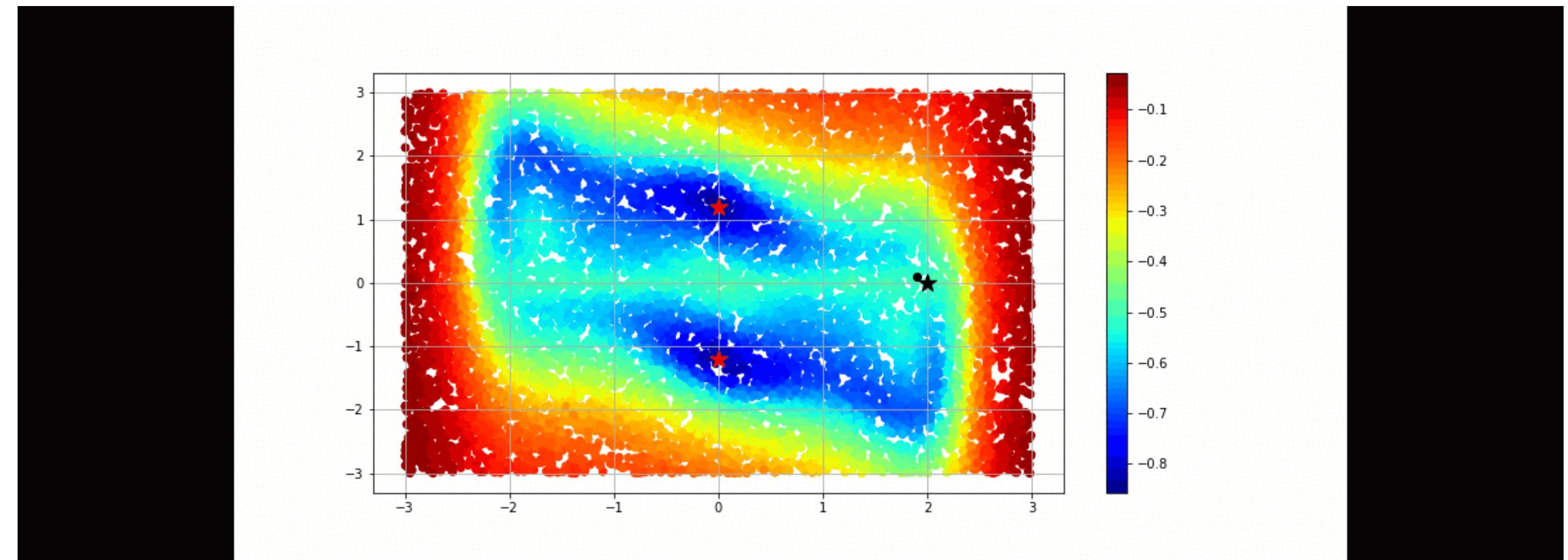


Local Generative Surrogates (L-GSO)

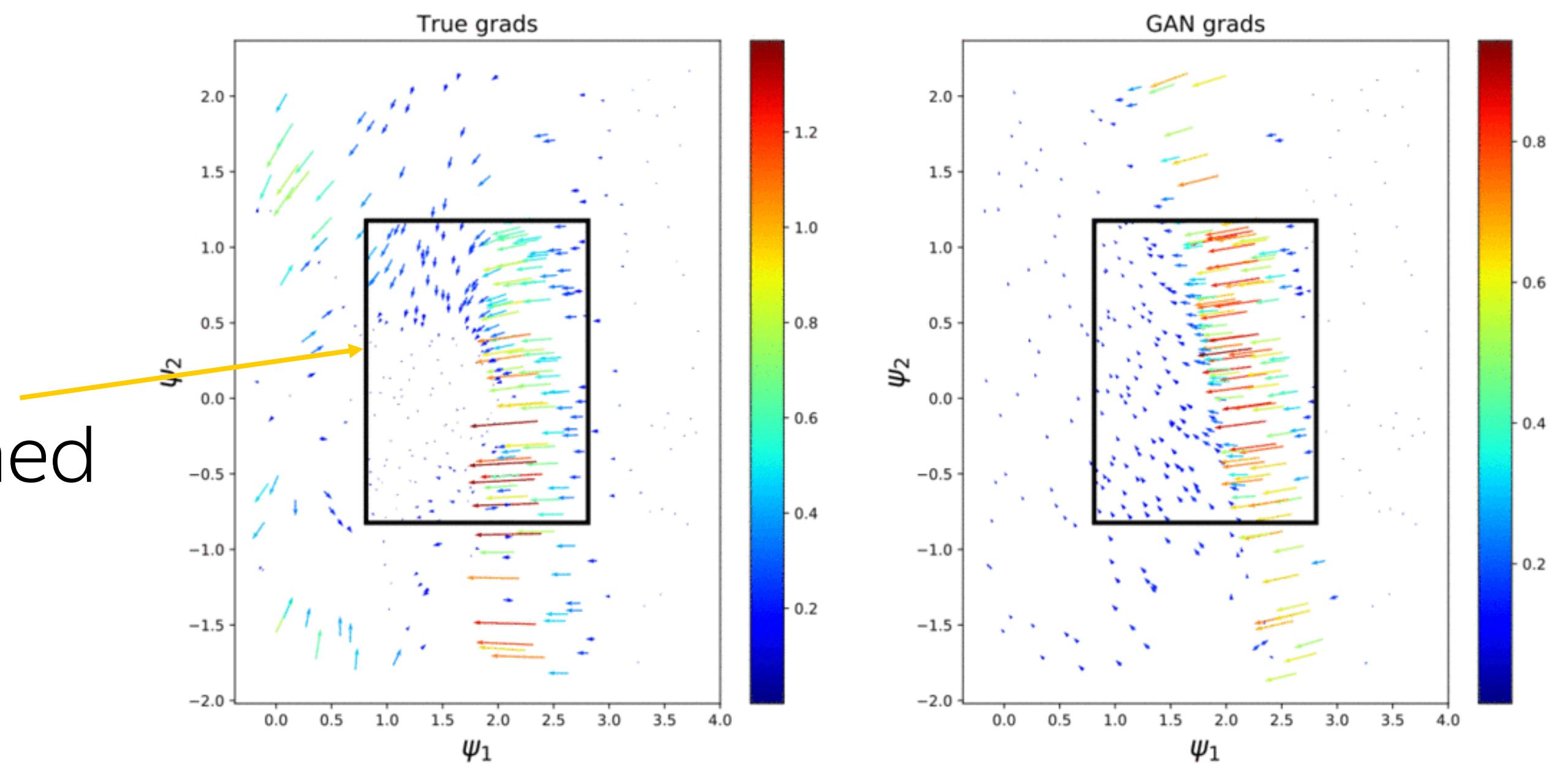
While ψ has not converged do:



Local Generative Surrogates (L-GSO)



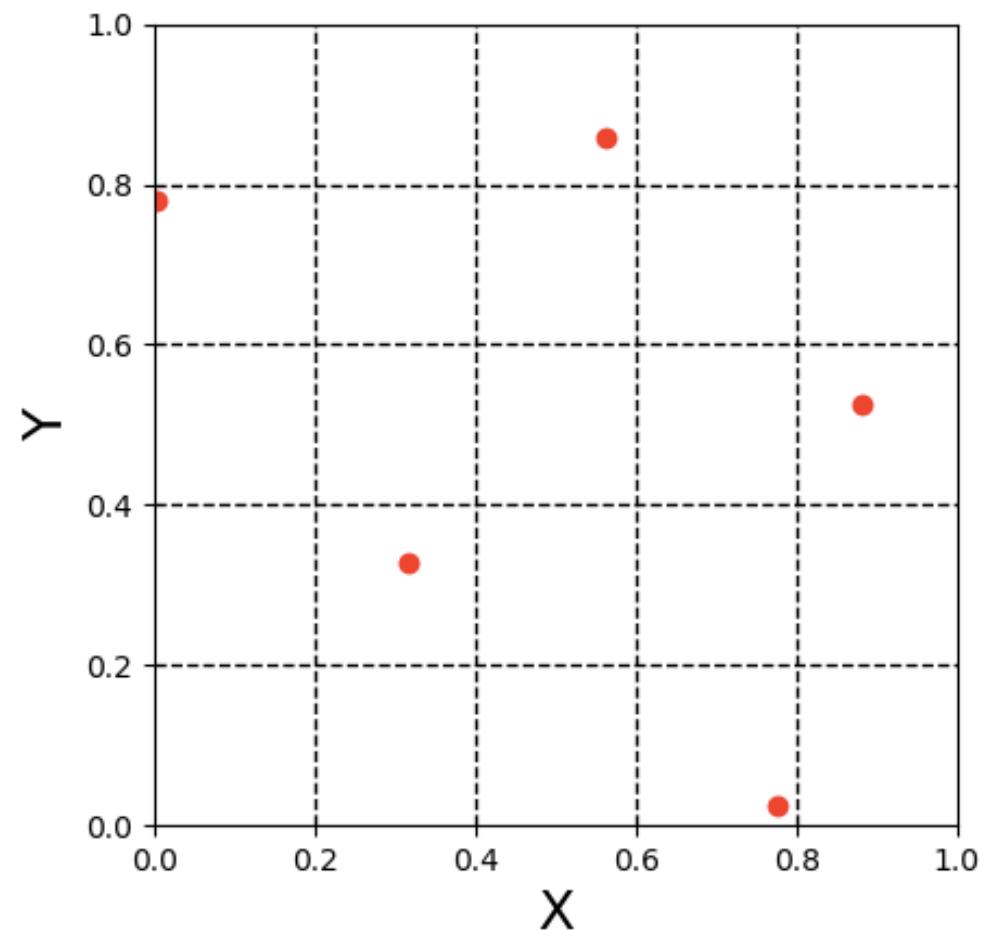
U_ψ where the
surrogate is trained



Gradients are well estimated in the local area

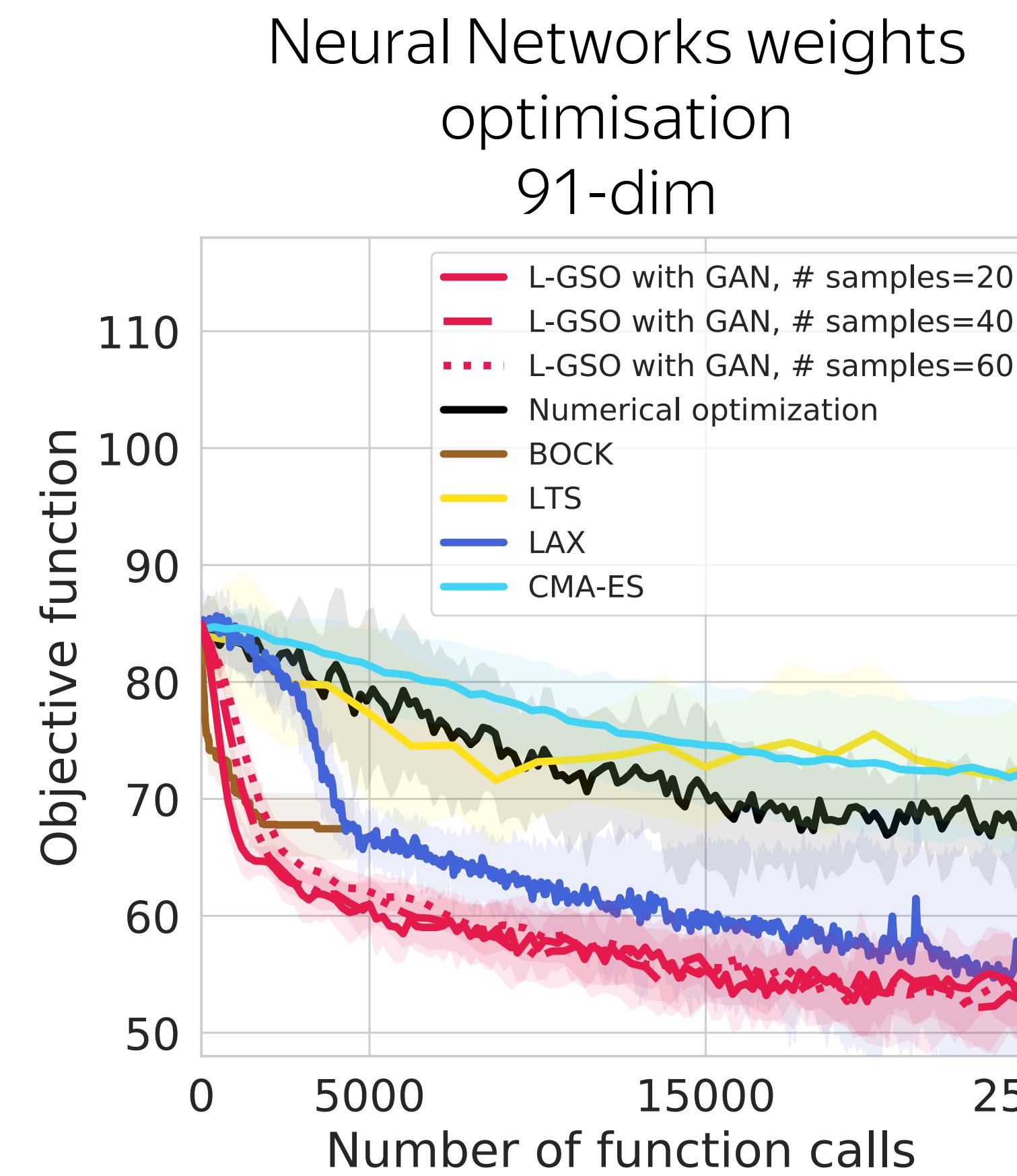
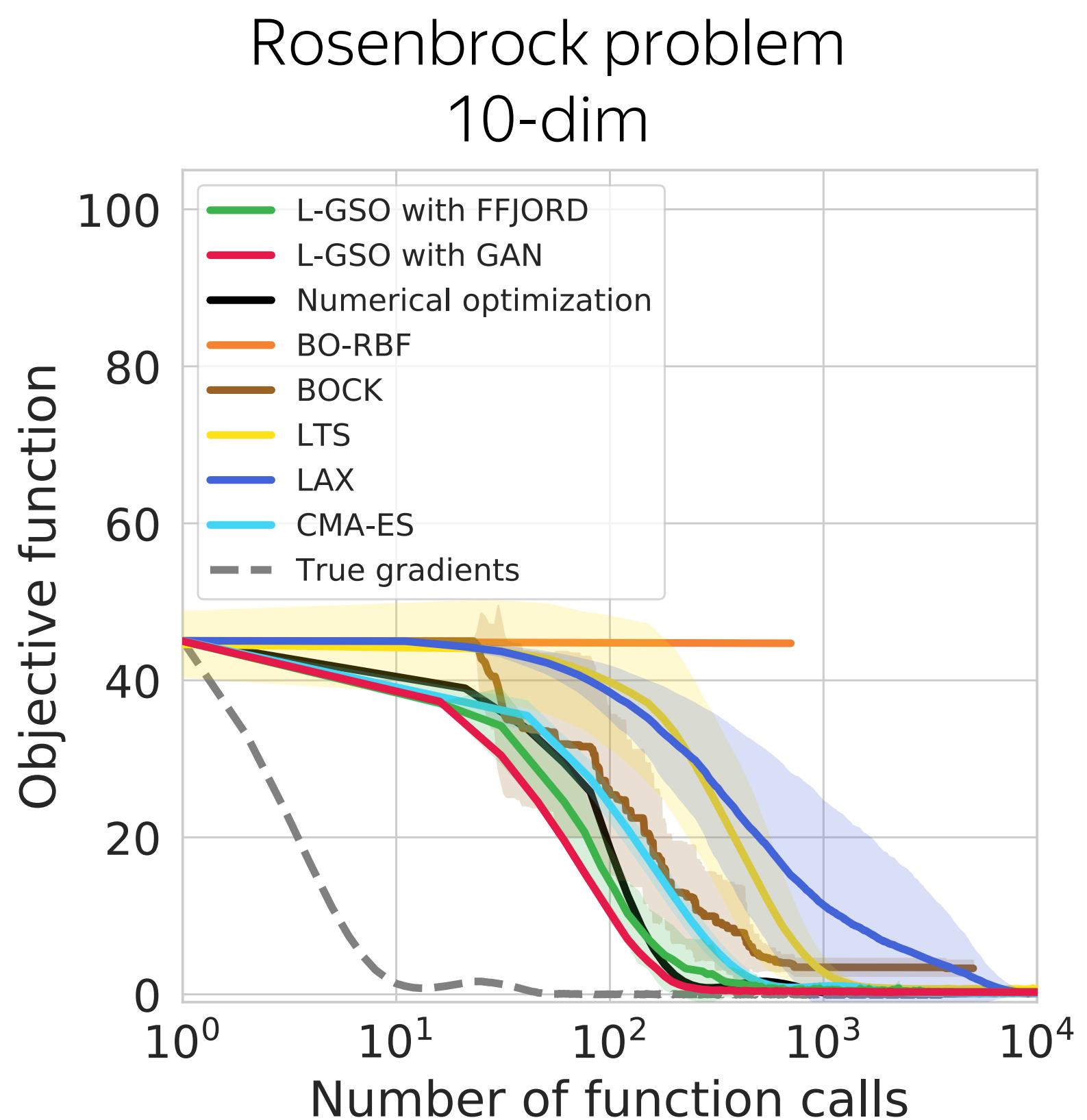
What is U_ψ ?

- $U_\psi = \{\psi_i : |\vec{\psi}_t - \vec{\psi}_i| < \epsilon\}$
- Fill U_ψ with Latin Hypercubes sampling
- ϵ is fixed
- Select ϵ : $E[|R(y_{\psi+\epsilon}) - R(y_{\psi-\epsilon})|] > Var[R(y_\psi)]$

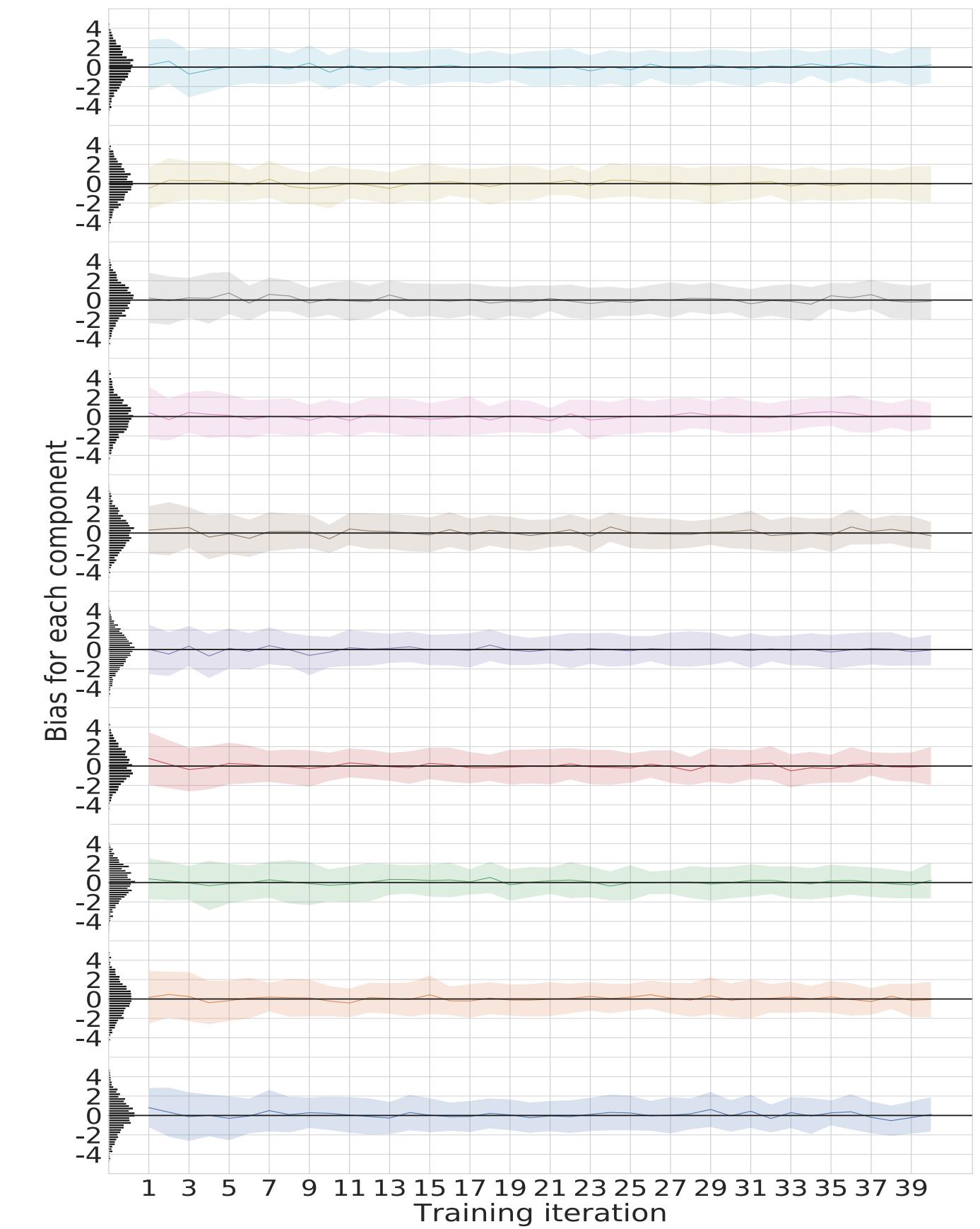
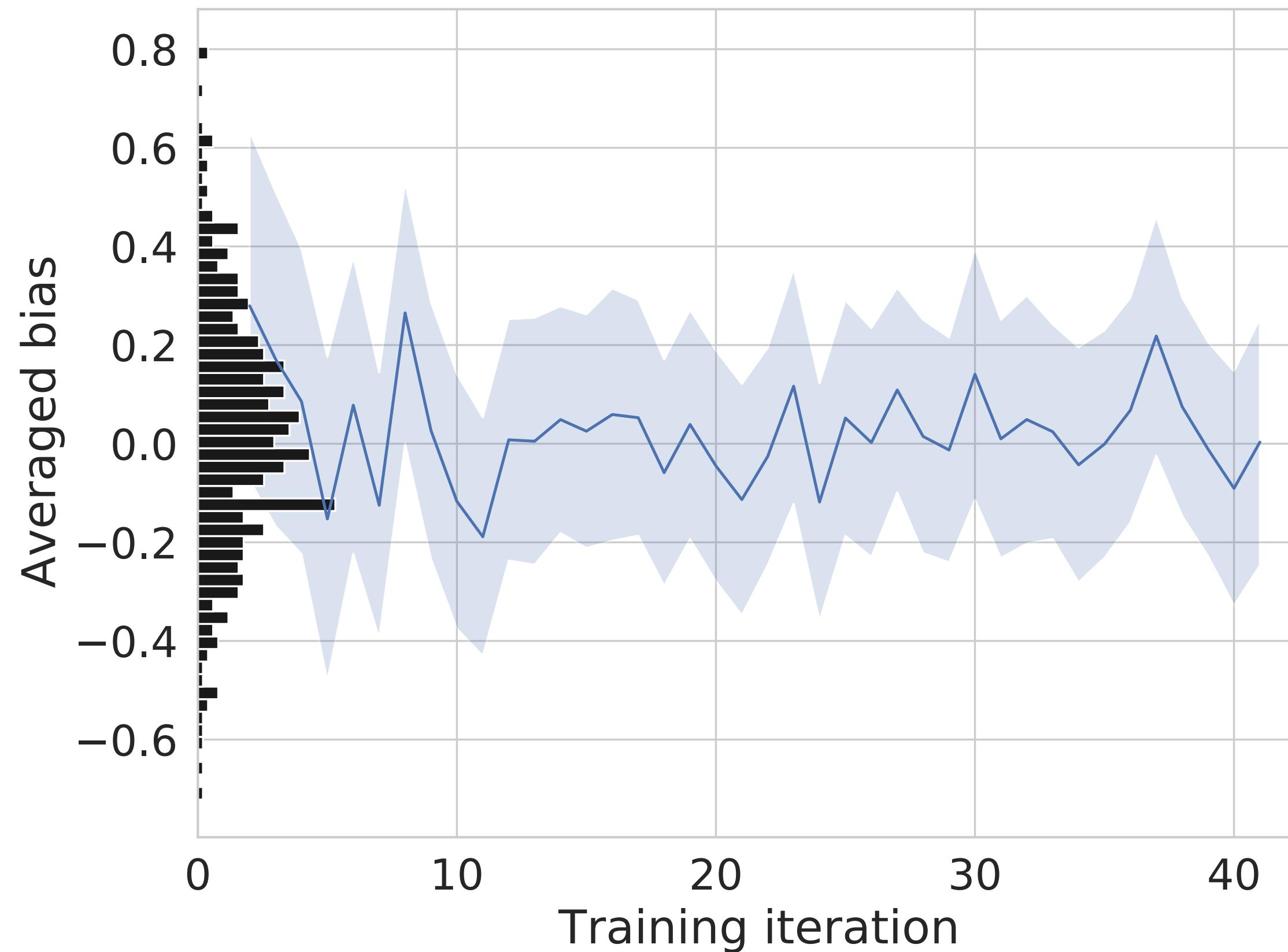


Toy Experiments

- 4 toy problems
- Various dimensions
- Degenerate parameters
- Costly simulator call
- Compare in:
 - number of calls
 - attained minimum



L-GSO bias



$$Bias_t = \nabla_{\psi|\psi_t} R(y_\psi) - \nabla_{\psi|\psi_t} R(\bar{y})$$

- No bias observed for L-GSO

SHiP: Shield optimisation

Muon kinematics, including start coordinate

$$x = \{P, \phi, \theta, Q, C\}, X \in \mathbb{R}^7$$

Output: coordinates of the muon hit

$$y = \{X, Y\}, y \in \mathbb{R}^2$$

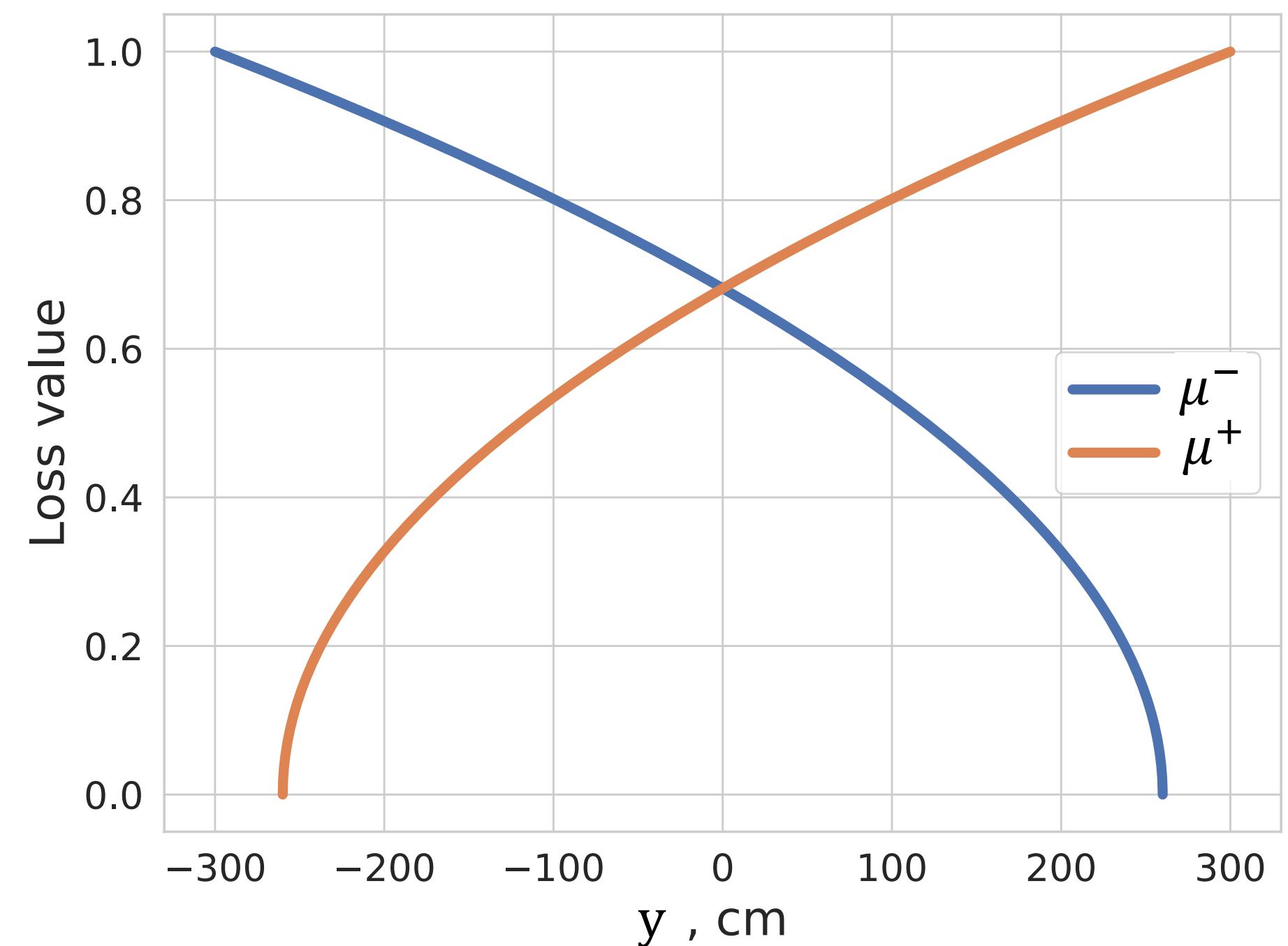
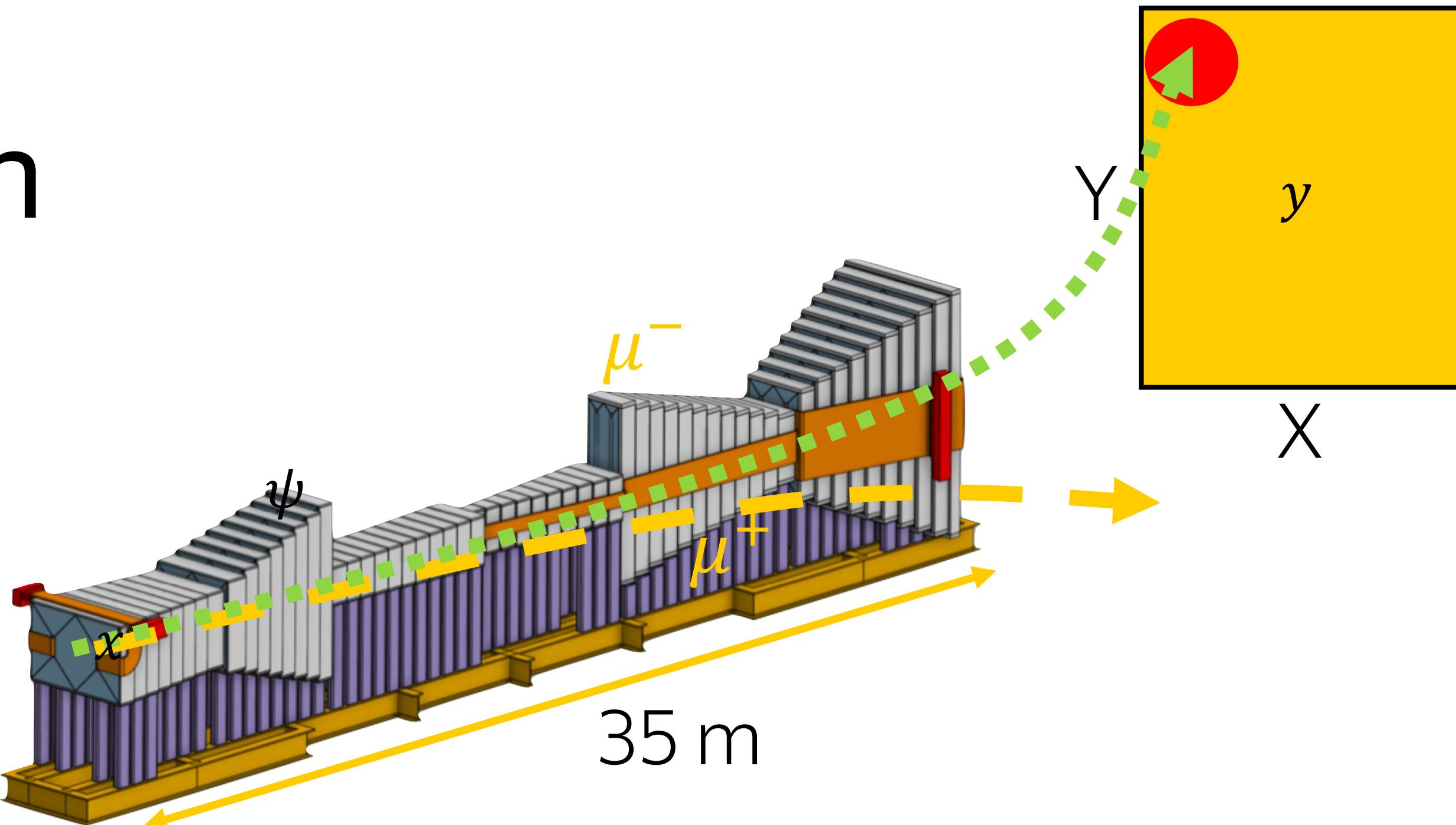
Optimised parameters: shield geometry

$$\psi \in \mathbb{R}^{42}$$

Objective function

$$R(y; \alpha) = \mathbf{1}_{Q=-1} \sqrt{(\alpha_1 - (y + \alpha_2))/\alpha_1} + \\ \mathbf{1}_{Q=1} \sqrt{(\alpha_1 + (y - \alpha_2))/\alpha_1}$$

Was previously optimised with BO



SHiP: Parameter changes

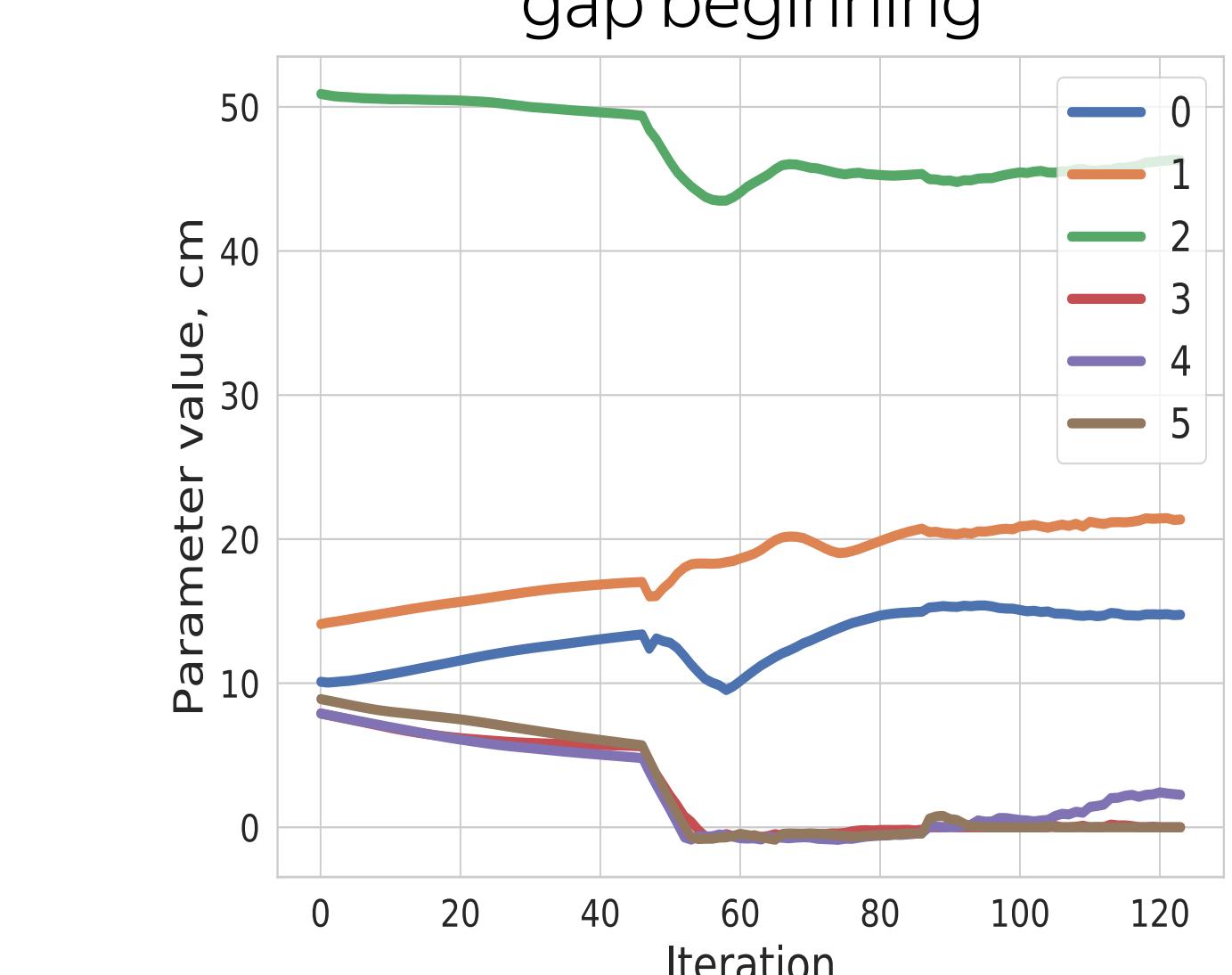
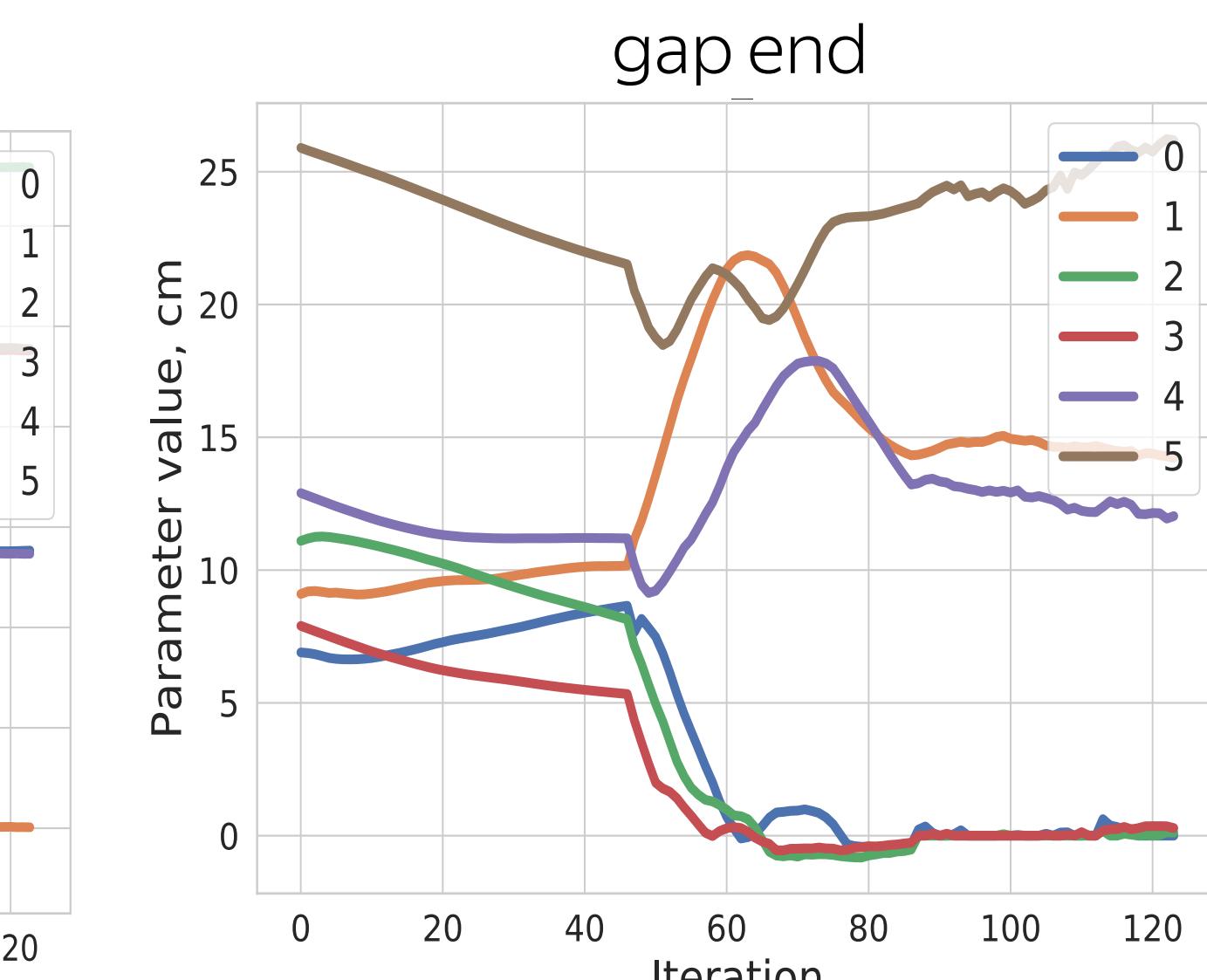
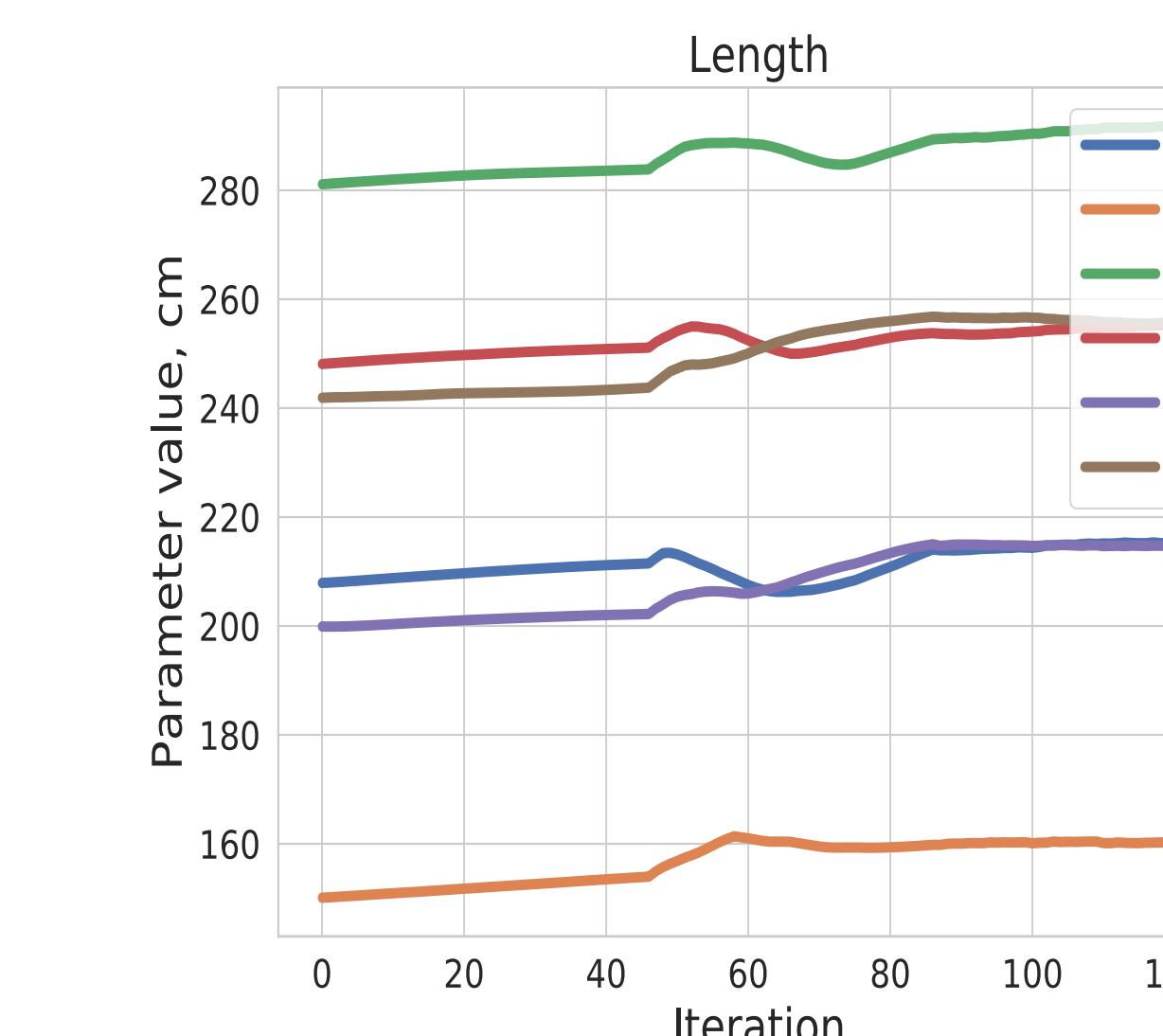
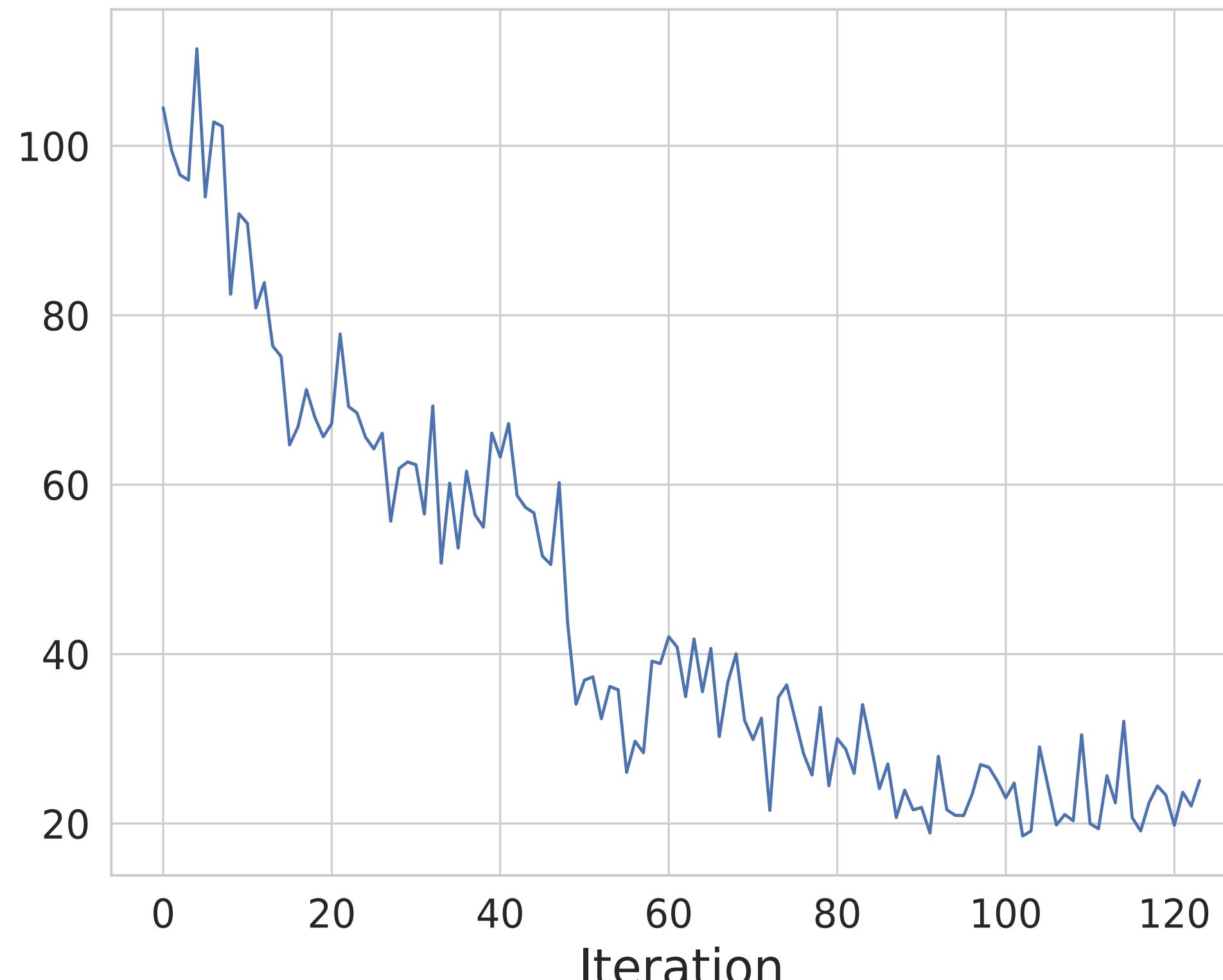
Loss

Parameter value, cm

Iteration

gap end

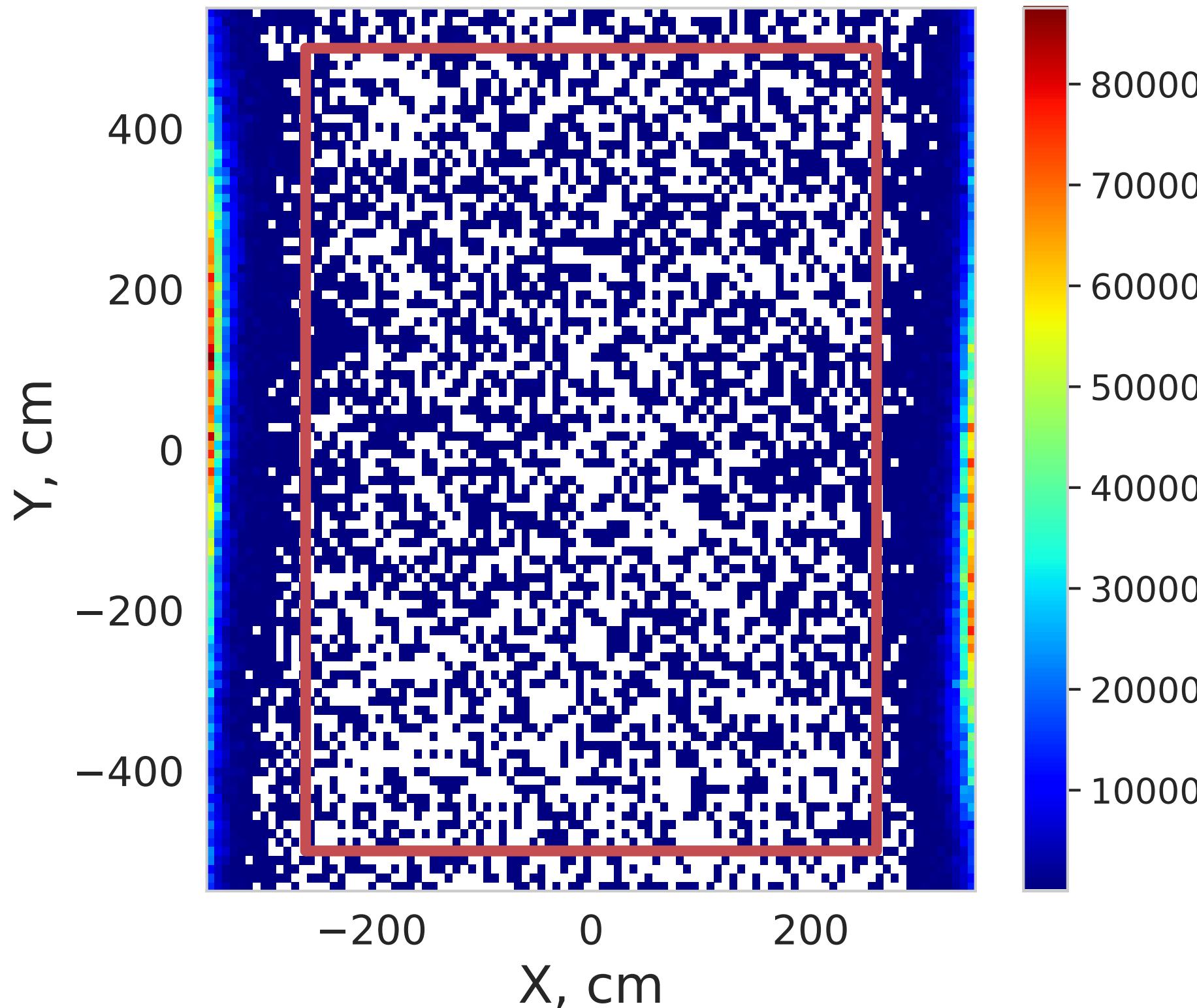
gap beginning



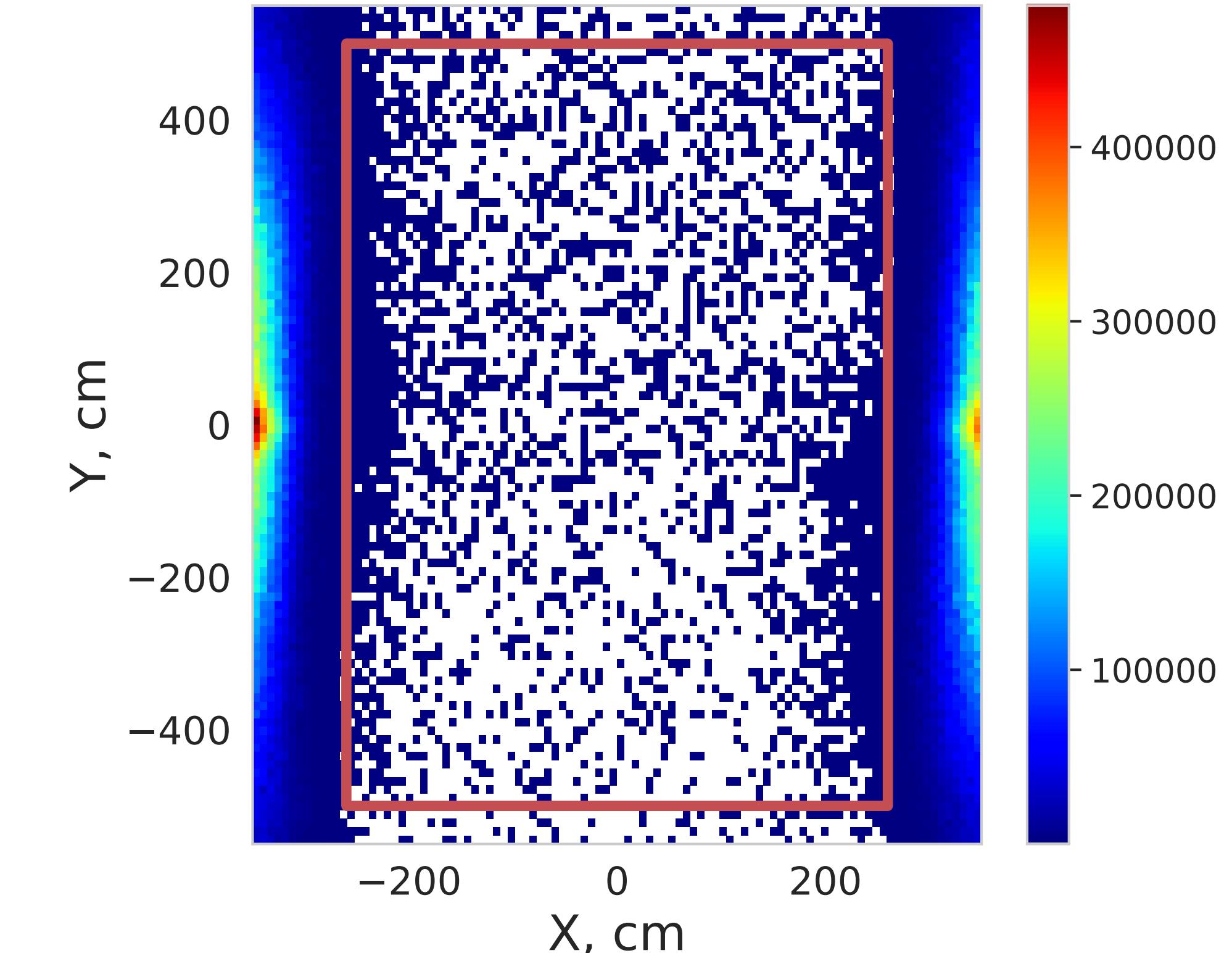
Continuous changes of the parameters:
might give some insights about physics!

SHiP: shield optimisation comparison

Previous optimum(BO)



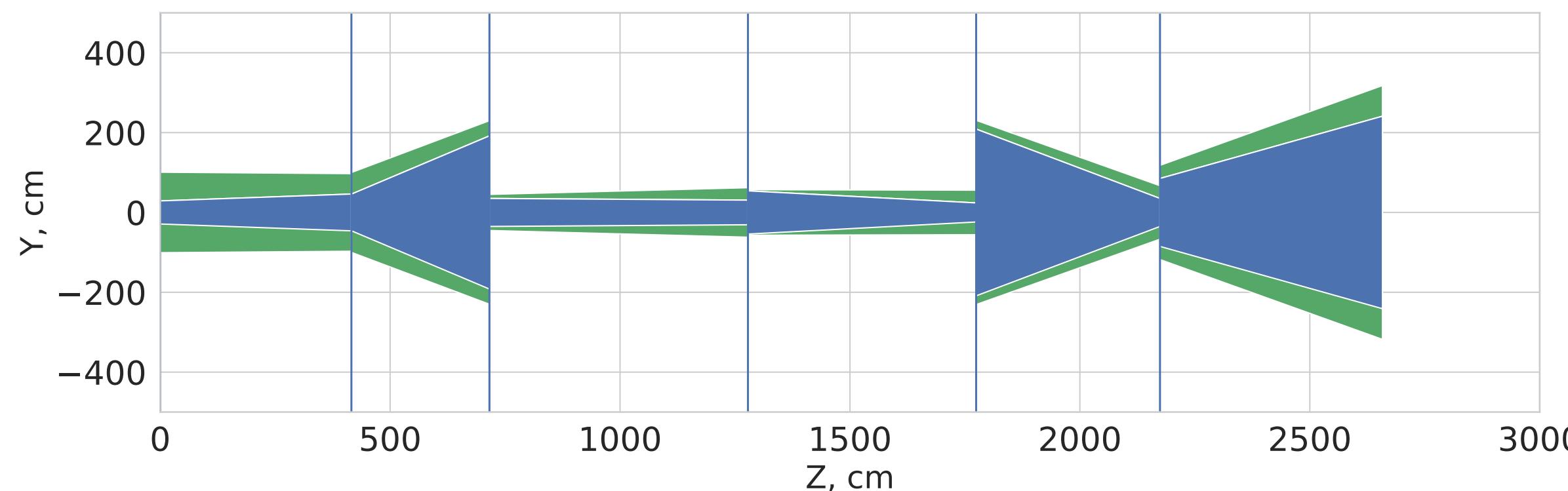
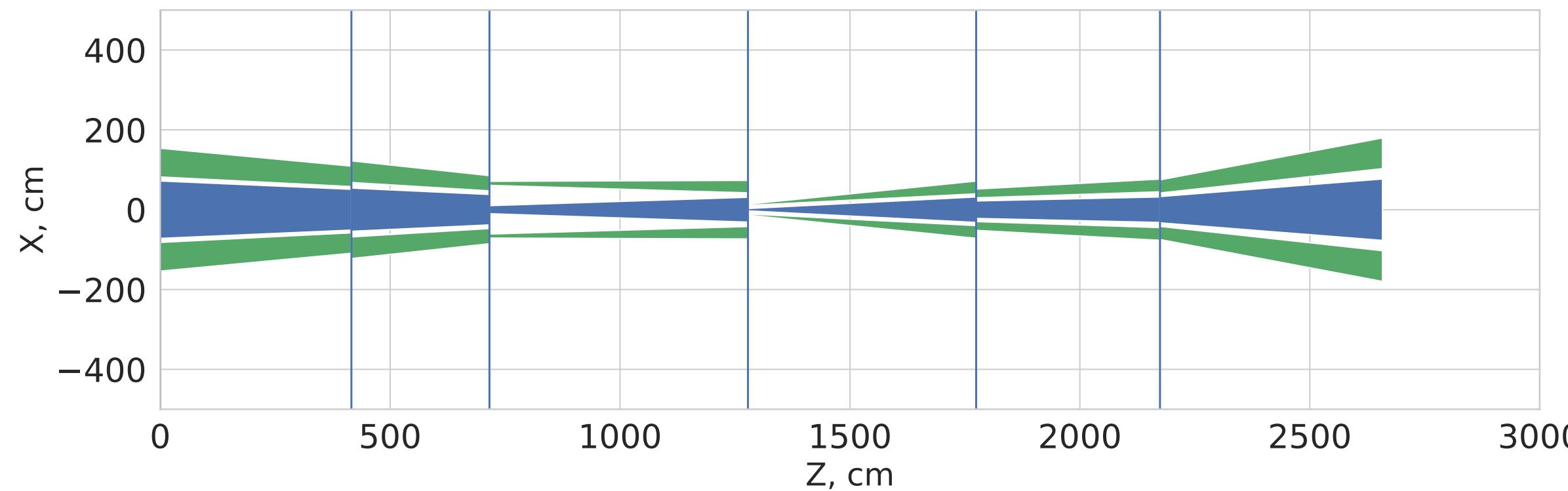
New optimum(L-GSO)



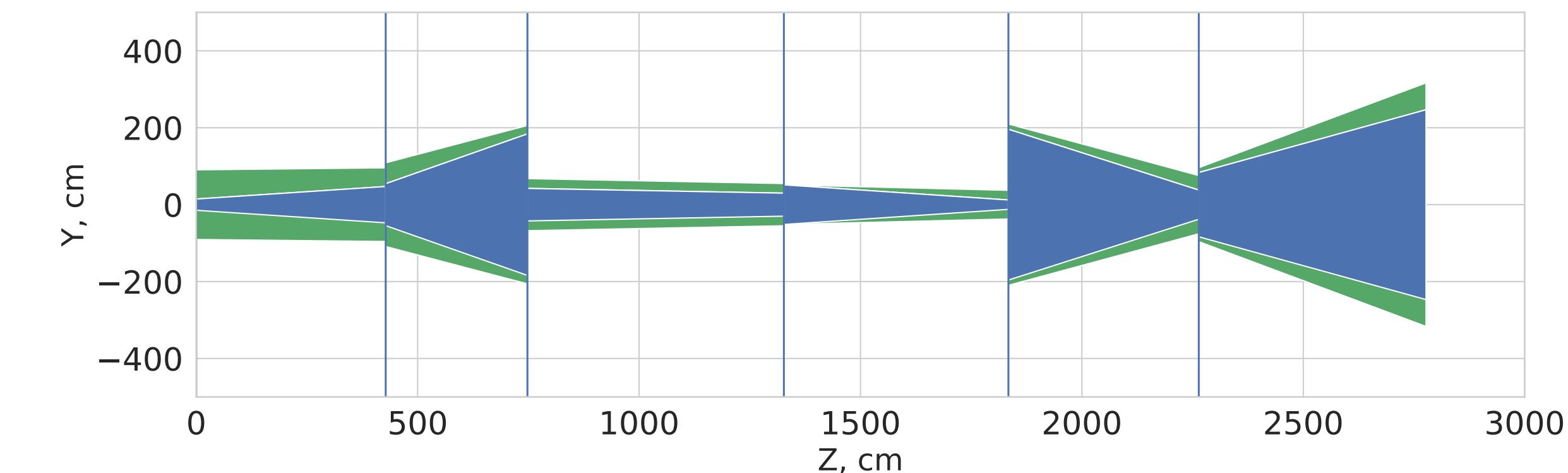
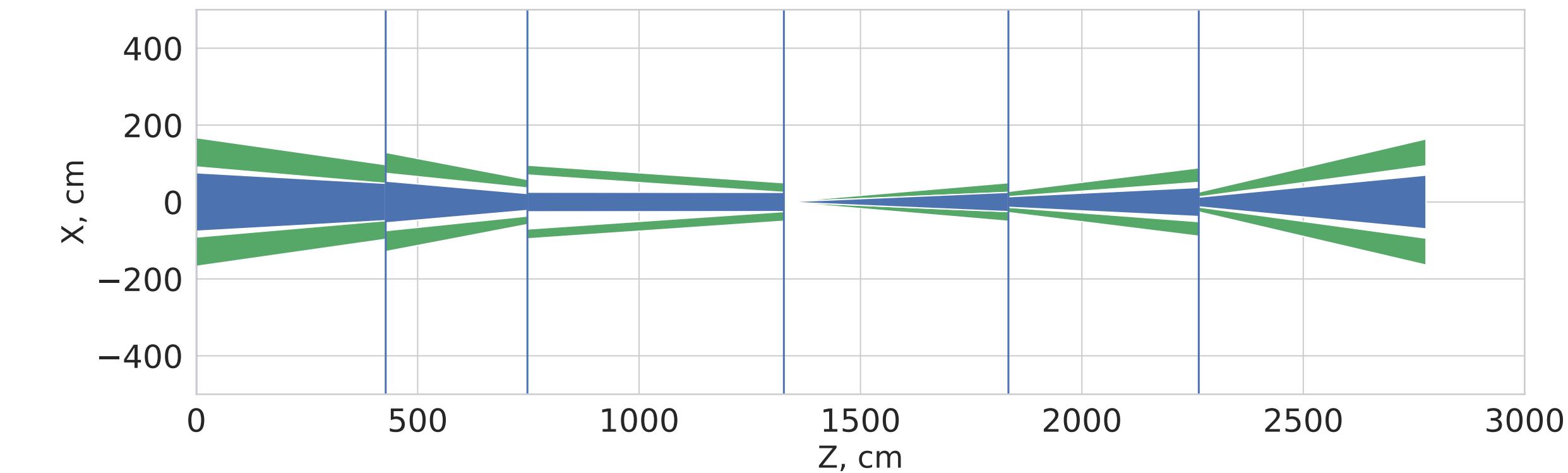
Method	Loss	Shield length (m)	Magnet weight (kt)
L-GSO	~ 2200	33.39	1.05
Bayesian opt.	~ 3000	35.44	1.27

SHiP: shield geometry change

Initial shape



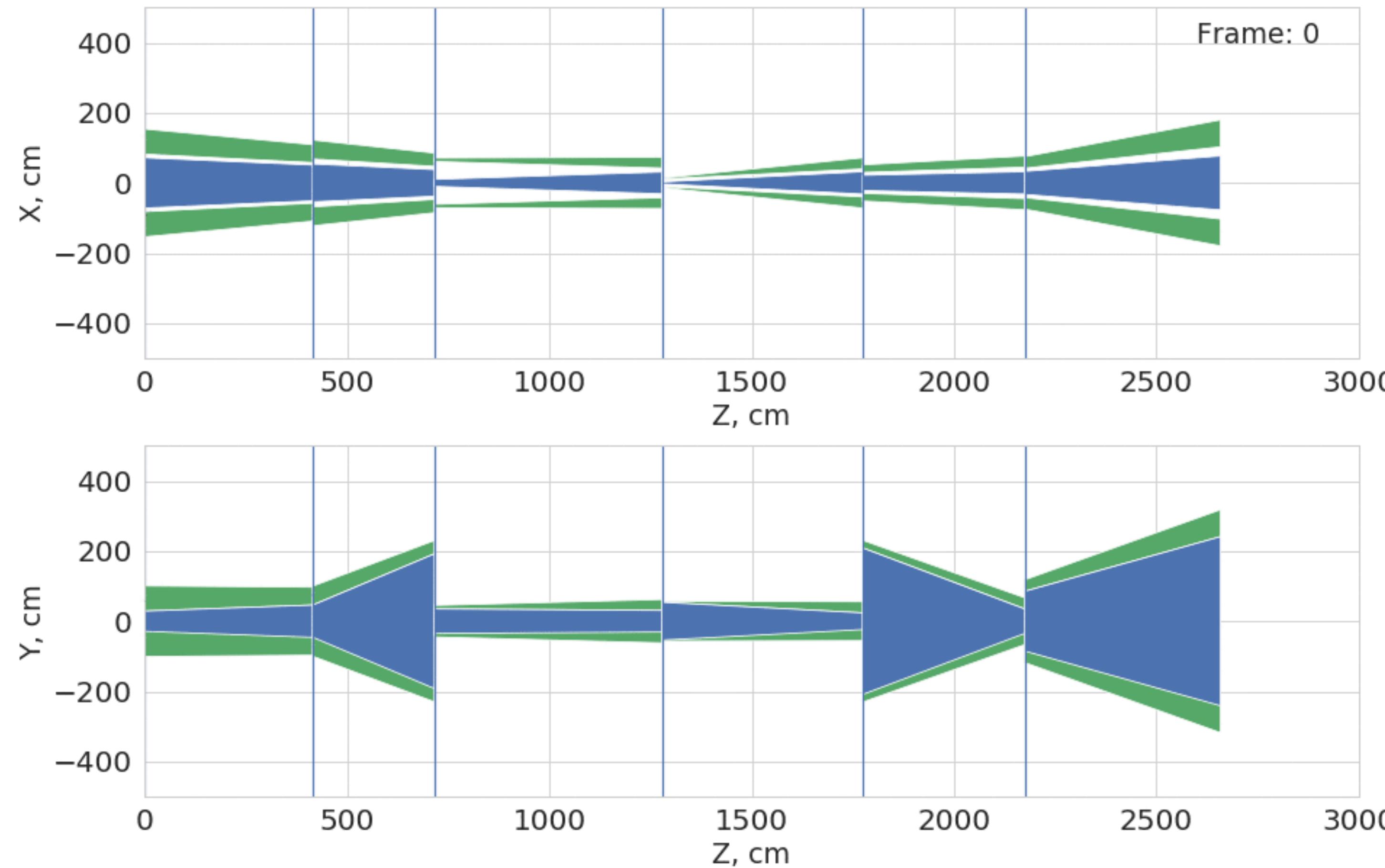
Final shape



Animation of the optimisation: <https://doi.org/10.6084/m9.figshare.11778684.v1>

SHiP: shield geometry change

Animation



Conclusion

- Overview of black-box optimisation methods / problem formulation
- Present novel optimisation approach: L-GSO
- Excel:
 - Parameters lie on a low-dimensional manifold
 - Simulator call is costly
- Empirically low variance
- Attained better minima than Bayesian optimisation in HEP problem

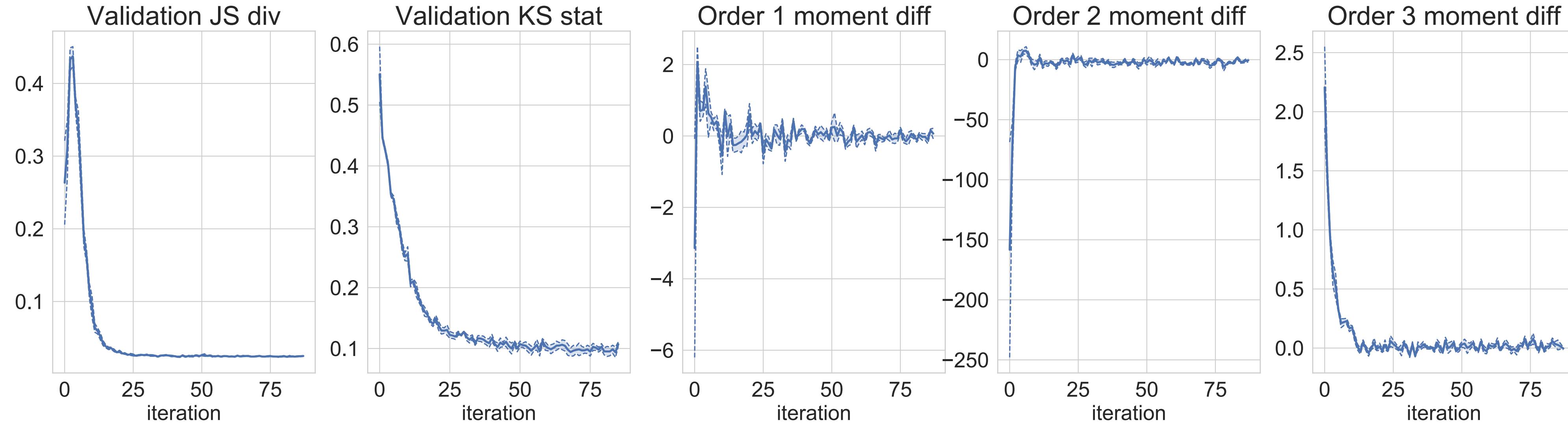
Future work:

- Implementation of trust-region methods
- Combination of BO and surrogate gradients

Backup



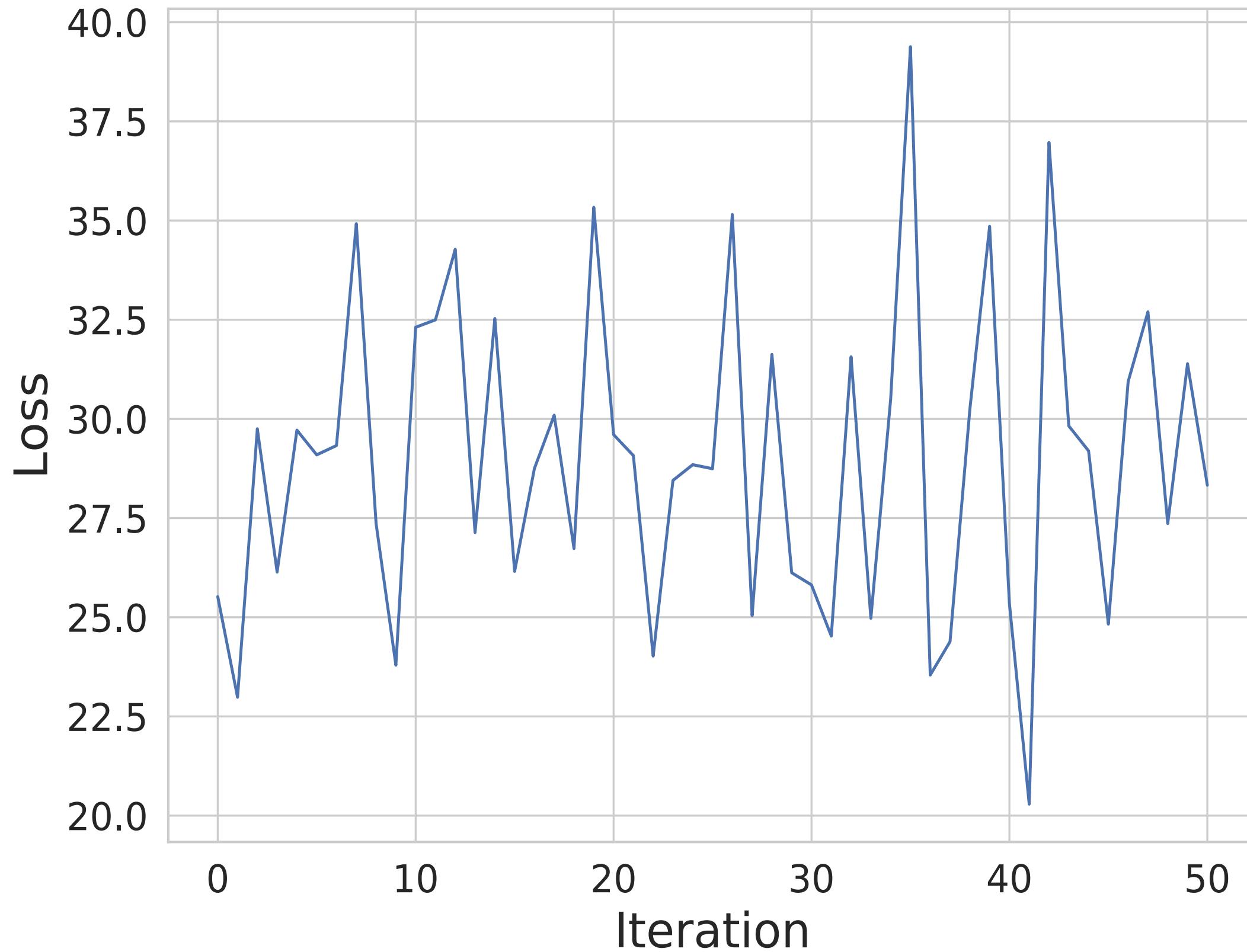
Monitoring of model performance



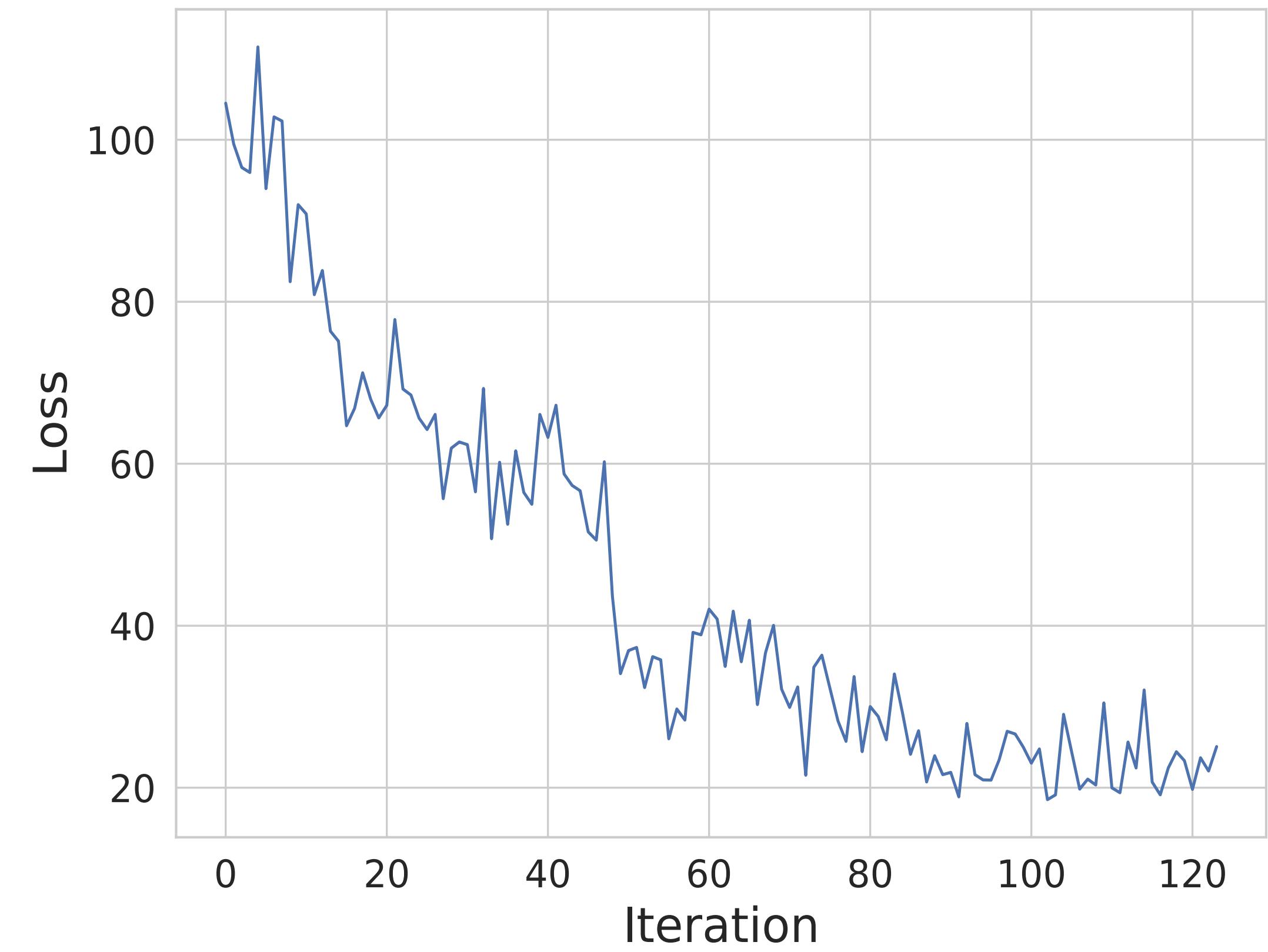
- Monitor various metrics between train distribution and sampled distribution
- Abort optimisation in case of divergence
- Adjust hyper parameters

SHiP: Shield optimisation

Initial point: previous optimum



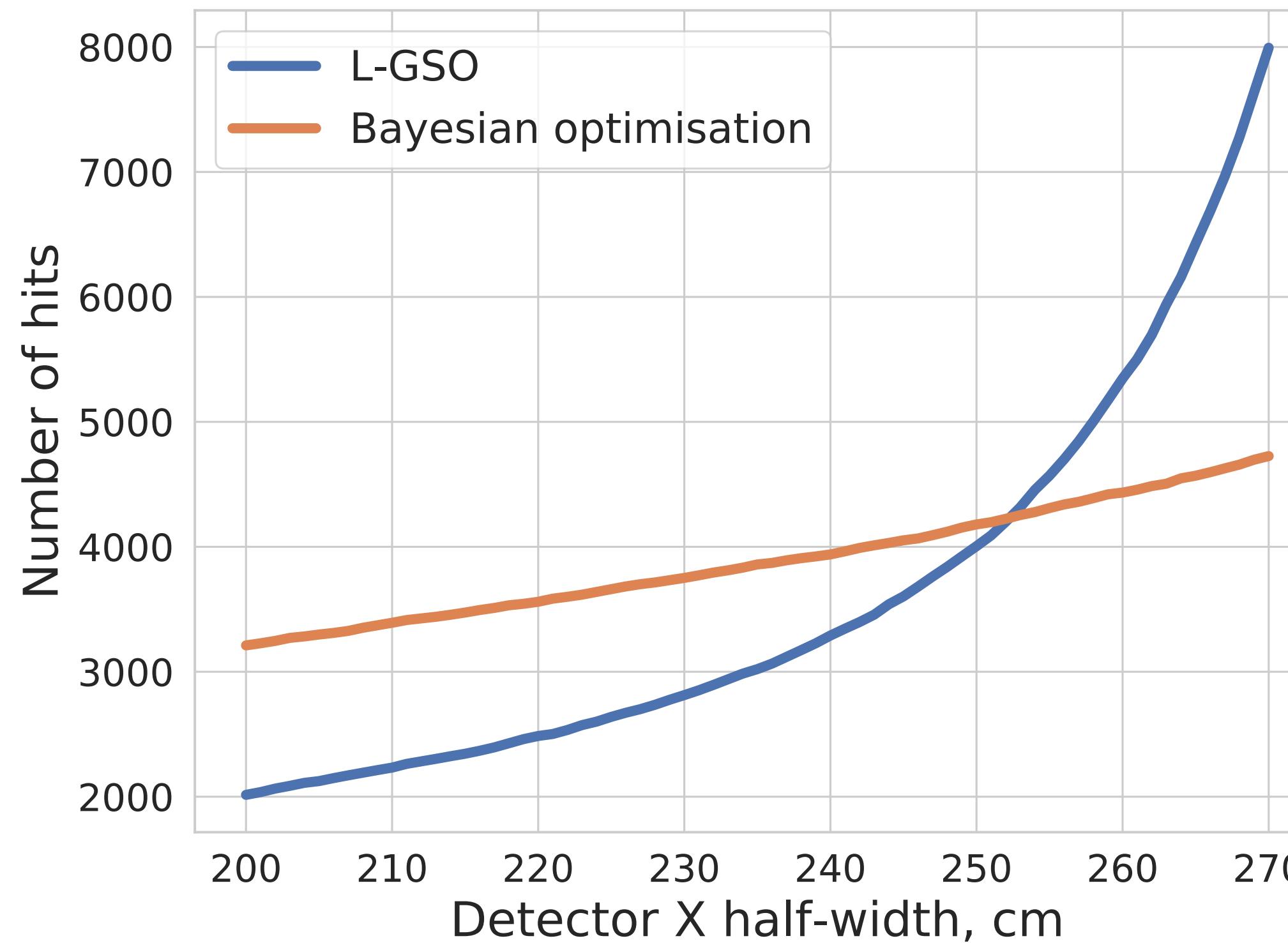
Initial point: new manually selected point



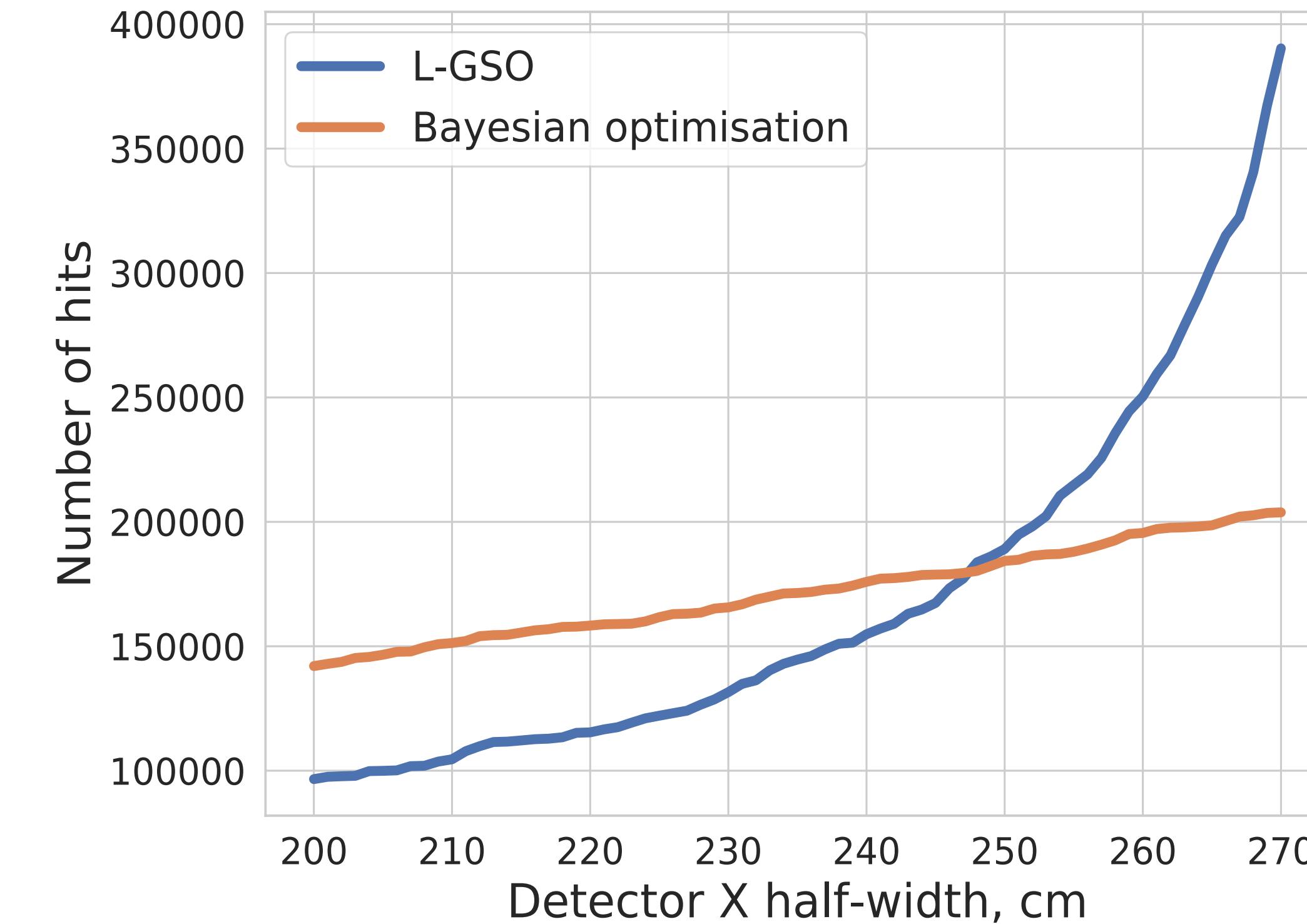
Is it the same optimum?

Shield optimization: comparison

Number of hits(unweighted)

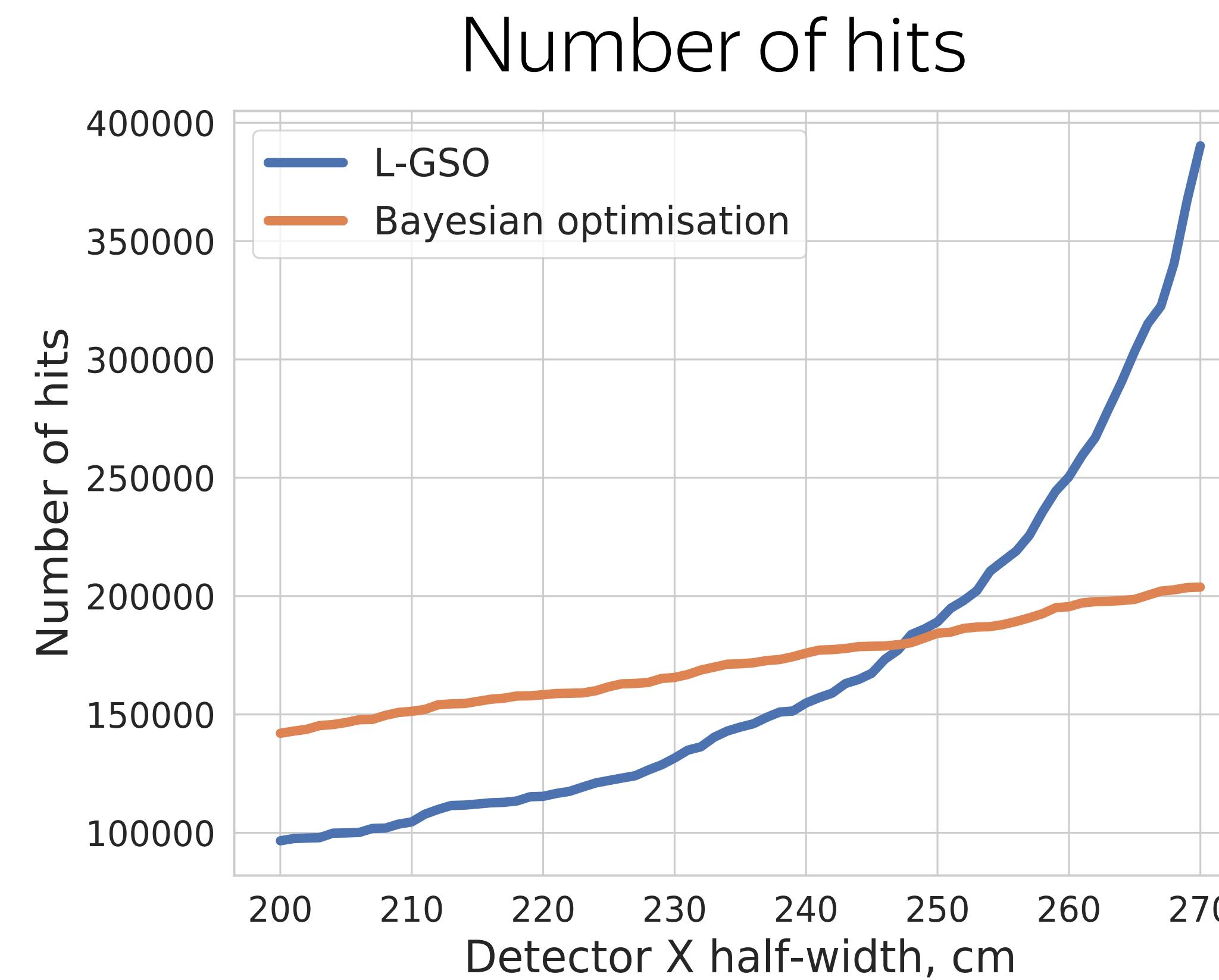


Number of hits(weighted)

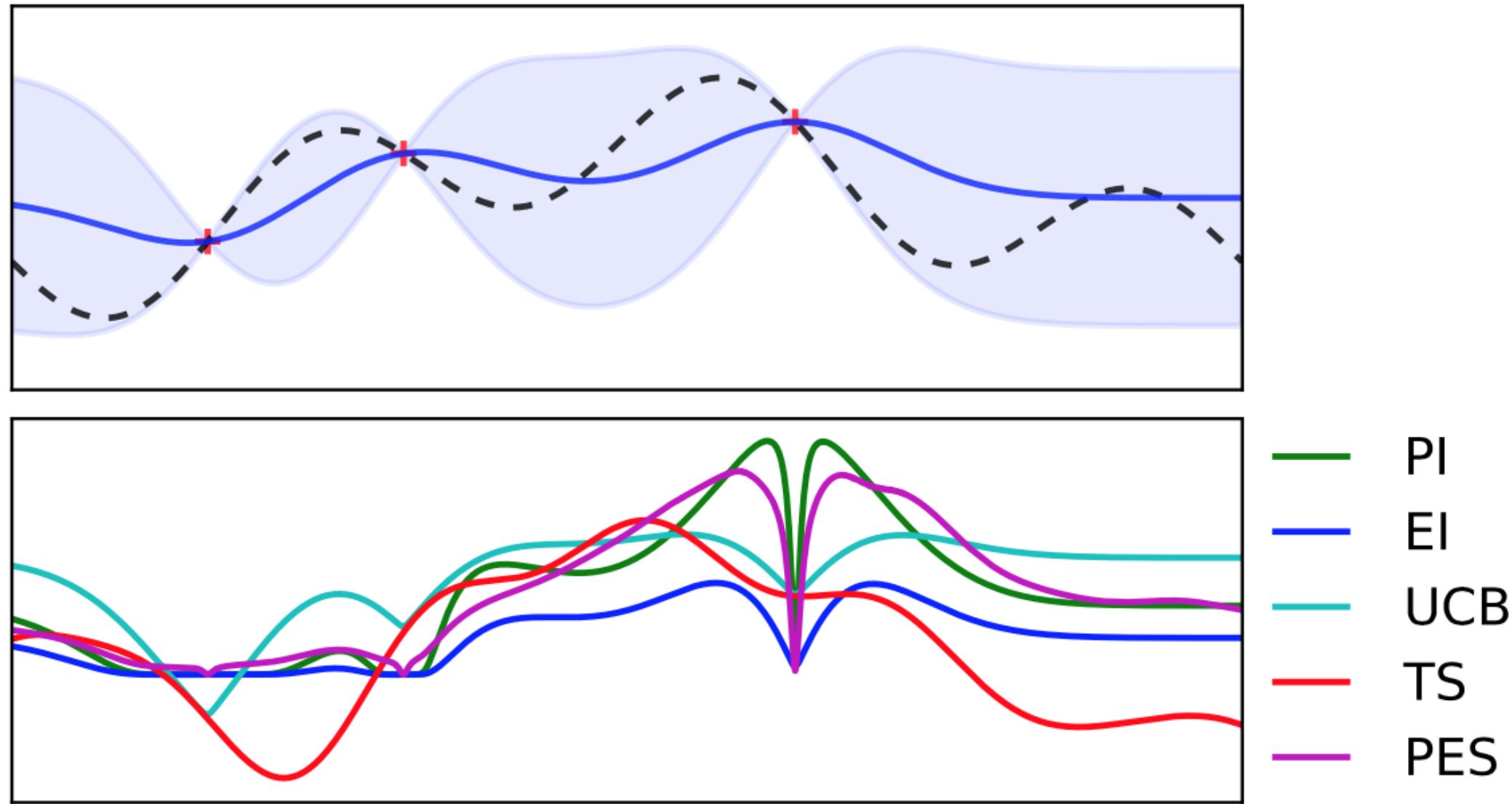


SHiP: shield optimisation comparison

■ L-GSO is very sensitive
to the optimised
parameter changes



Bayesian optimisation with Gaussian Process



- Optimise $\operatorname{argmin}_\psi f(\psi)$
- Approximates $f(\psi)$ with probabilistic model
- Provide mean μ and variance σ
- Maximise surrogate acquisition function $\alpha(x)$.
Example: UCB $\alpha(x) = -\mu_*(x) + \eta \sigma_*(x)$

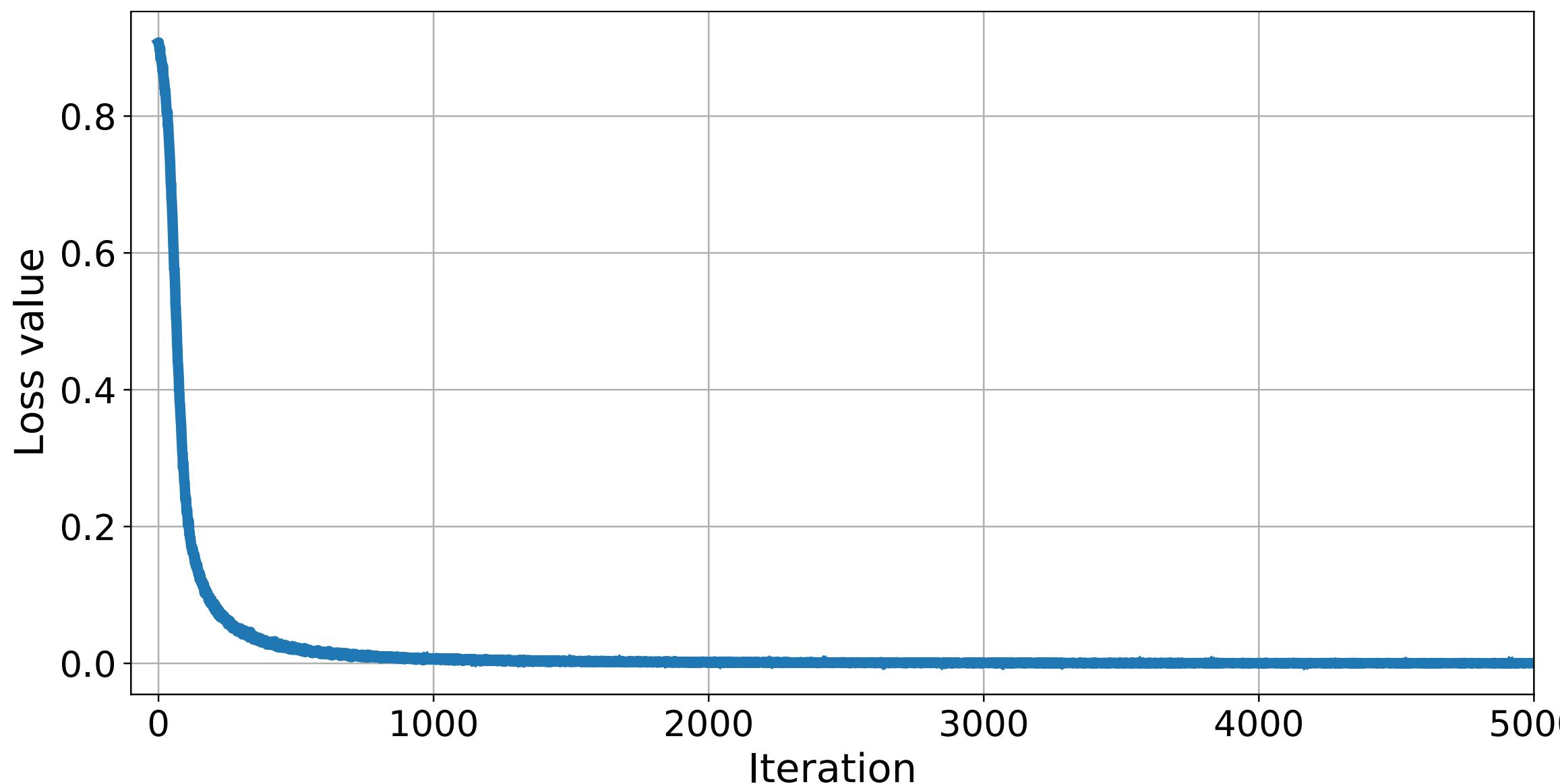
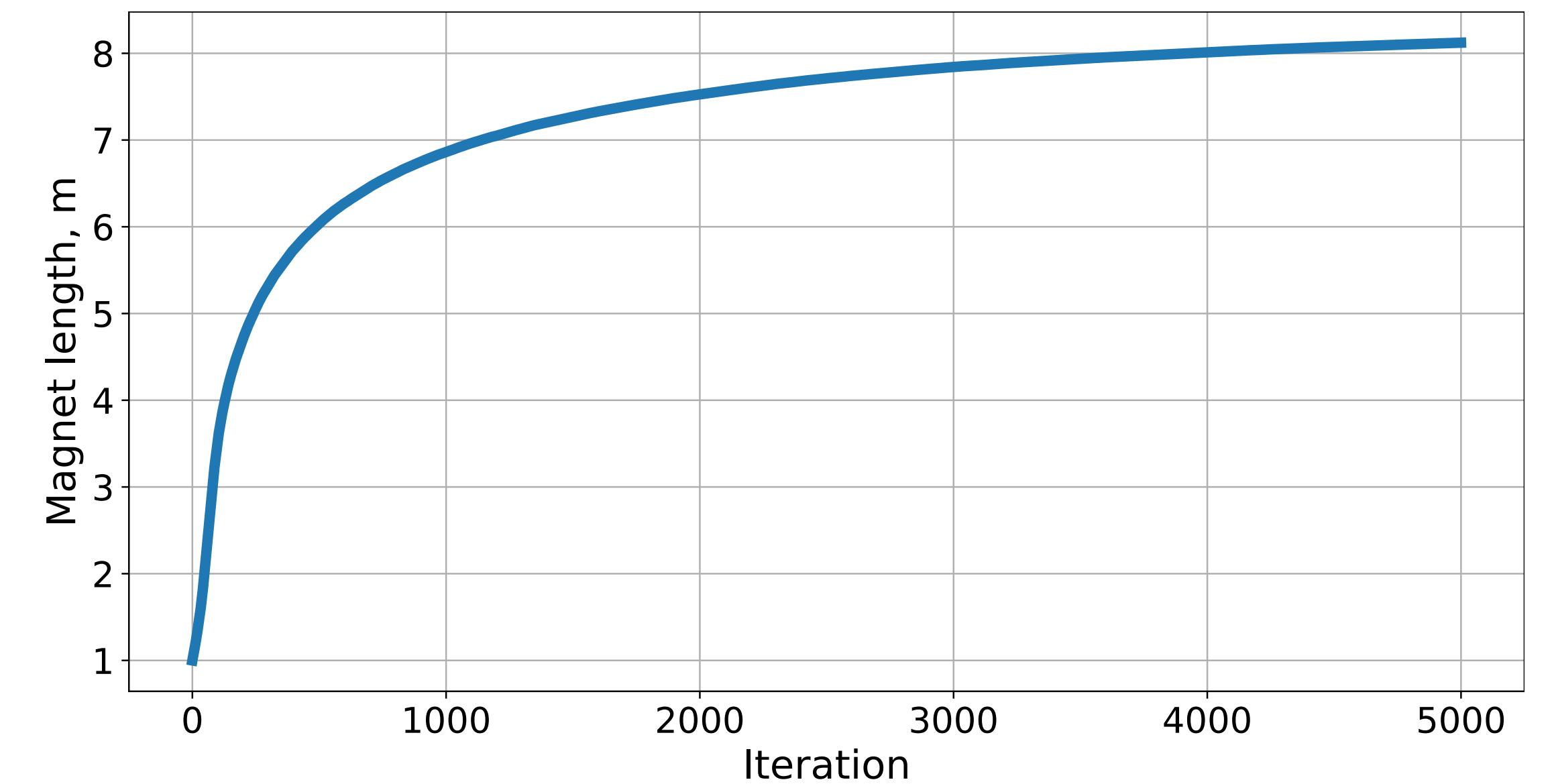
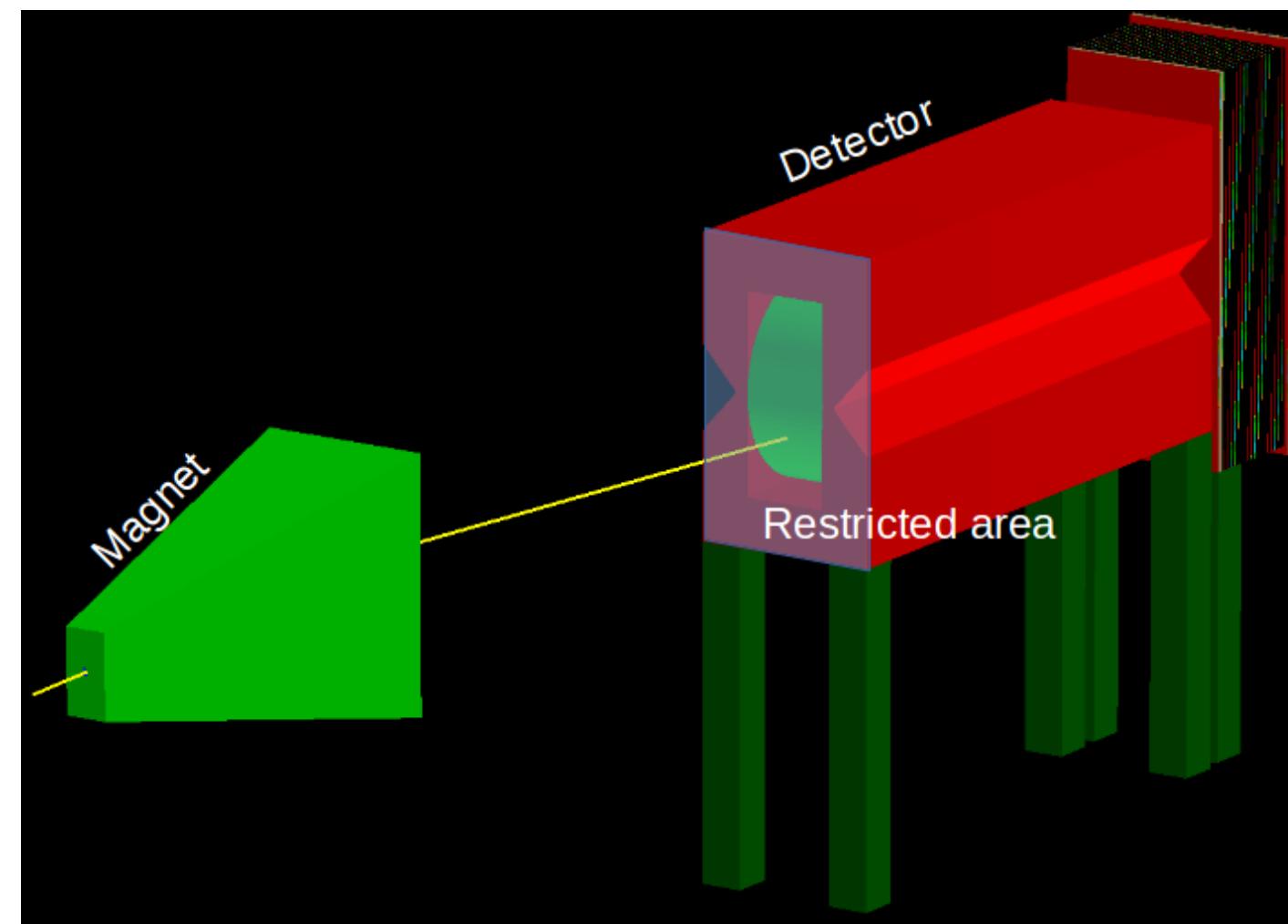
$$y_i = f_i + \epsilon, \quad \epsilon \sim N(0, \sigma_{obs}^2)$$

$$\mu(x_* | D) = K_{*D} (K_{DD} + \sigma_{obs}^2 I)^{-1} y$$

$$\sigma^2(x_* | D) = K_{**} - K_{*D} (K_{DD} + \sigma_{obs}^2 I)^{-1} K_{*D}^T$$

Generative surrogates: physics toy example

$$\psi \in \mathbb{R}^1$$



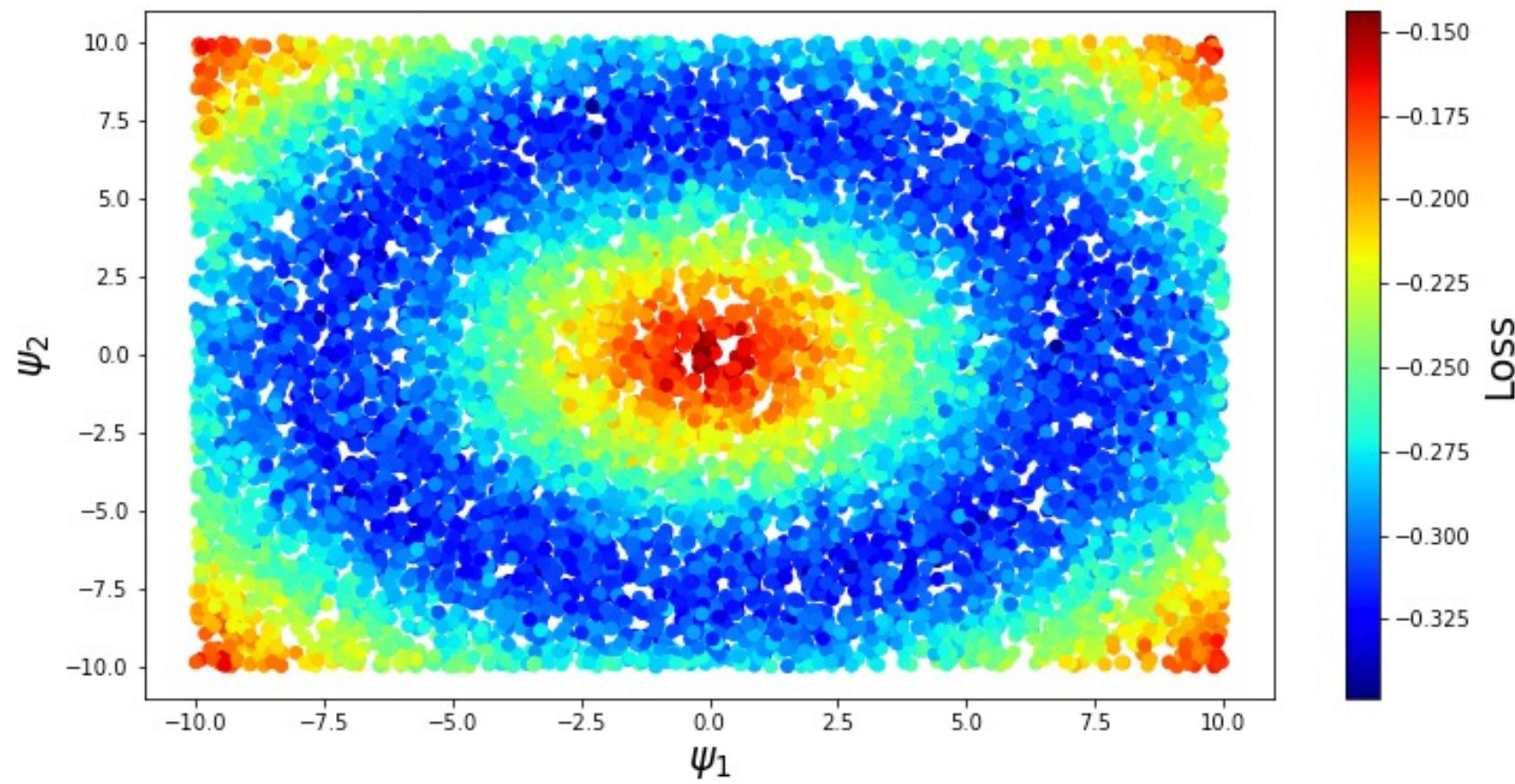
Generative surrogates: toy example

$$x \sim U(-10, 10), x_{latent} \sim N(x, 1), \psi \in \mathbb{R}^2$$

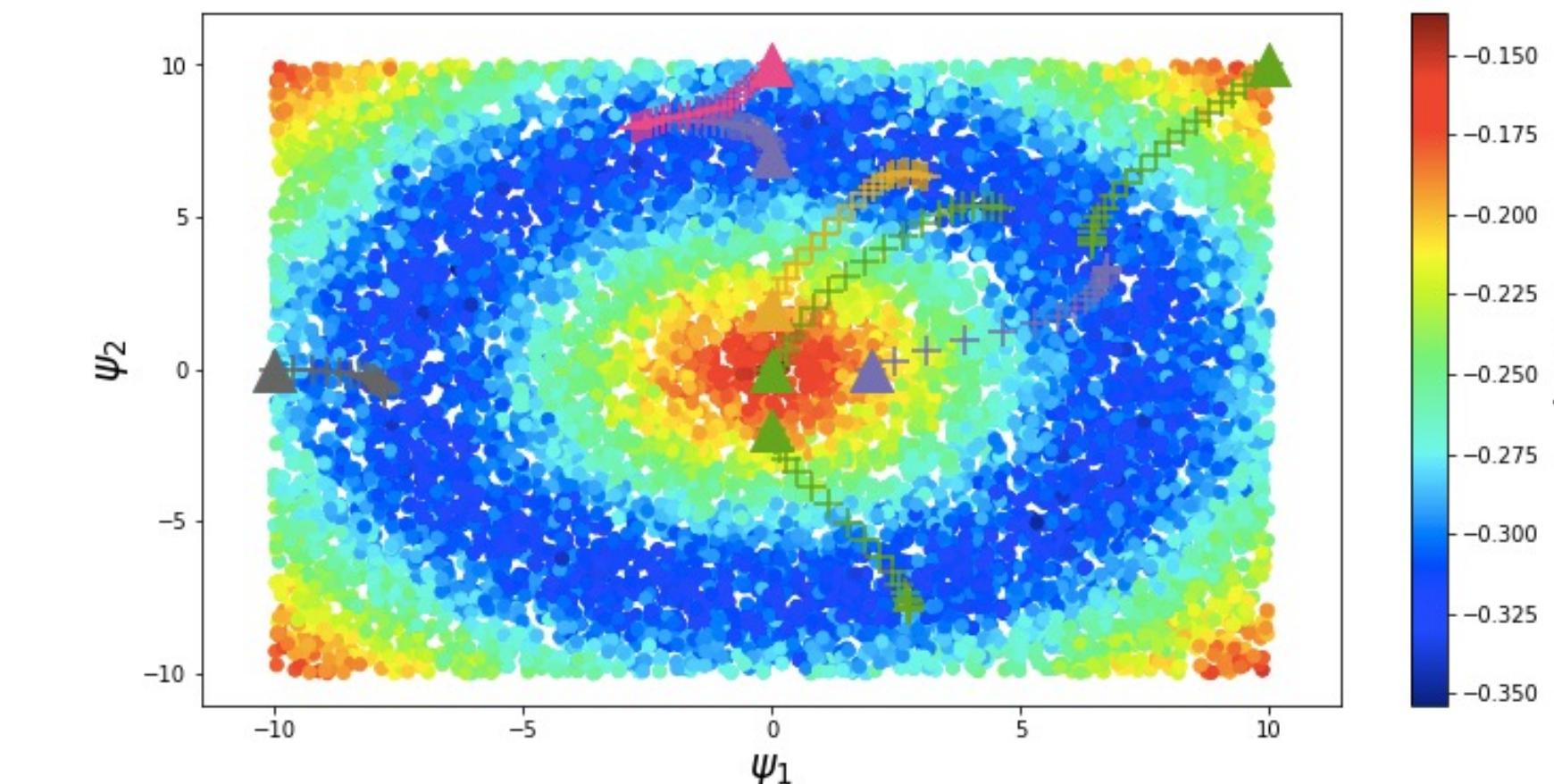
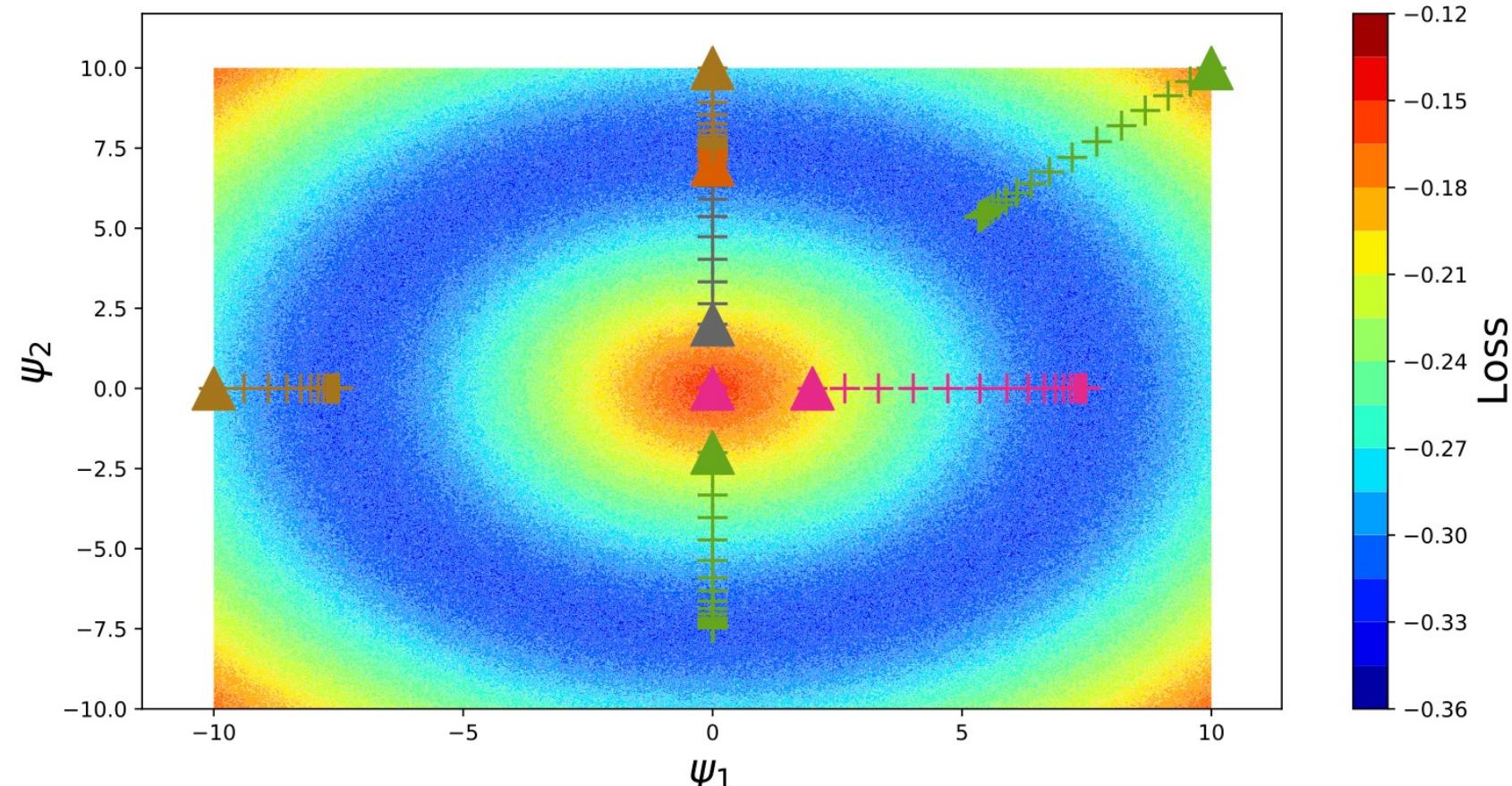
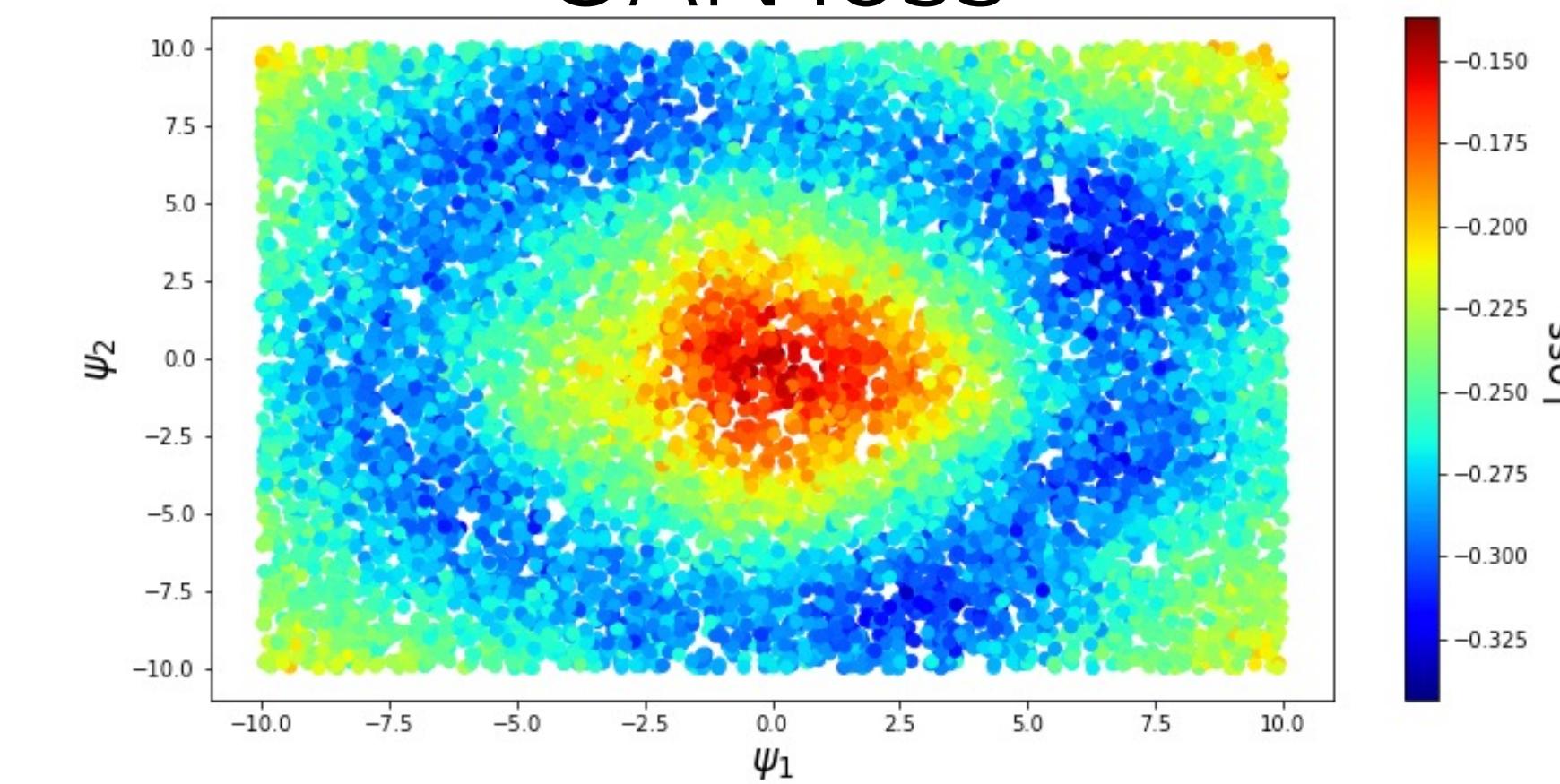
$$y \sim N(\|\psi\|_{L_2} + x_{latent}, 0.1 + 0.5 |x_{latent}|)$$

$$R(y) = \sigma(y - 10) - \sigma(y - 5)$$

Simulator loss



GAN loss

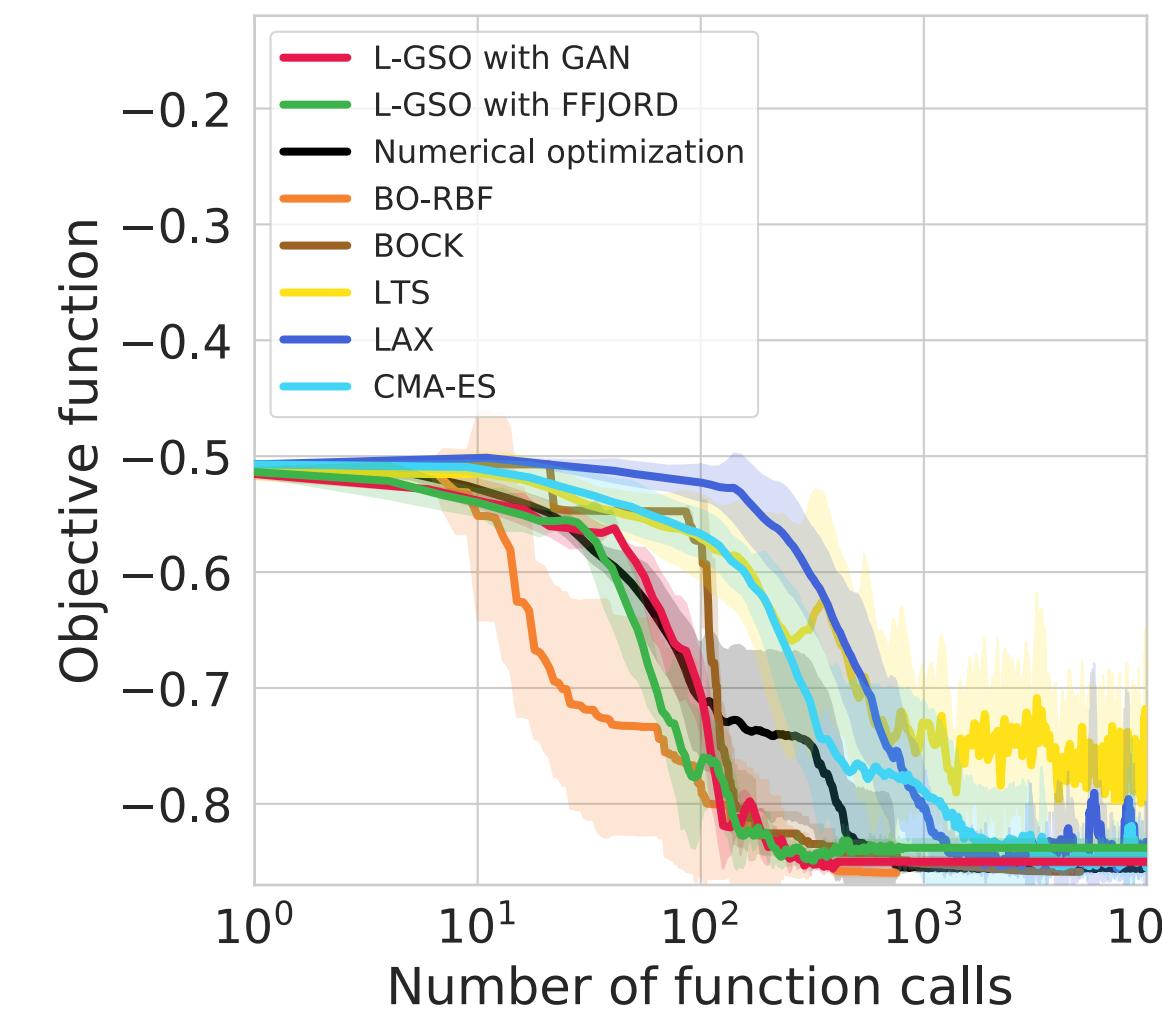


Toy Experiments

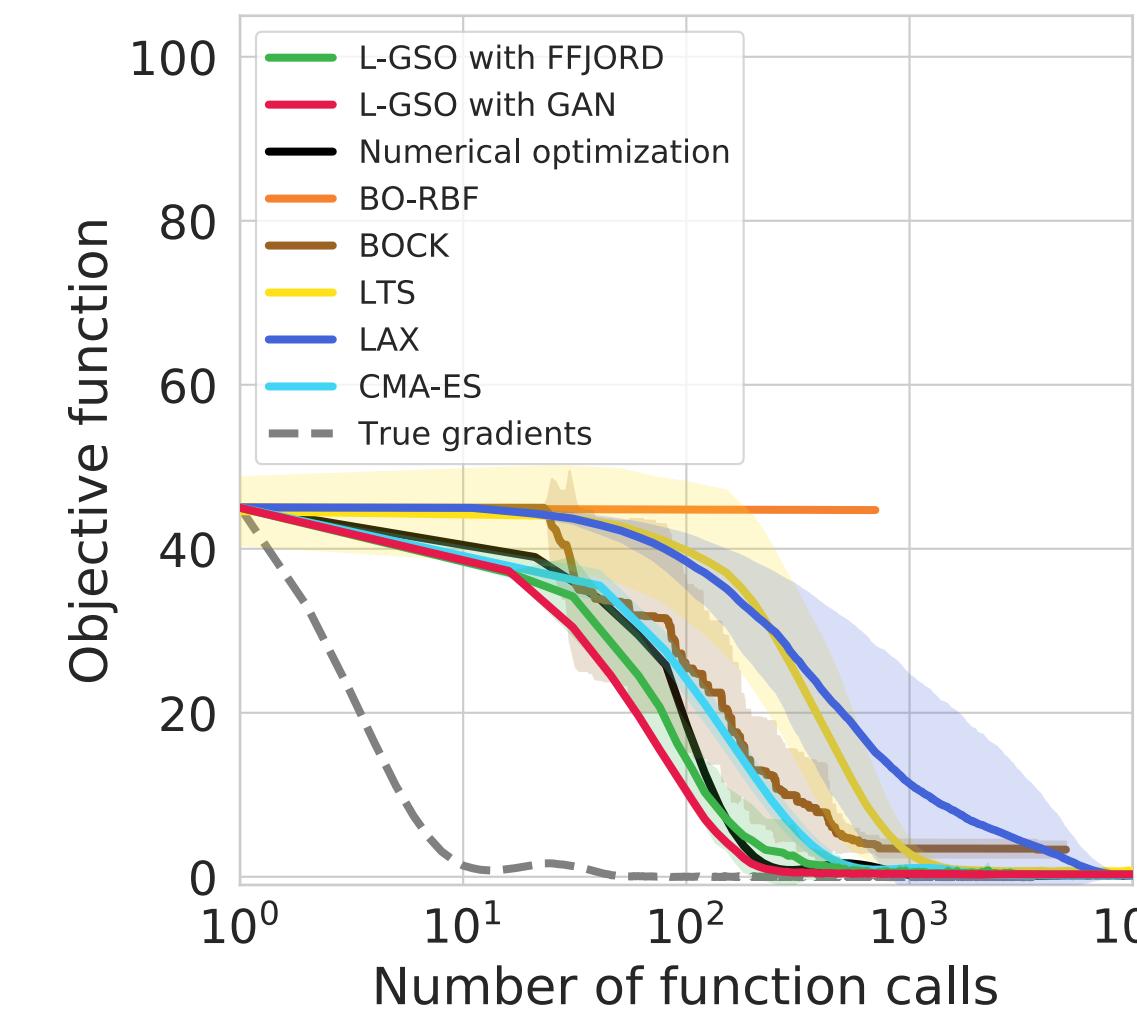
- We run experiments on a set of toy problems, simple ones and with effective dimensions.
- We want to compare L-GSO with other algorithms in small and large dimensional problems.
- We want to understand the effect of projecting parameters on the submanifold, as it is often the case in real life.
- We compare results in term of number of simulator calls and attained minima.
- We assume that a call to a simulator dominates the optimisation time

Toy Experiments

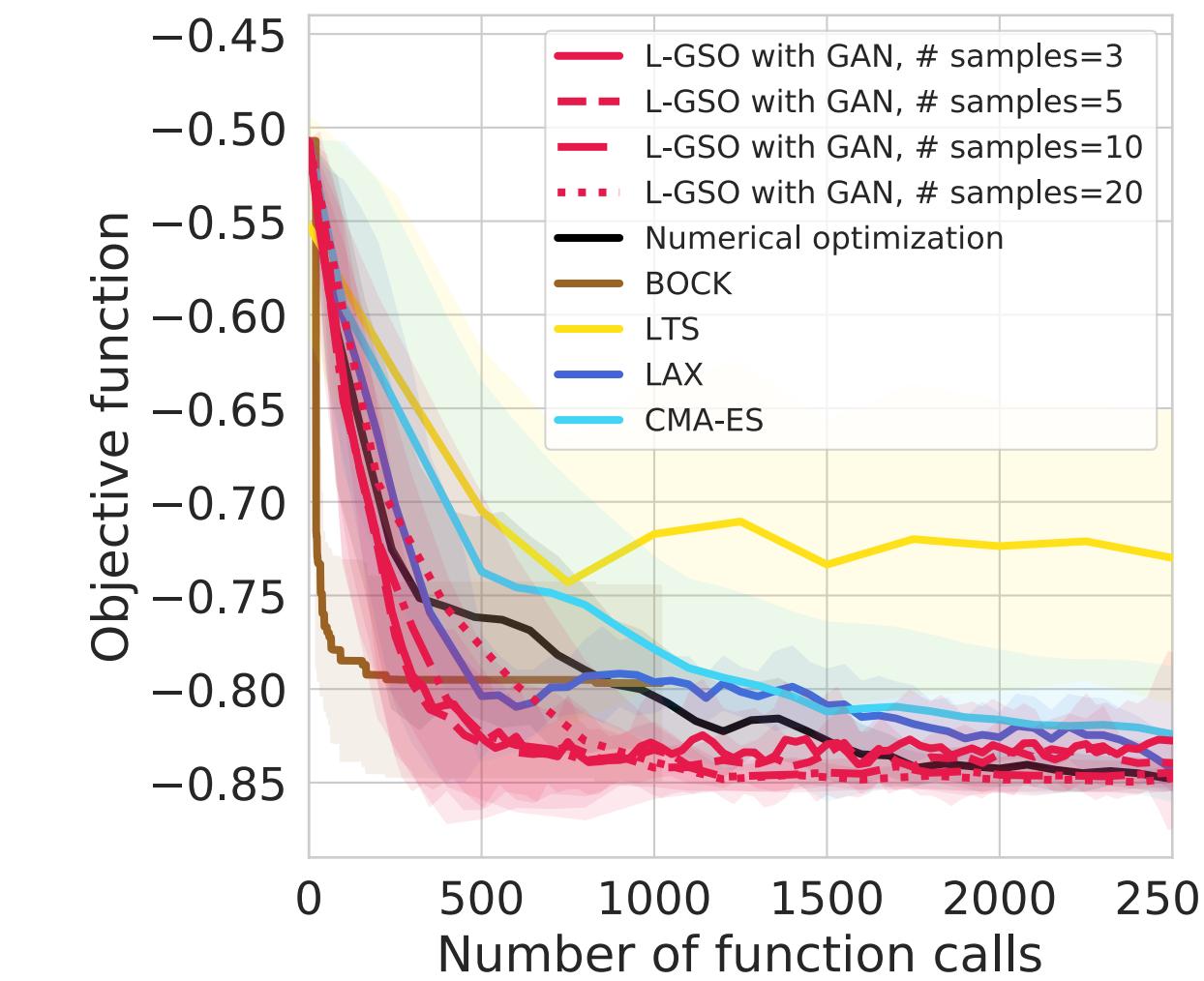
Three-Hump problem
2-dim



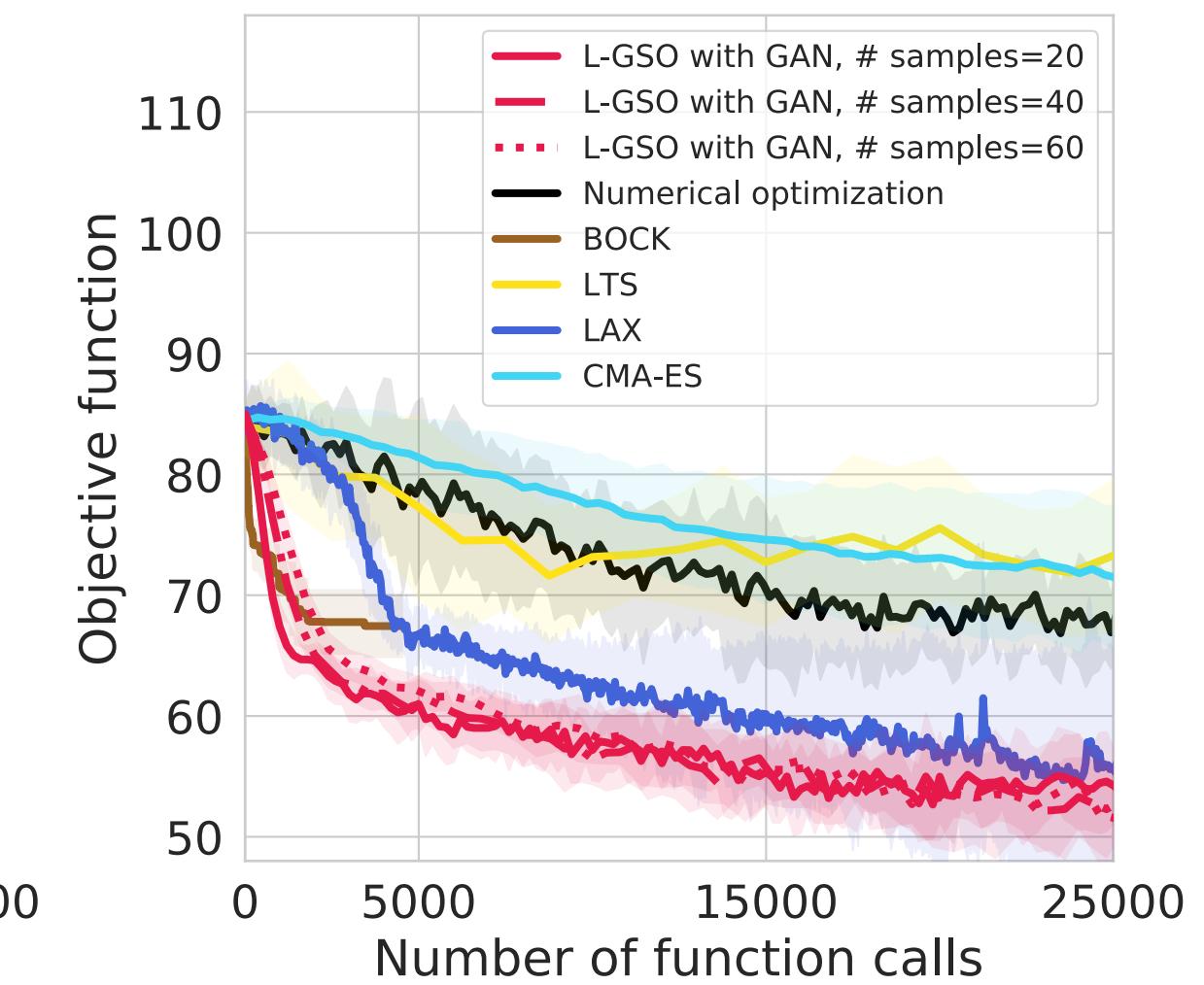
Rosenbrock problem
10-dim



Nonlinear Three-Hump
problem,
40-dim



Neural Networks weights
optimisation
91-dim



- L-GSO comparable to **all** baselines in low-dim problems in the speed of convergence
- L-GSO **outperforms all** baselines in a high-dim setting when parameters lie on a lower dimensional manifold.
- L-GSO has lower variance in resulting objective function value than other methods