Bayesian Optimization Tutorial

Module 4: Beyond Black-box Bayesian Optimization

Joel Paulson
Assistant Professor, Department of Chemical and Biomolecular Engineering, The Ohio State University

Great Lakes PSE Student Workshop, 2023

For copies of slides & code, see
https://github.com/joelpaulson/Great_Lakes_PSE_Workshop_2023

The Principles of Bayesian Optimization are Extremely Flexible

while {budget not exhausted}

Fit a Bayesian machine learning model
(usually Gaussian process regression)
to observations {x, f(x)}

Find x that maximizes acquisition(x, posterior)

Sample x & then observe f(x)

end

The Principles of Bayesian Optimization are Extremely Flexible

Elicit some **prior distribution** on the functions of interest

while {budget not exhausted}

Find **information source** whose **value of information** is the largest in the set of options

Query corresponding **information source**

Update the **posterior distribution** of the functions

end

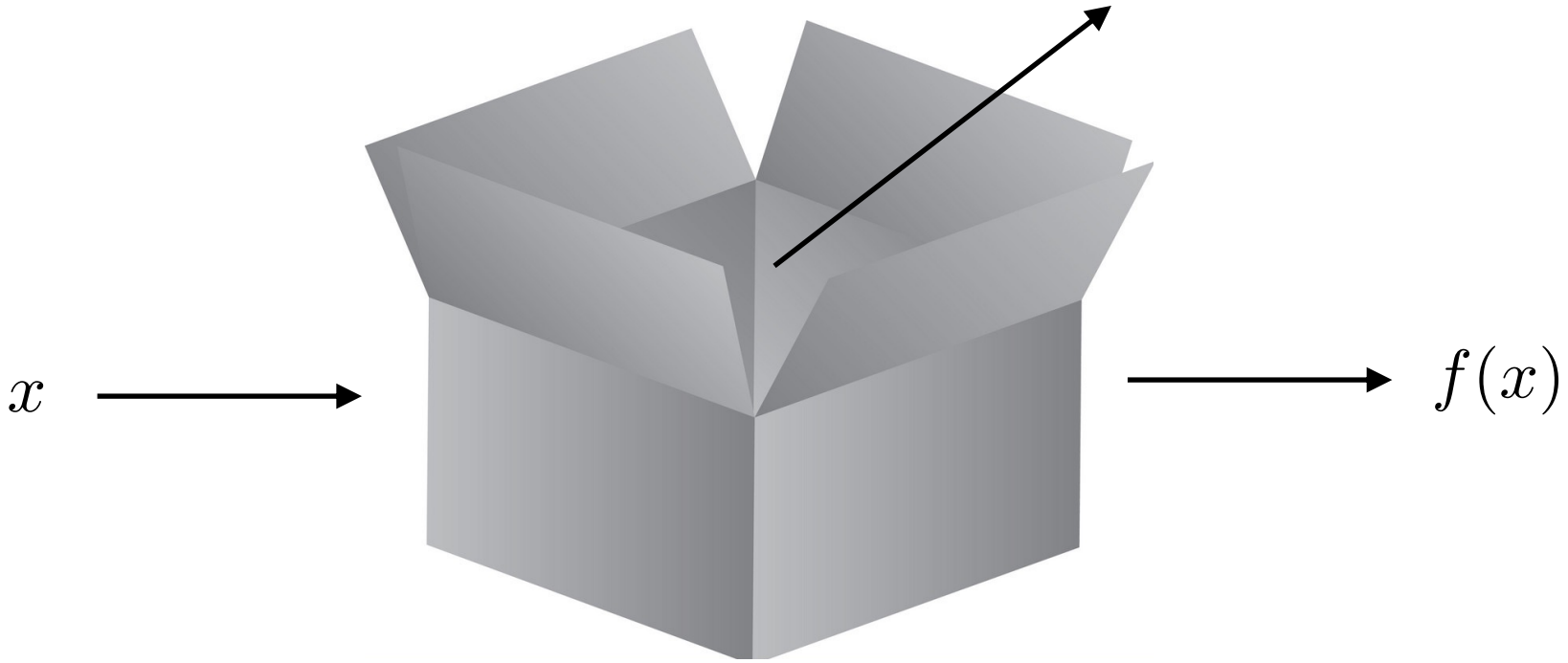
Grey-Box Perspective

"one should avoid learning what you already know"

We can do better by peeking <u>INSIDE</u> of the box

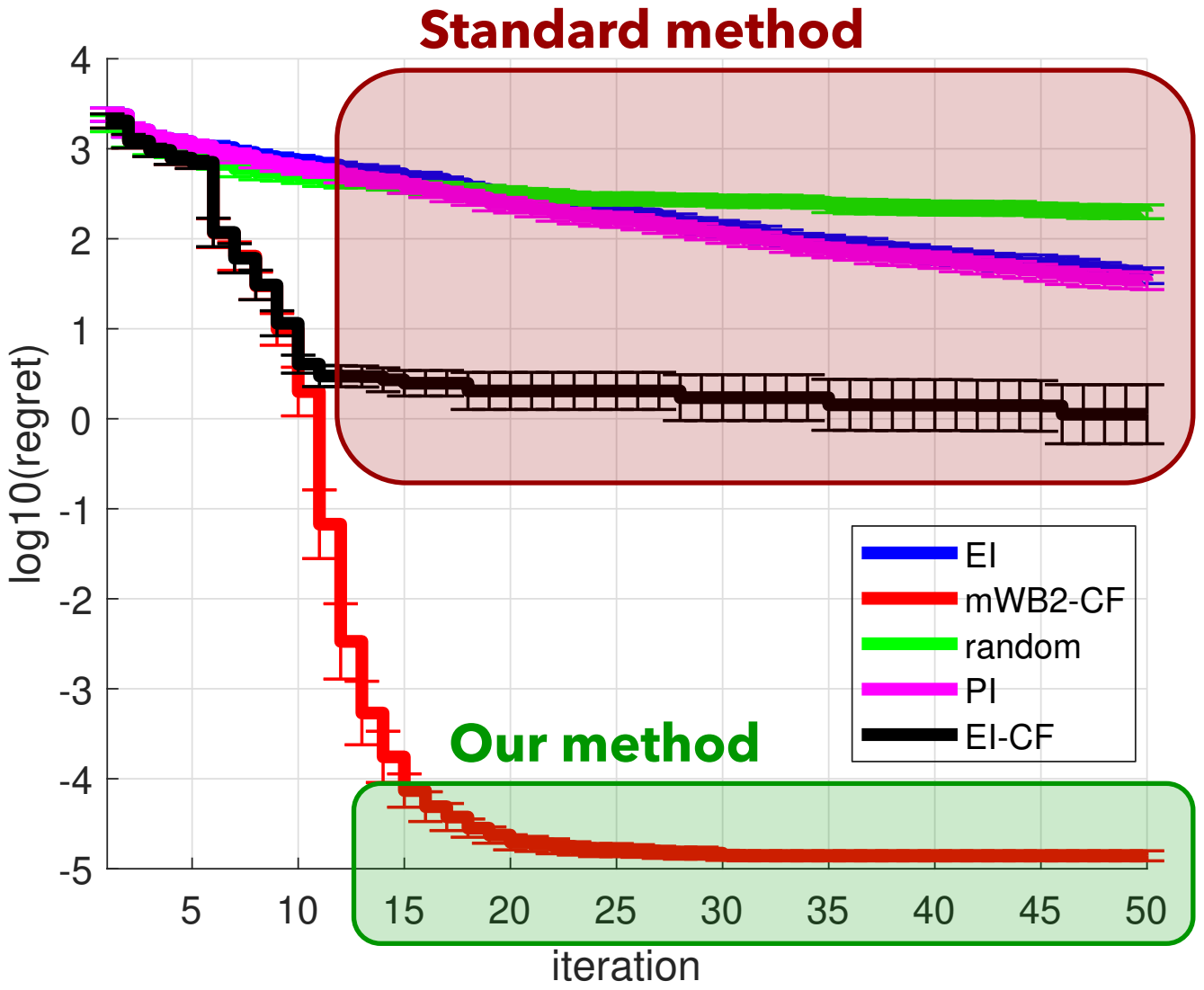
(really just any additional information about problem and/or simulator)

"knowledge" or "physics"



*sometimes referred to as grey-box (or hybrid) optimization

We can do MUCH better by peeking inside of the box



treats problem as fully black-box

partially looks inside box
→ "grey-box"

Theoretical Justification: Why is Grey-Box View Important?

- Simple regret: $\mathrm{Regret}_n = f(\tilde{x}_n) - f(x^\star)$
- Recommended point after n evaluations: $\tilde{x}_n$

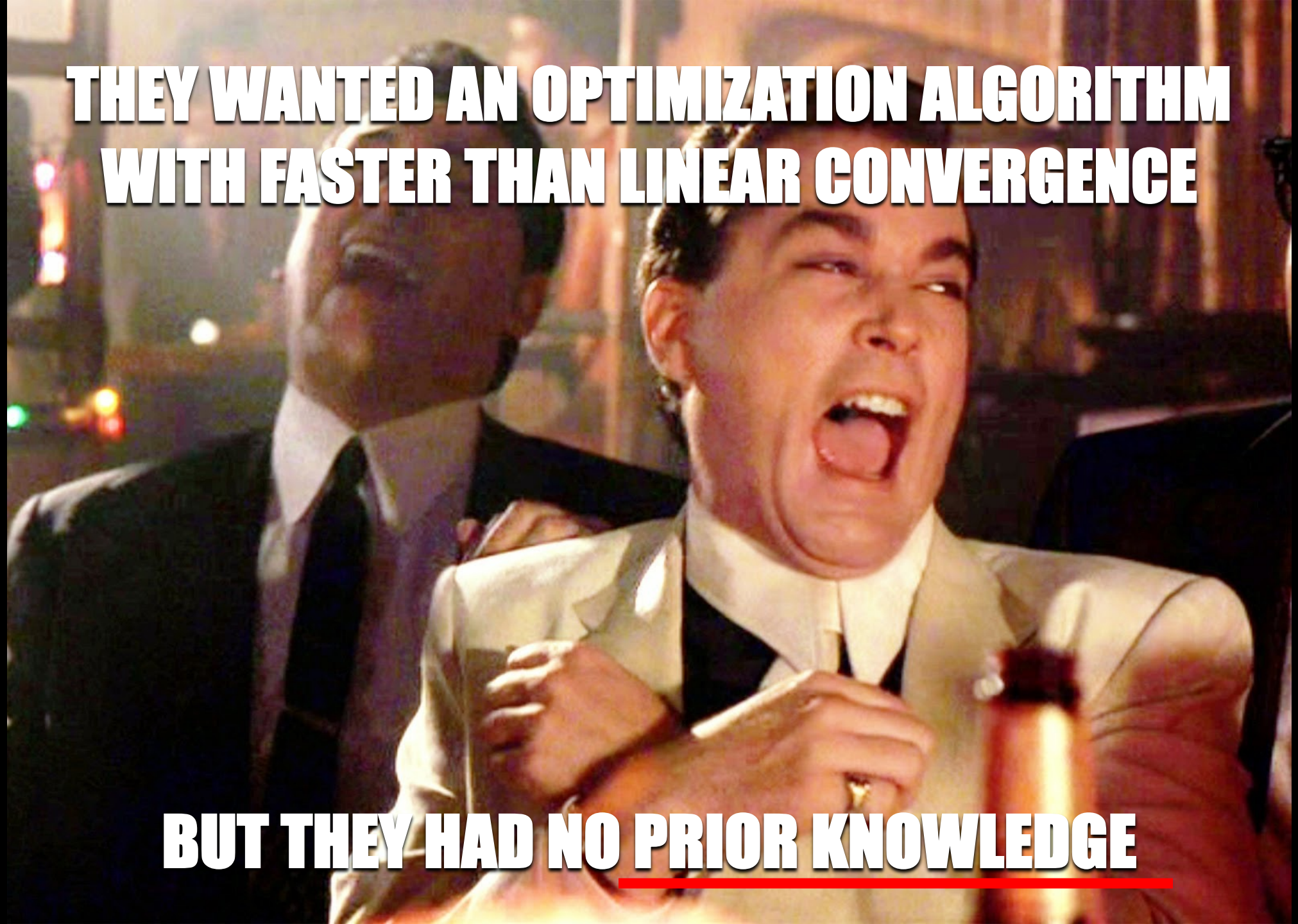
Random variables due to noise + initial samples

- Bound on performance of **ANY black-box** querying algorithm:

$$\exists C \geq 0,\ \exists n_0,\ \text{such that } \mathbb{E}\left\{\mathrm{Regret}_n\right\} \leq \frac{C}{n},\ \forall f \in \mathcal{F},\ \forall n \geq n_0$$

[Chen, 1988]

- It is not possible to do better than linear convergence (on average)!
  - Must exploit other information to overcome this limiting barrier


THEY WANTED AN OPTIMIZATION ALGORITHM
WITH FASTER THAN LINEAR CONVERGENCE
BUT THEY HAD NO PRIOR KNOWLEDGE

Prior / Domain Knowledge Can Come in Many Forms

- Composite Functions (+ Constraints)
  - Represent function with a graph-like composite structure $f(x) = g(h(x))$ where only part of the graph $h(x)$ is unknown
- Multi-Information Source Problems
  - Represent function $f(x) = g(x, 1)$ as limit of a higher-dimensional function $g(x, s)$ that involves correlated source parameters s
- Local Derivative & Noise Information
  - Incorporate any information relevant to noise and/or rate-of-change of $f(x)$

Prior / Domain Knowledge Can Come in Many Forms

- Robust and Stochastic Optimization
  - Represent function $f(x, w)$ as function of design x and uncertainty w (with additional operators over w such as sums, integrals, max)

- Low-Dimensional Embeddings
  - Whenever $x \in \mathcal{X}$ is a large, identify an embedding space $z = h(x)$ of significantly lower dimension that captures dominant behavior

- Unknown Objective Functions (Preference Learning)
  - Try to infer objective function $f(x)$ from human-labeled preferences (as opposed to assuming we know the right function from the start)

Prior / Domain Knowledge Can Come in Many Forms

…and many, many more.

Let me just take a single example (**composite functions**) to step through the details

The COBALT Method

Our Approach: The COBALT Method

Code available : https://github.com/joelpaulson/COBALT

while {budget not exhausted}

Fit **multi-output** Gaussian process regression to observations {x, **h(x)**}

Find x that maximizes a **new acquisition function**
COBALT(x) = E[{**g(h(x))** – f*}$^+$]

Sample x & then observe **h(x)**, f(x)

end

We have further modified acquisition to account for unknown constraints

*Recall: objective is f(x) = g(h(x))

[Paulson and Lu, *Computers & Chemical Engineering*, 2021]

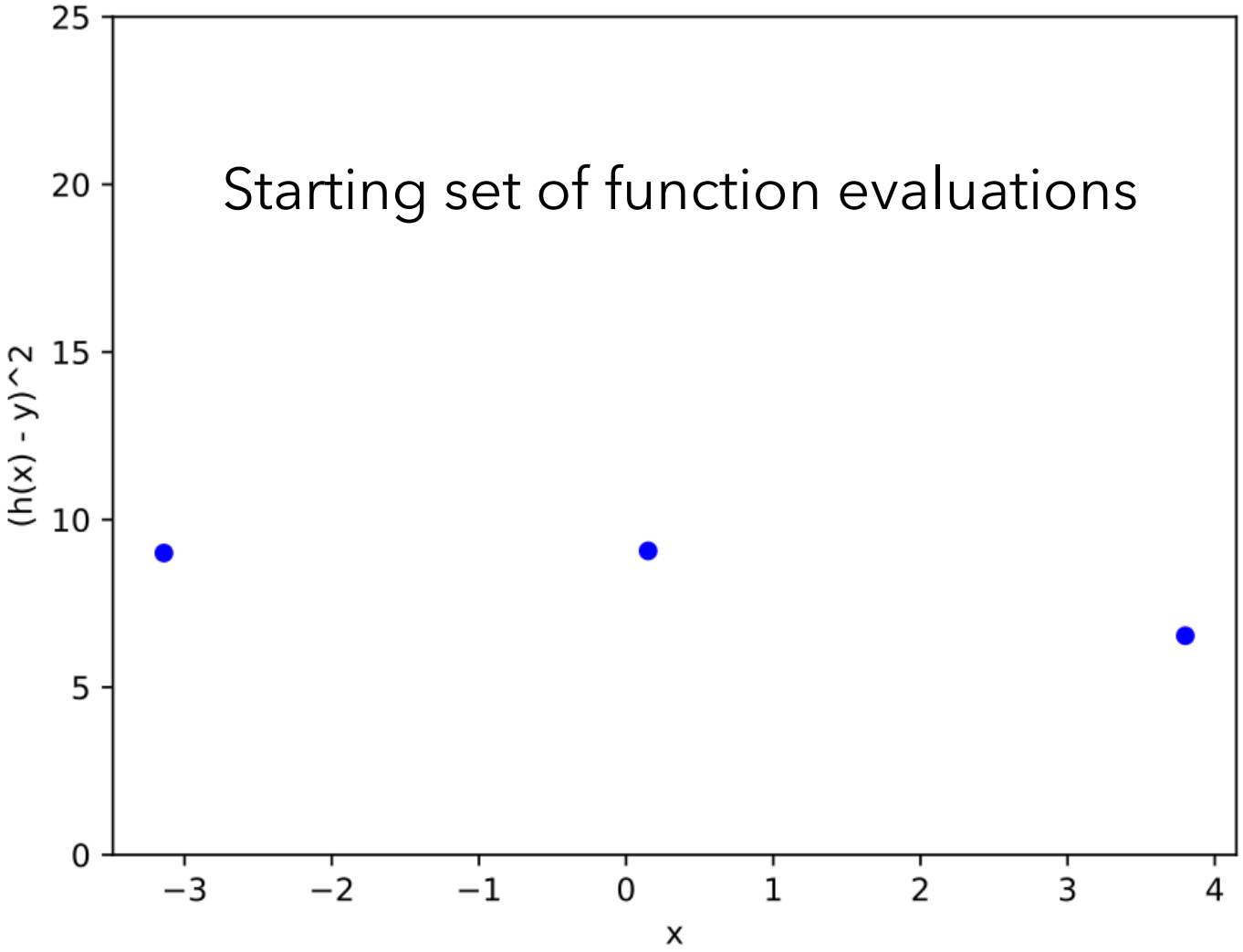
Let's see why COBALT works with a simple example

- Assume that we have the following variable declarations:
  - x is a parameter of a black-box simulator (e.g., Aspen)
  - $h(x)$ is the simulator's prediction given x
  - y is our observed data that we would like our simulator to match

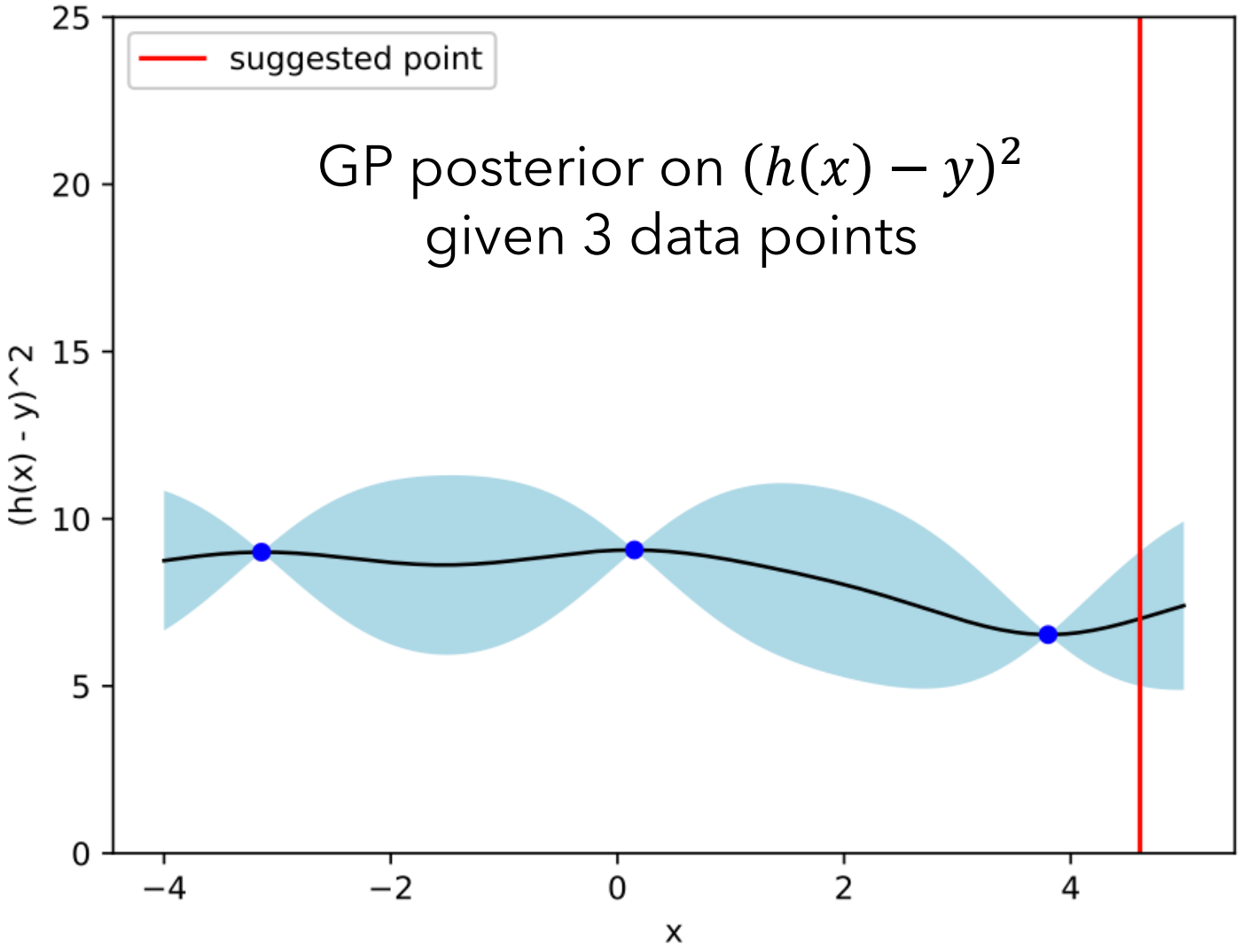
- To calibrate our simulation model, we want to solve

$$\min_{x} \ (h(x) - y)^2$$

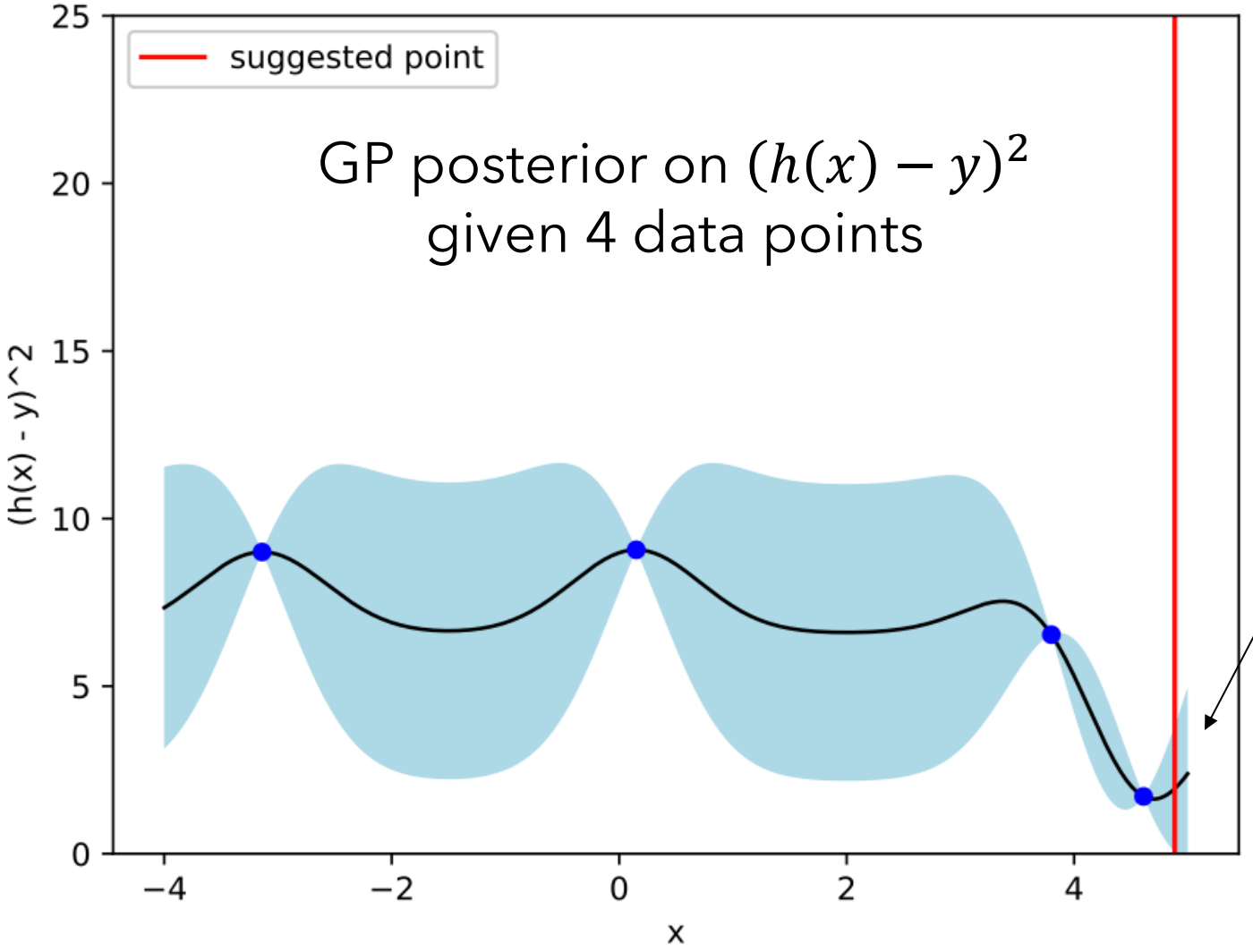
Let's Solve Example using Standard Bayesian Optimization



Let's Solve Example using Standard Bayesian Optimization

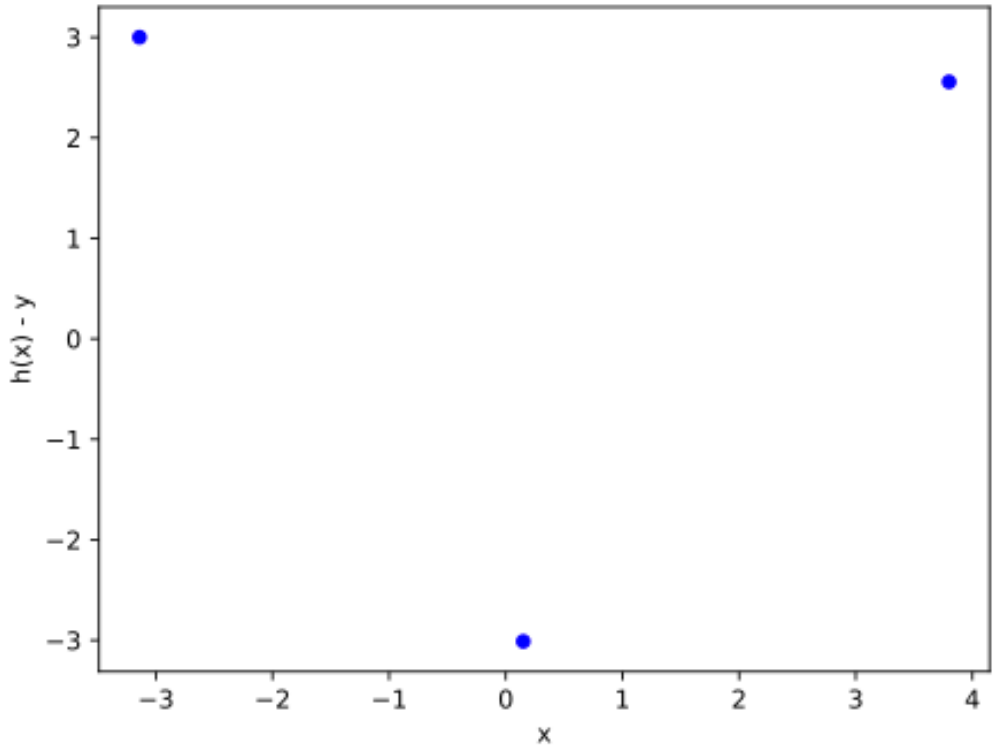


Let's Solve Example using Standard Bayesian Optimization



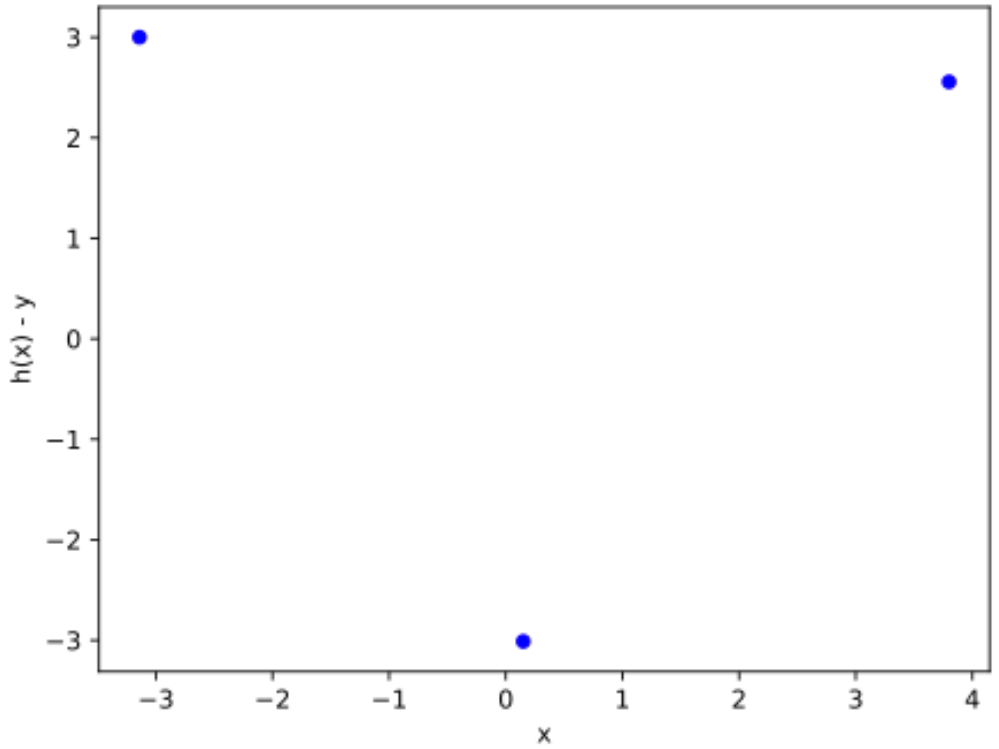
Notice how BO continues to sample in the right corner of the function

Now Let's Solve the Same Example using COBALT

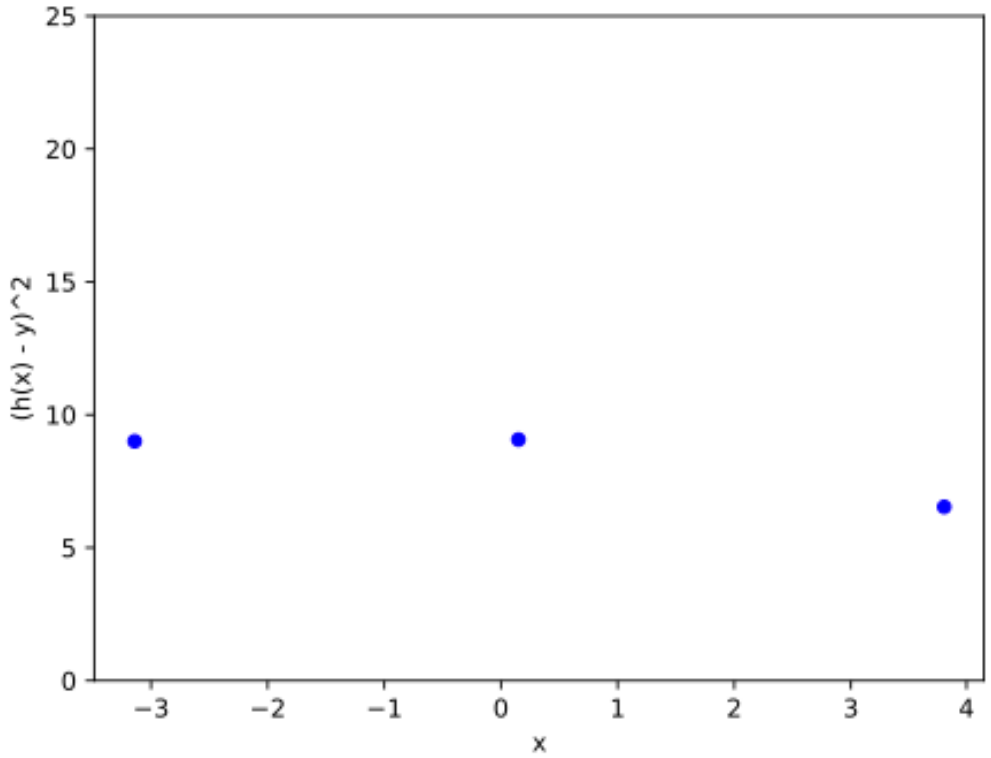


Evaluations of $h(x) - y$

Now Let's Solve the Same Example using COBALT

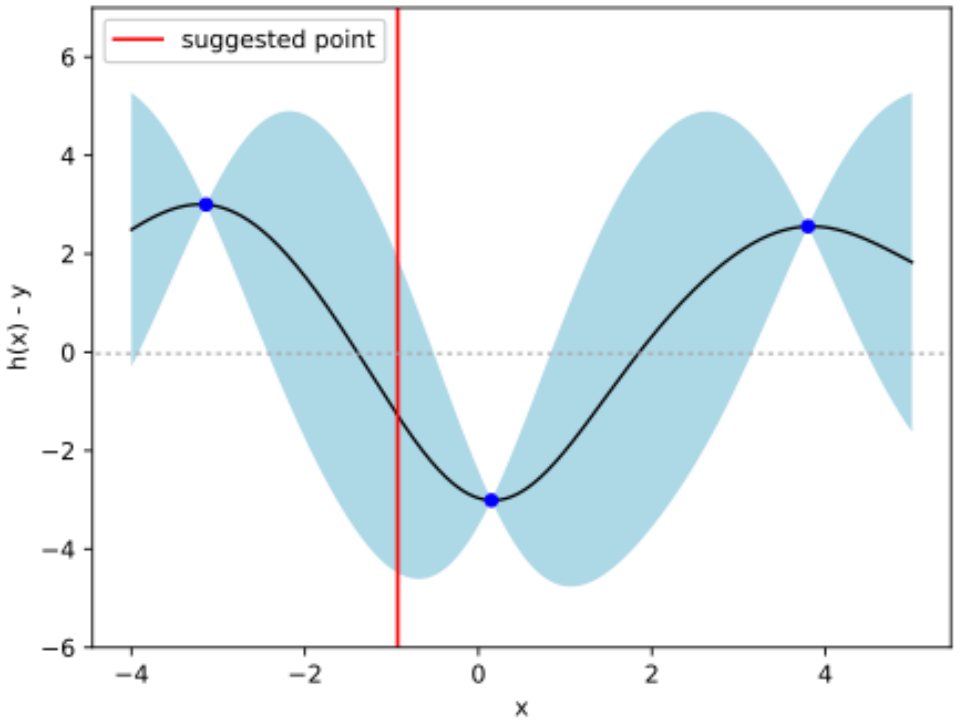


Evaluations of $h(x) - y$

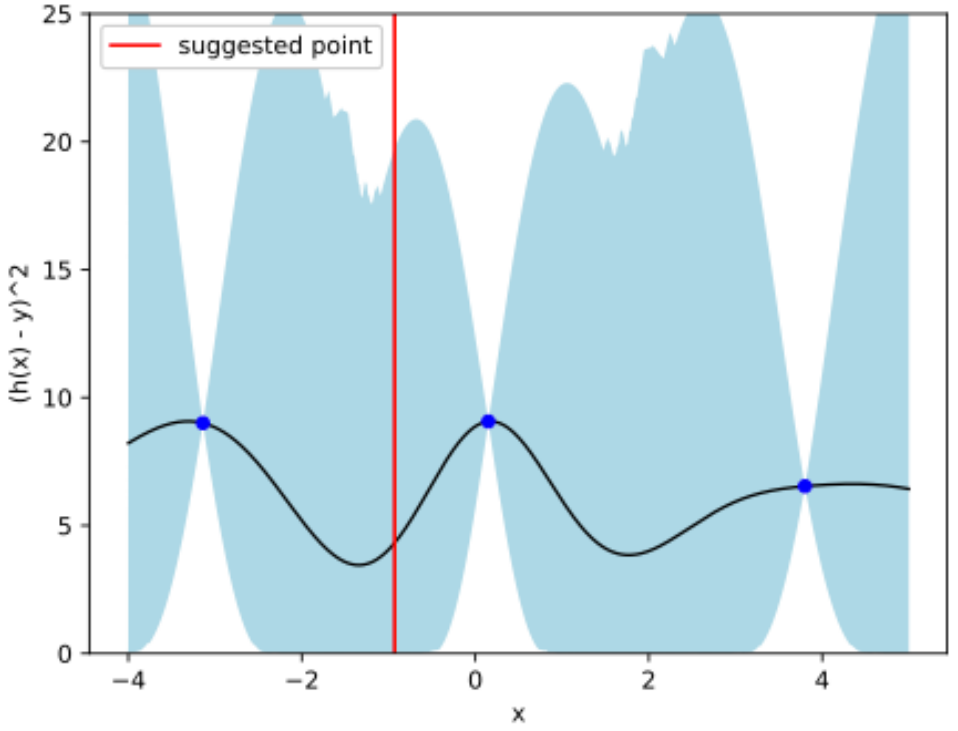


Evaluations of $(h(x) - y)^2$

Now Let's Solve the Same Example using COBALT



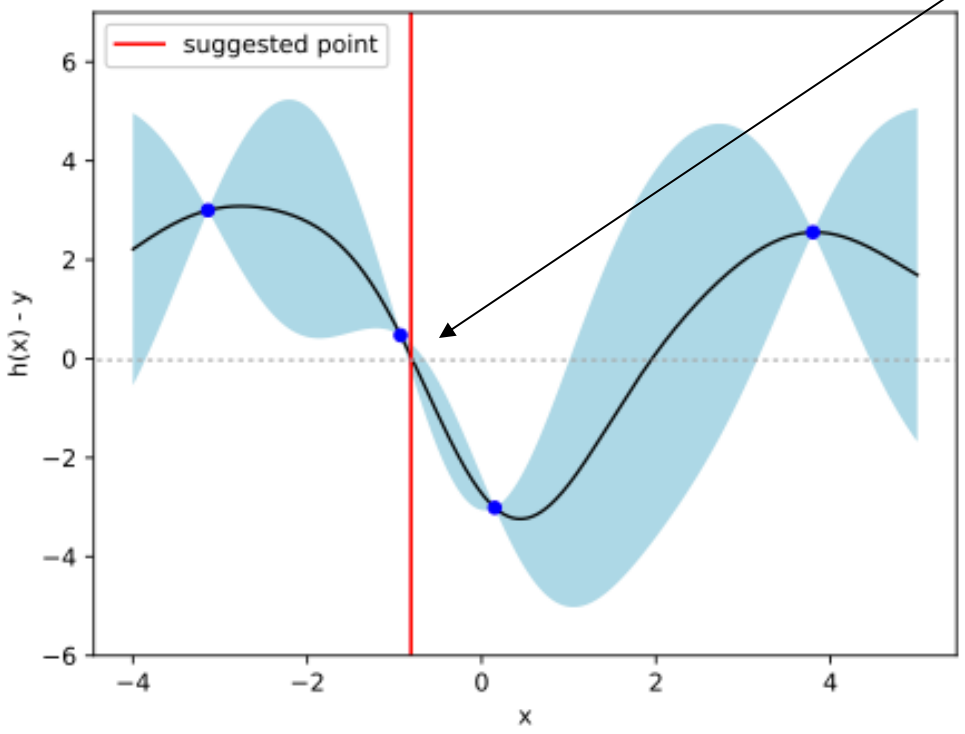
GP posterior on $h(x) - y$
using 3 data points



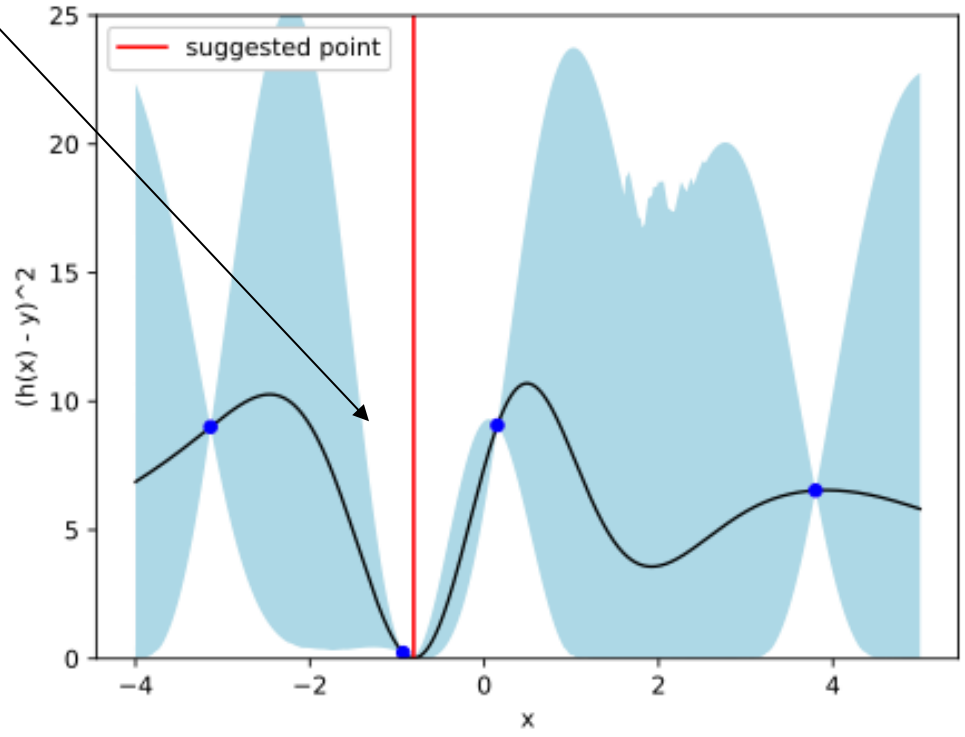
Implied posterior on $(h(x) - y)^2$
using 3 data points

Now Let's Solve the Same Example using COBALT

Notice how COBALT exploits positivity of the loss function to find a value of x that is more likely to make h(x) match y



GP posterior on $h(x) - y$ using 4 data points



Implied posterior on $(h(x) - y)^2$ using 4 data points

Challenge: Maximizing COBALT is Hard!

- In standard Bayesian optimization, prediction of $f(x)$ is Gaussian so that expected improvement (EI) has a closed-form expression
- When prediction of $h(x)$ is Gaussian and $g(\cdot)$ is nonlinear, $f(x) = g(h(x))$ is no longer Gaussian
- COBALT has **no closed form** representation, making it harder to optimize
  - Can use same principles that we discussed for knowledge gradient (KG) to construct efficient algorithms (even when constraints are present)
- General trend: Additional operators provide very useful information but break Gaussian prediction when they are nonlinear → increase complexity

Composite Functions arise in Many Practical Examples

- **Calibration of Expensive Black-box Forward Models**
  - h(x) = prediction of observed data as function of parameters
  - g(h(x)) = negative log-likelihood (+ regularization)

- **Materials Design**
  - h(x) = vector of different material attributes
  - g(h(x)) = combined performance measure over attributes

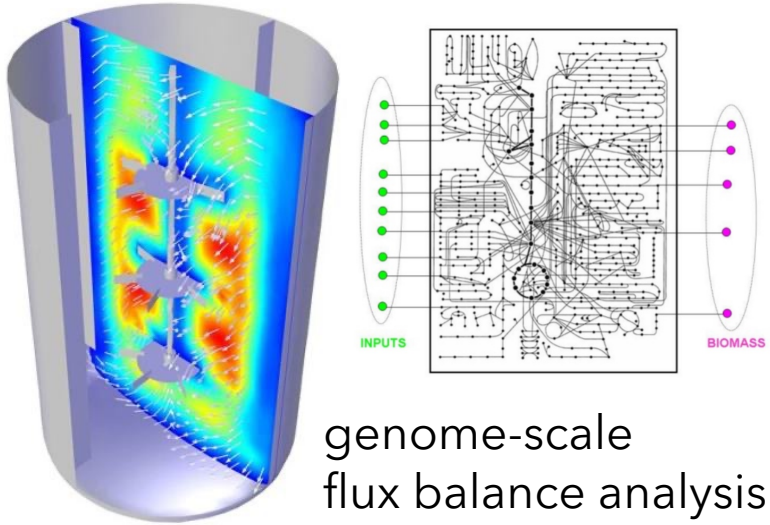
- **Process Flowsheet Optimization**
  - h(x) = reaction & separation efficiency as function of temp. and concentration
  - g(h(x)) = return on investment

Genome-scale Bioreactor Model Calibration Test Problem

Experimental measurements

[Hanly, Urello, and Henson, 2012]

Expensive computer simulation



genome-scale
flux balance analysis

Compare

- Optimize simulation's parameters such that **log-likelihood** is maximized
  - six parameters related to bounds on extracellular uptake rate

Genome-scale Bioreactor Model Calibration Test Problem

Negative log-likelihood is a composite function:

$$f(x) = g(h(x)) = \sum_{j=1}^{N} \log(0.0025 h_j^2(x)) + 400 h_j^{-2}(x)(y_j - h_j(x))^2$$

these terms are related to measurement noise term that depends on concentration

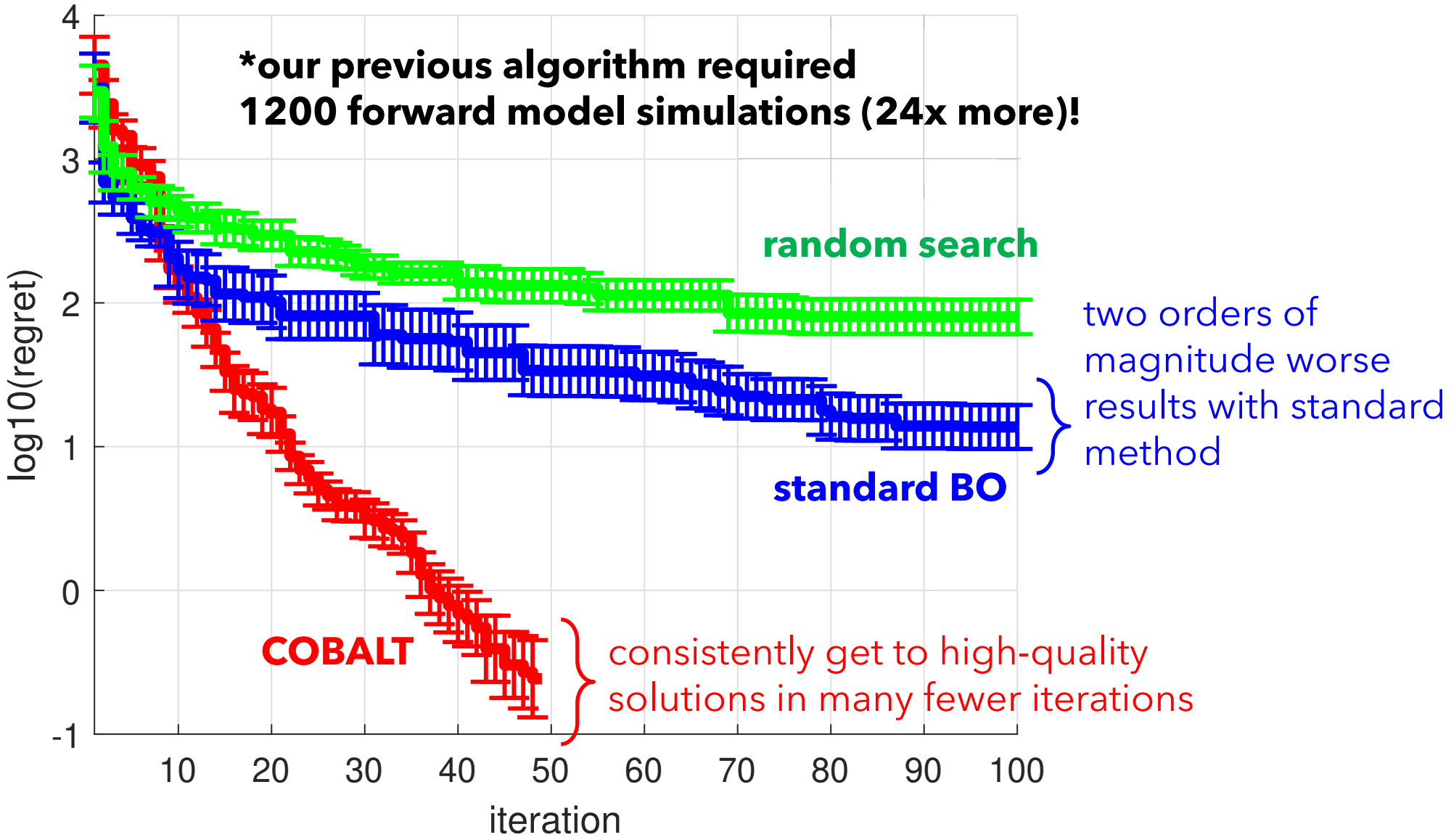
where x is the vector of parameters that must be estimated

y_j is the j[th] measured datapoint (e.g., extracellular concentrations)

$h_j(x)$ is the forward model prediction for the j[th] measurement

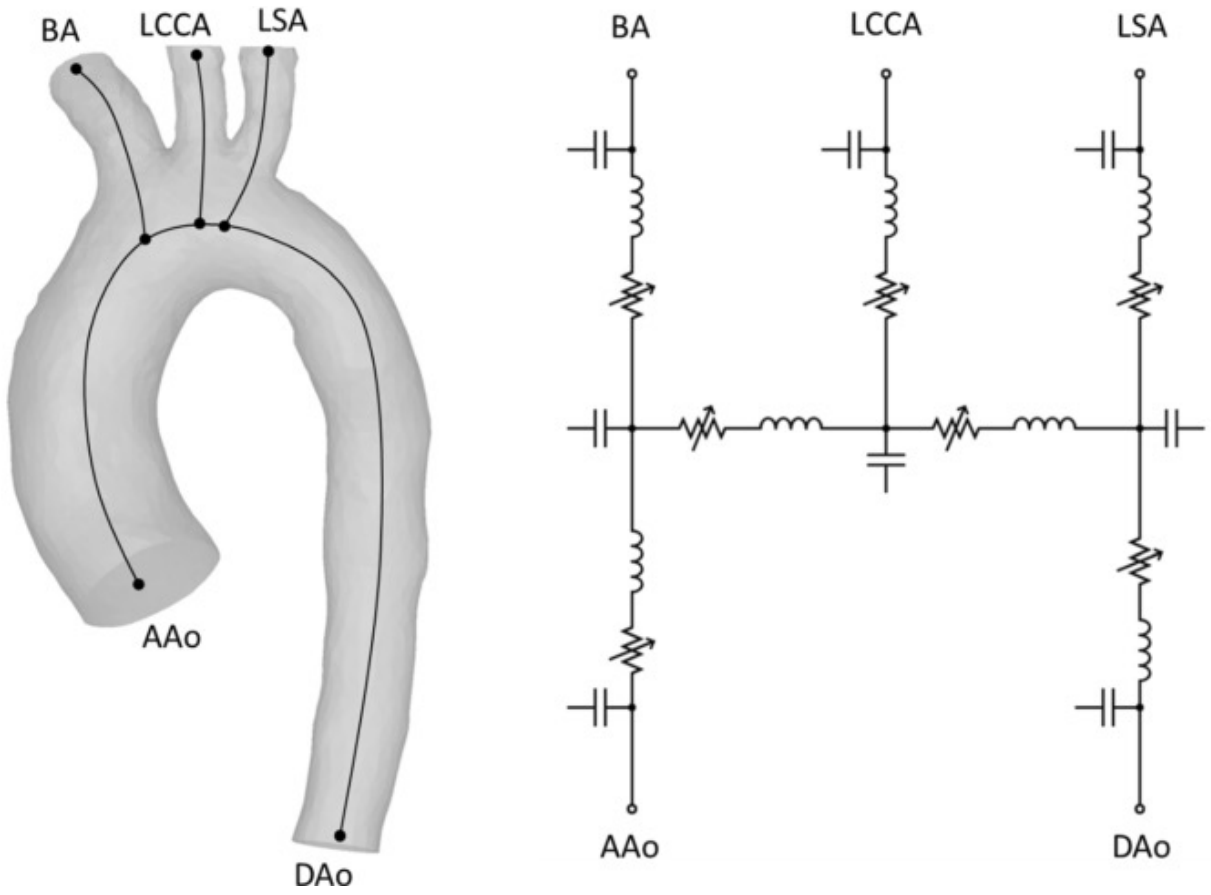
[Paulson et al., 2019]

Results: Log10(Regret) versus Number of Evaluations



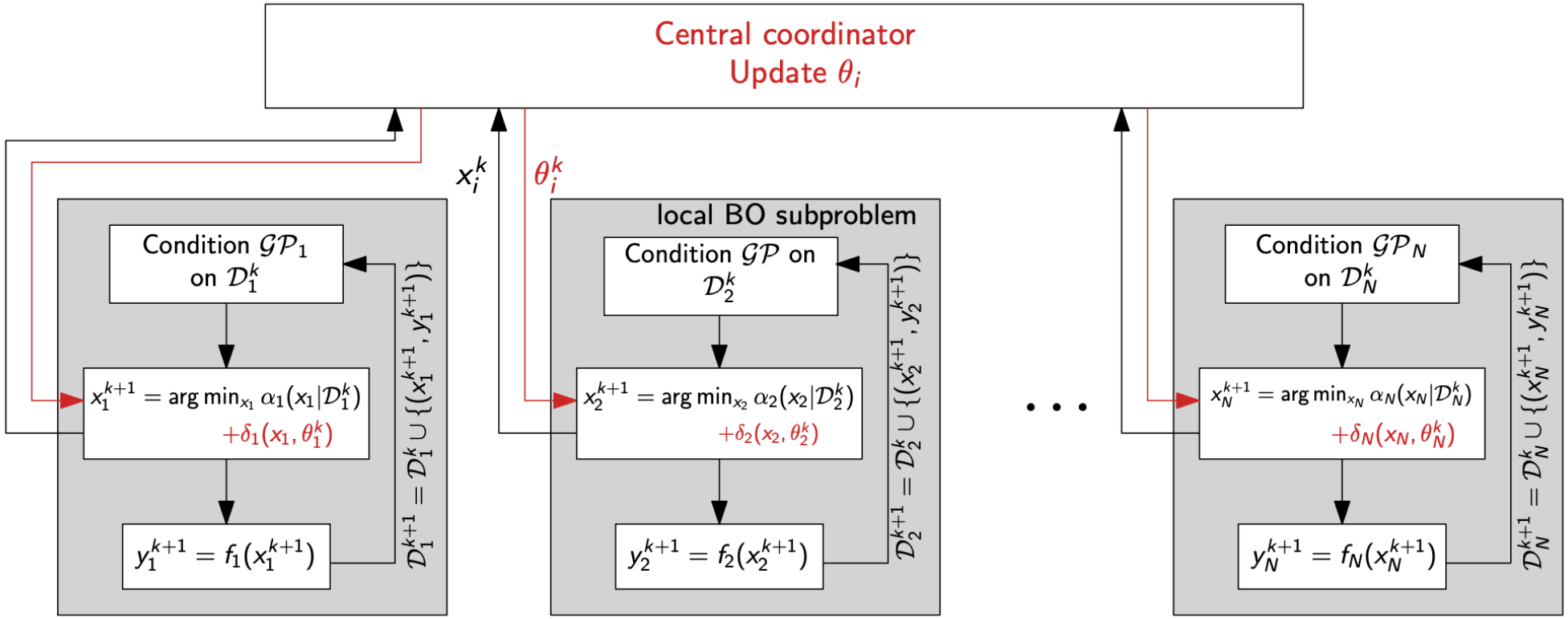
A Few Practical Examples

We can extend Bayesian Optimization to multiple "models" with different accuracies & costs



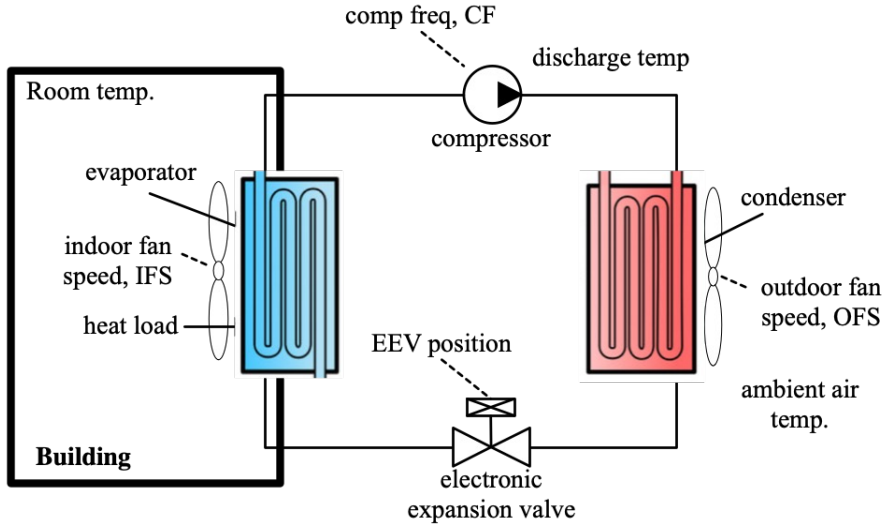
We can extend Bayesian Optimization to multi-agent problems

$$\min_{x \in \mathcal{X}} \; f(x) = \sum_{i=1}^{N} f_i(x)$$

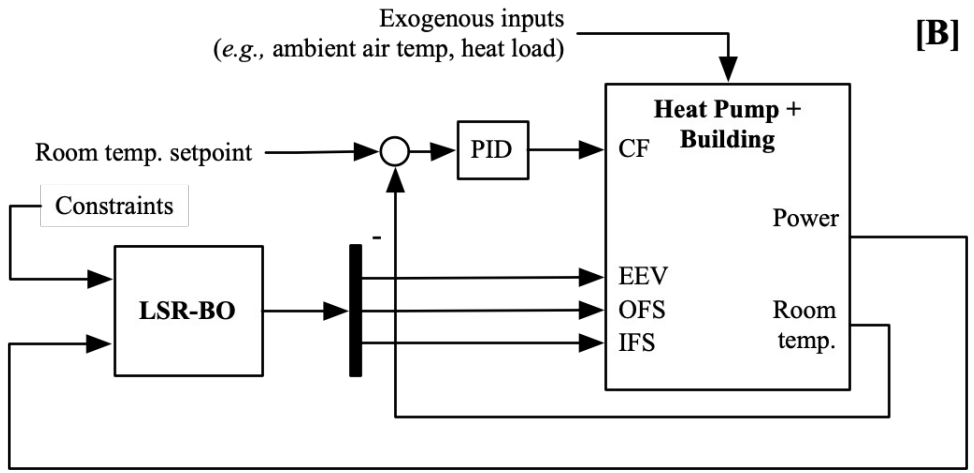


D. Krishnamoorthy and J.A. Paulson, "Multi-agent Black-box Optimization using a Bayesian Approach to Alternating Direction Method of Multipliers", IFAC World Congress 2023 (accepted), https://arxiv.org/pdf/2303.14414.pdf

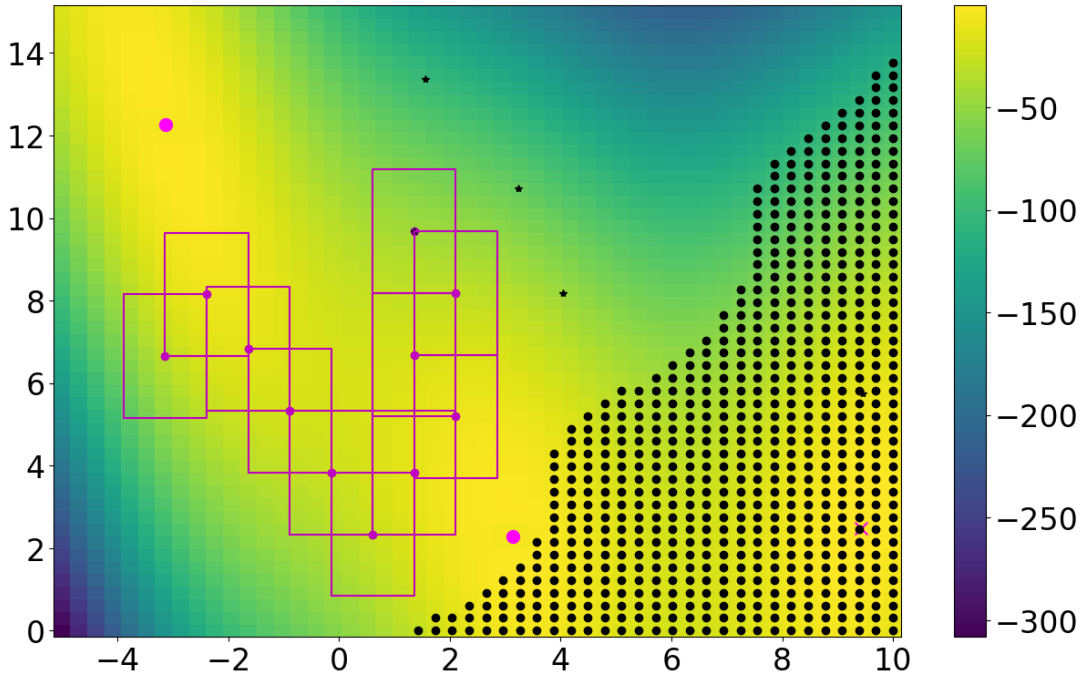
We can extend Bayesian optimization to work in real-world experimental systems (safety)



[A]



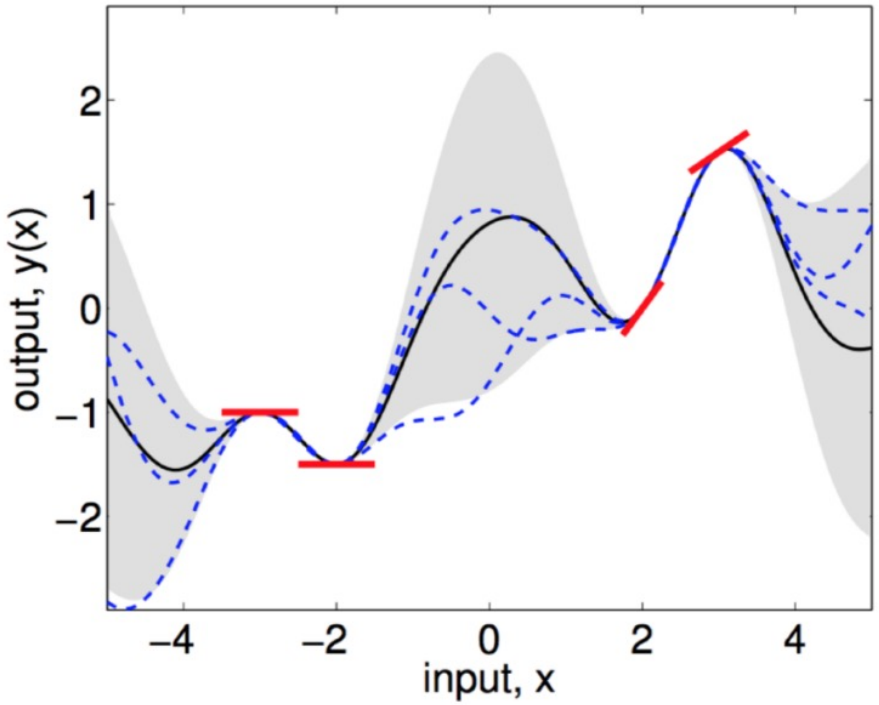
[B]



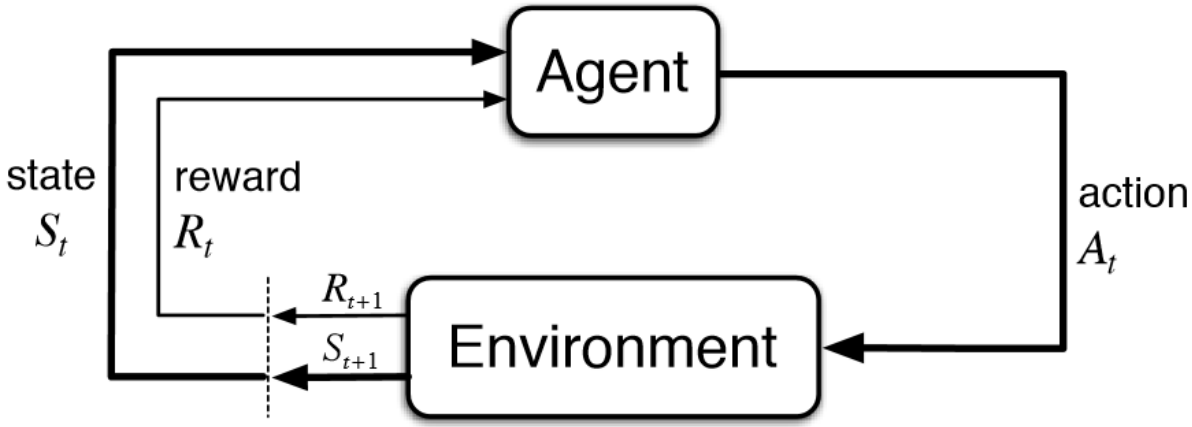
MITSUBISHI ELECTRIC
Changes for the Better

We can extend Bayesian Optimization to handle (partial) gradient information

GP regression naturally handles gradients (since it is linear operator)



We can estimate gradients in reinforcement learning (policy gradient theorem)



Thanks for your attention

Questions?