

# Active Learning for Level Set Estimation

Alkis Gotovos<sup>1</sup> Nathalie Casati<sup>1,2</sup> Gregory Hitz<sup>1</sup> Andreas Krause<sup>1</sup>

<sup>1</sup>Department of Computer Science  
ETH Zurich

<sup>2</sup>IBM Research – Zurich

IJCAI '13

# Swimmers of Lake Zurich, beware!



Steffen Schmidt / EPA

## Swimmers of Lake Zurich, beware!

“The warming waters of one of central Europe's most popular holiday destinations, Switzerland's Lake Zurich, have created an ideal environment for a population explosion of algae including *Planktothrix rubescens*, [...]”

— *Scientific American*

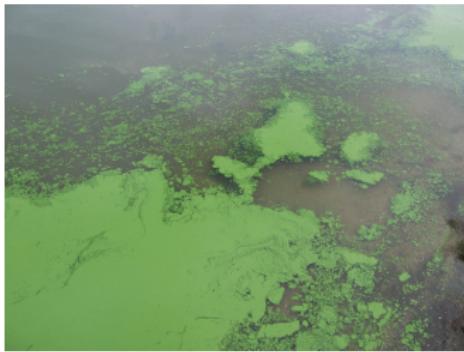
## Swimmers of Lake Zurich, beware!



[www.limnobiotics.ch](http://www.limnobotics.ch)

"The warming waters of one of central Europe's most popular holiday destinations, Switzerland's Lake Zurich, have created an ideal environment for a population explosion of algae including *Planktothrix rubescens*, [...]"

— *Scientific American*



Flickr/Dr. Jennifer L. Graham/U.S. Geological Survey

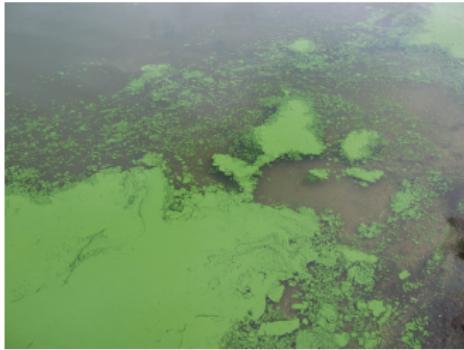
## Swimmers of Lake Zurich, beware!



[www.limnobotics.ch](http://www.limnobotics.ch)

"The warming waters of one of central Europe's most popular holiday destinations, Switzerland's Lake Zurich, have created an ideal environment for a population explosion of algae including *Planktothrix rubescens*, [...]"

— *Scientific American*



Flickr/Dr. Jennifer L. Graham/U.S. Geological Survey

"*Planktothrix rubescens* can account for half of the total phytoplankton biomass in Lake Zurich in summer [...]"

"*Planktothrix rubescens* are among the most important producers of hepatotoxic microcystins in freshwaters [...]"

— *Silke Van den Wyngaert et al., ASLO, 2011*

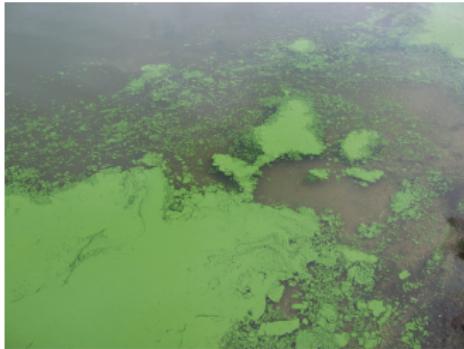
## Swimmers of Lake Zurich, beware!



[www.limnobiotics.ch](http://www.limnobotics.ch)

"The warming waters of one of central Europe's most popular holiday destinations, Switzerland's Lake Zurich, have created an ideal environment for a population explosion of algae including *Planktothrix rubescens*, [...]"

— *Scientific American*



Flickr/Dr. Jennifer L. Graham/U.S. Geological Survey

"*Planktothrix rubescens* can account for half of the total phytoplankton biomass in Lake Zurich in summer [...]"

"*Planktothrix rubescens* are among the most important producers of hepatotoxic microcystins in freshwaters [...]"

— *Silke Van den Wyngaert et al., ASLO, 2011*

"Microcystins [...] are cyanotoxins and can be very toxic for plants and animals including humans. Their hepatotoxicity may cause serious damage to the liver."

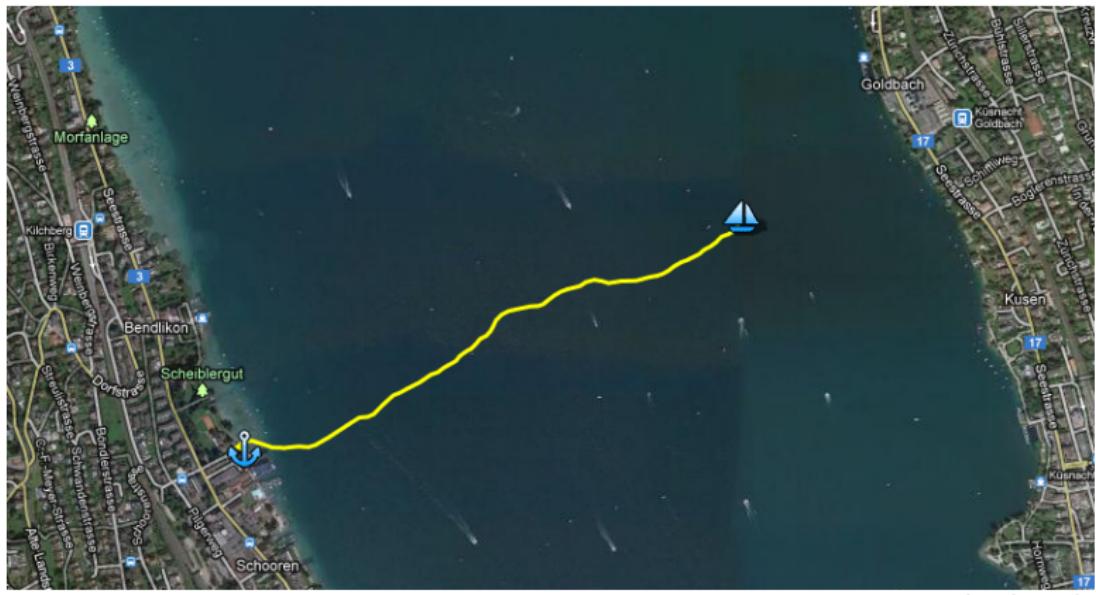
— *Wikipedia*



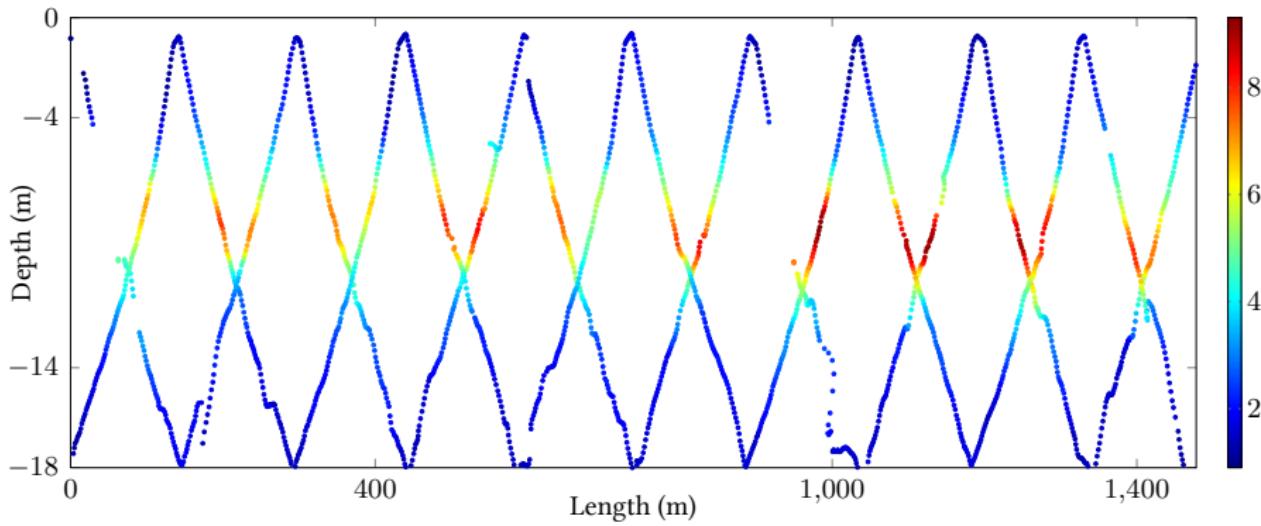
[www.limnobotics.ch](http://www.limnobotics.ch)

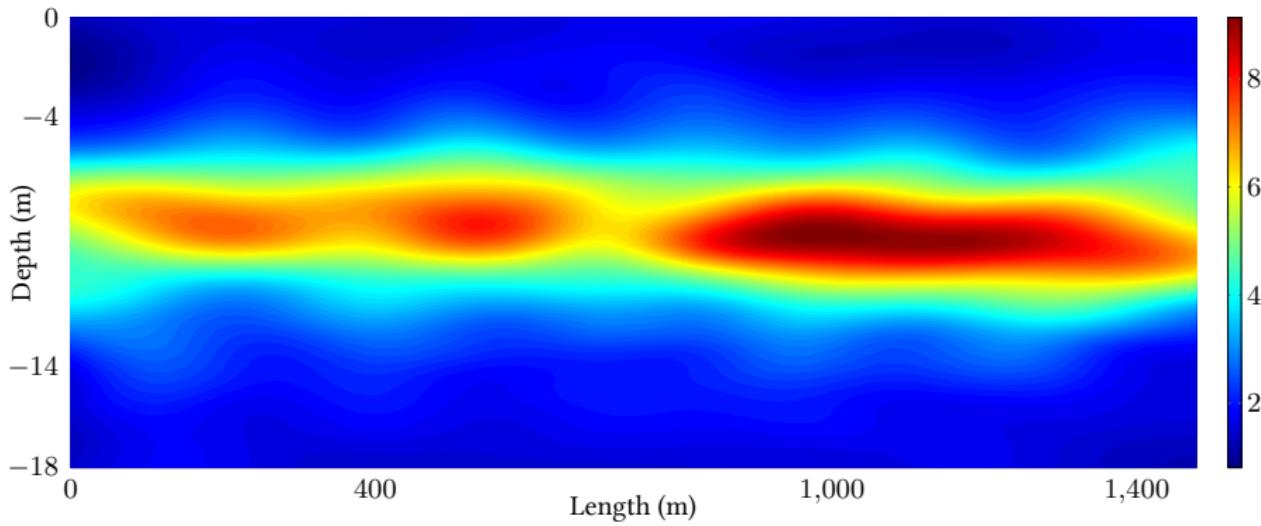


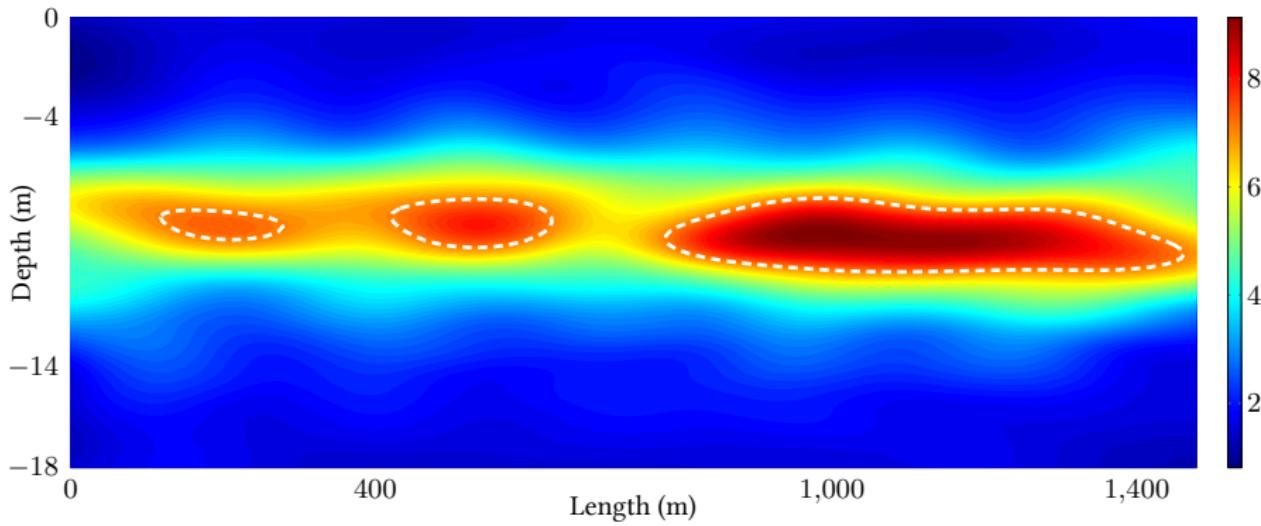
[www.limnobotics.ch](http://www.limnobotics.ch)

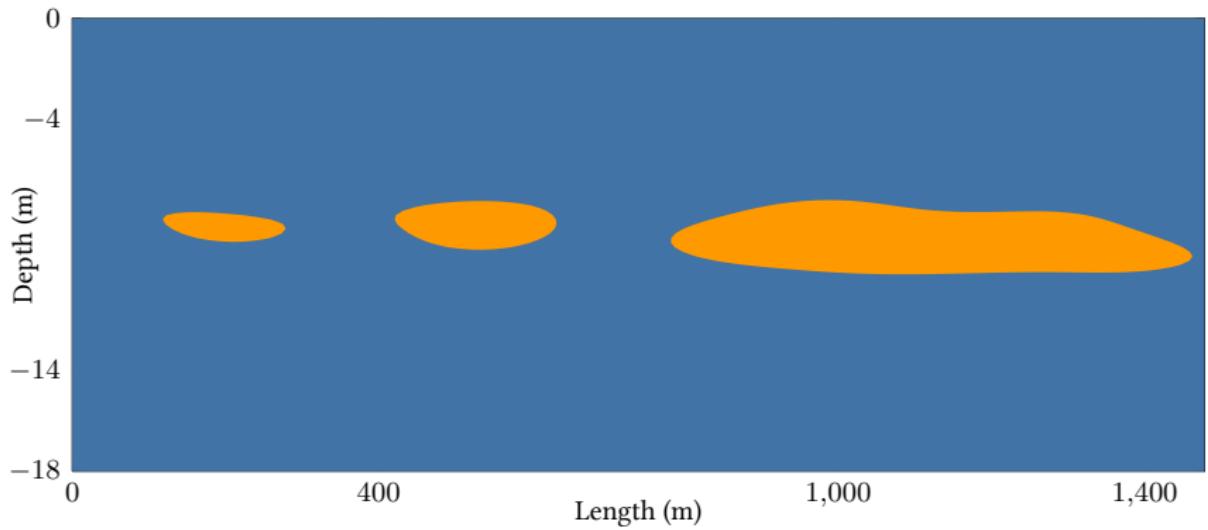


[www.limnobotics.ch](http://www.limnobotics.ch)









Pose as a sequential decision making problem (*pool-based active learning*):

- ▶ No measurements available in advance, just a set (pool) of possible sampling locations ( $D$ )

Pose as a sequential decision making problem (*pool-based active learning*):

- ▶ No measurements available in advance, just a set (pool) of possible sampling locations ( $D$ )

At each iteration  $t \geq 1$ :

- ▶ Decide where to measure next ( $x_t \in D$ )

Pose as a sequential decision making problem (*pool-based active learning*):

- ▶ No measurements available in advance, just a set (pool) of possible sampling locations ( $D$ )

At each iteration  $t \geq 1$ :

- ▶ Decide where to measure next ( $\mathbf{x}_t \in D$ )
- ▶ Obtain noisy observation ( $y_t = f(\mathbf{x}_t) + n_t$ )

Pose as a sequential decision making problem (*pool-based active learning*):

- ▶ No measurements available in advance, just a set (pool) of possible sampling locations ( $D$ )

At each iteration  $t \geq 1$ :

- ▶ Decide where to measure next ( $\mathbf{x}_t \in D$ )
- ▶ Obtain noisy observation ( $y_t = f(\mathbf{x}_t) + n_t$ )
- ▶ Update our estimate of  $f$

## 1. How do we **estimate** the function?

1. How do we **estimate** the function?
2. How do we **classify**?

1. How do we **estimate** the function?
2. How do we **classify**?
3. Each measurement is expensive (time, battery power).  
How do we **select** “informative” measurements?

1. How do we **estimate** the function?
2. How do we **classify**?
3. Each measurement is expensive (time, battery power).  
How do we **select** “informative” measurements?

Gaussian processes to the rescue!

## Gaussian processes

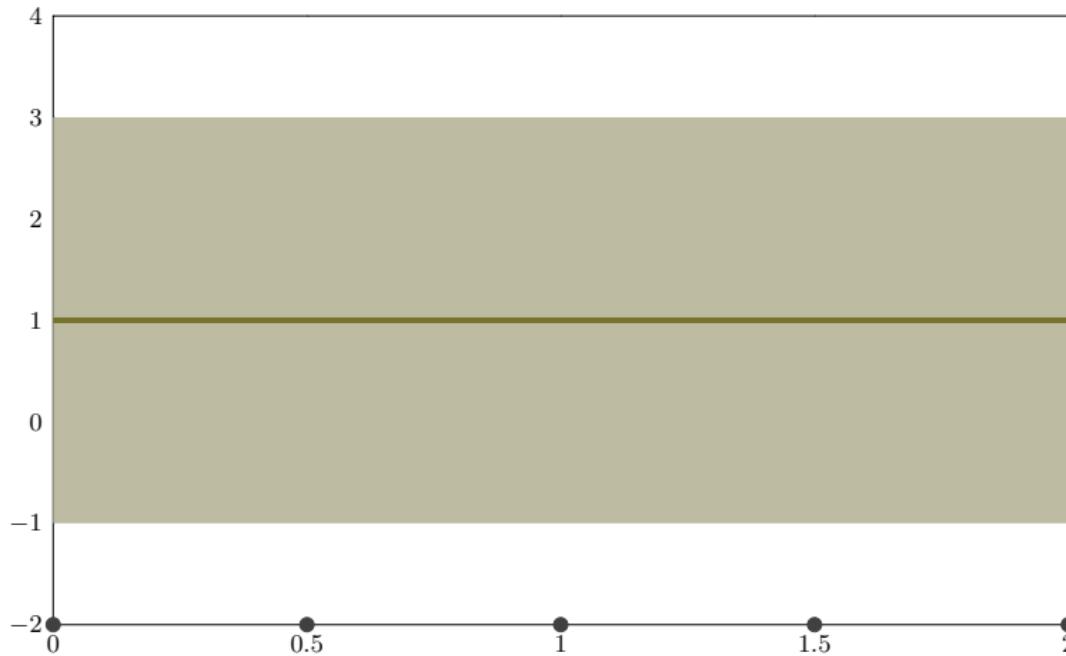
- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$

## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$

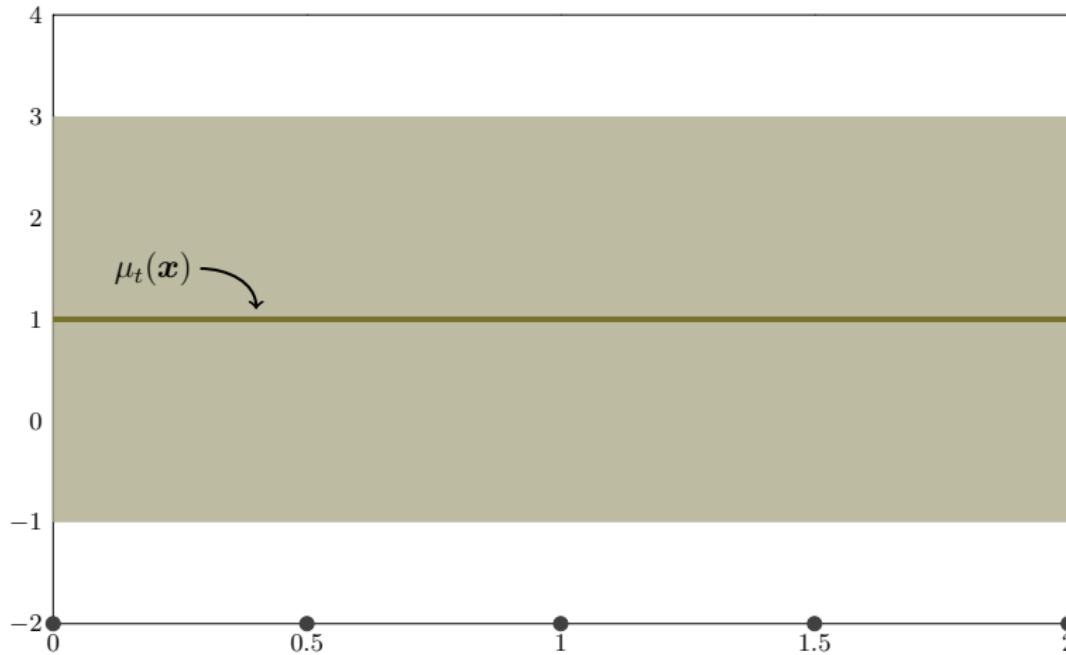
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$



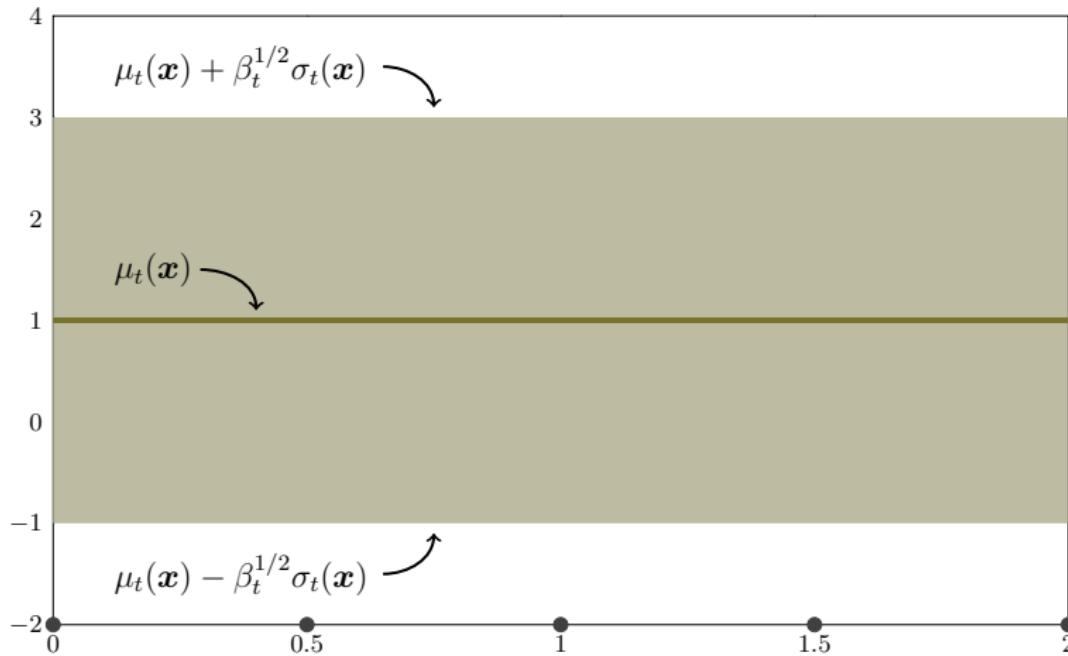
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$



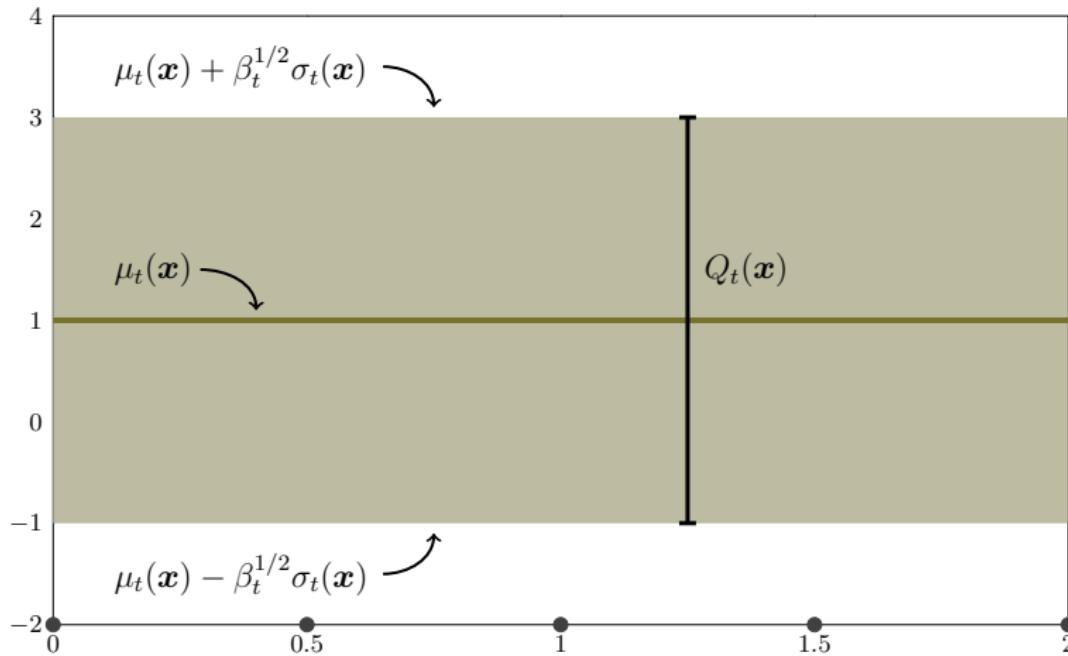
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$



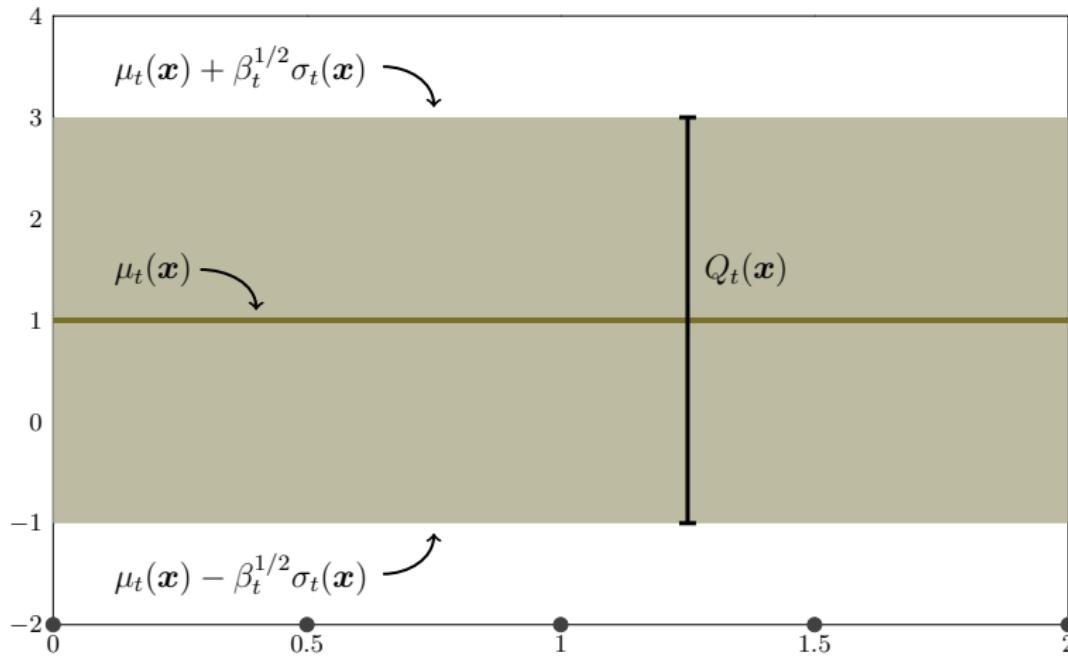
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance estimates:** construct confidence intervals  $Q_t(\mathbf{x})$



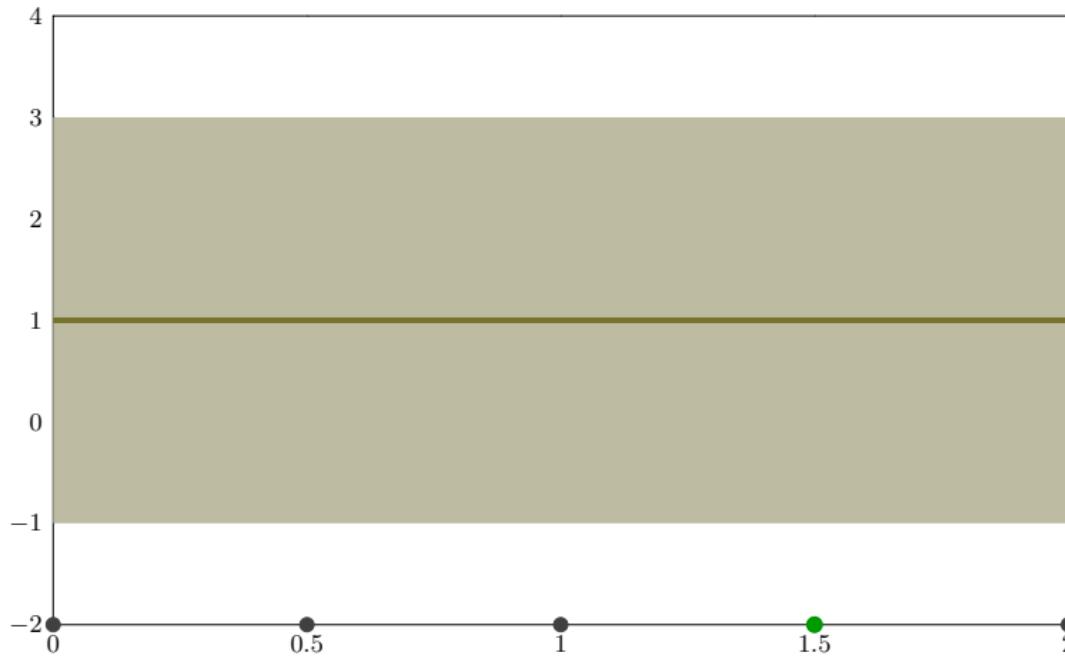
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates



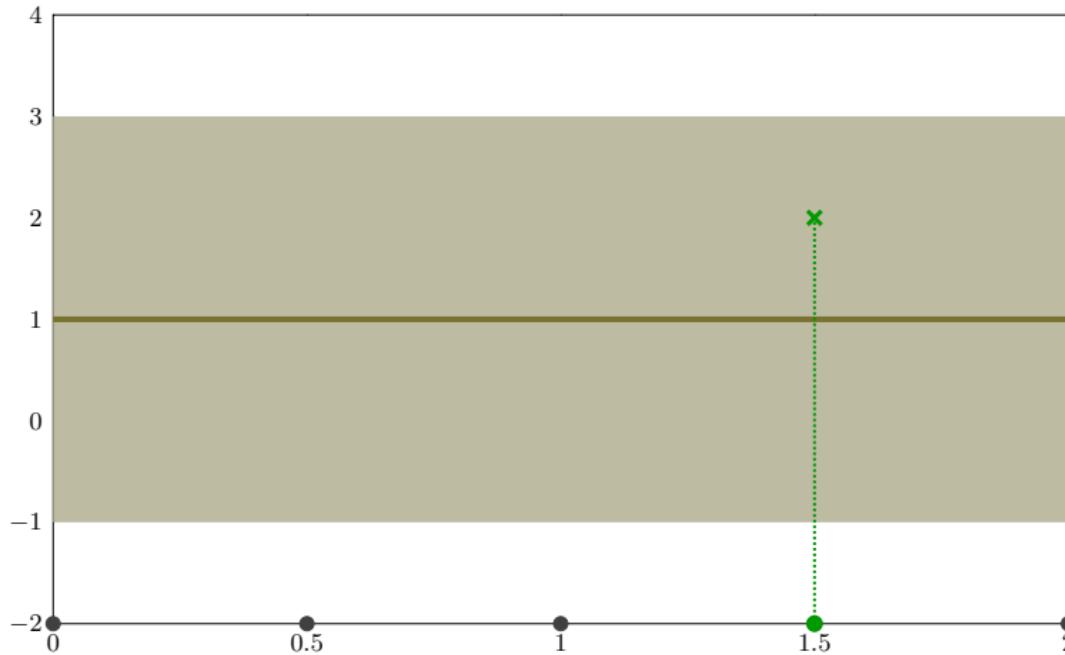
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates



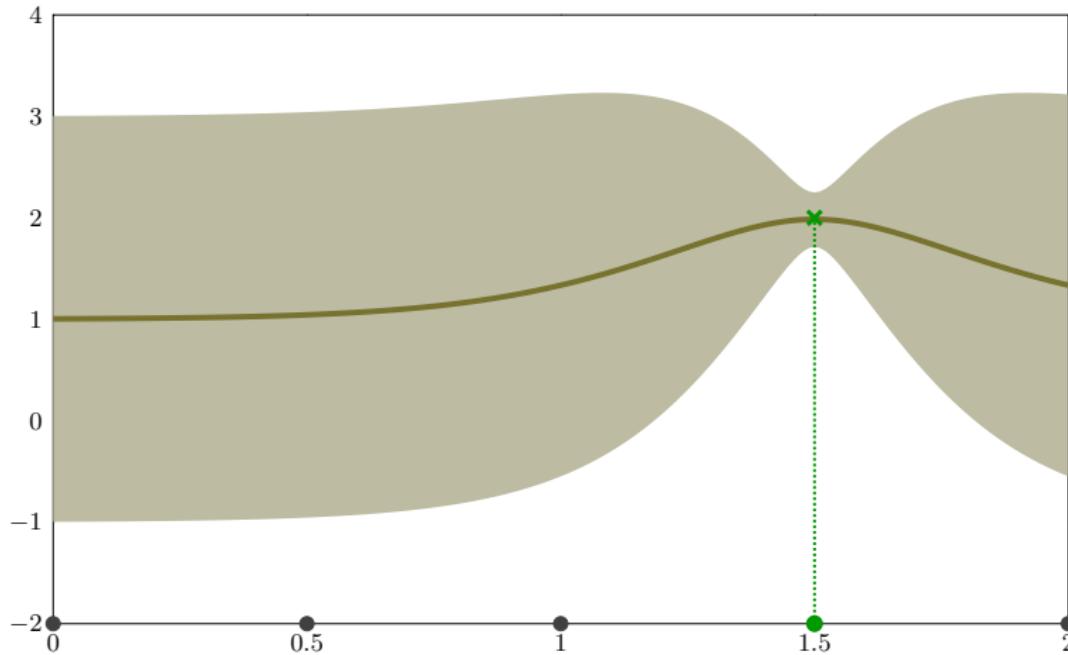
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates



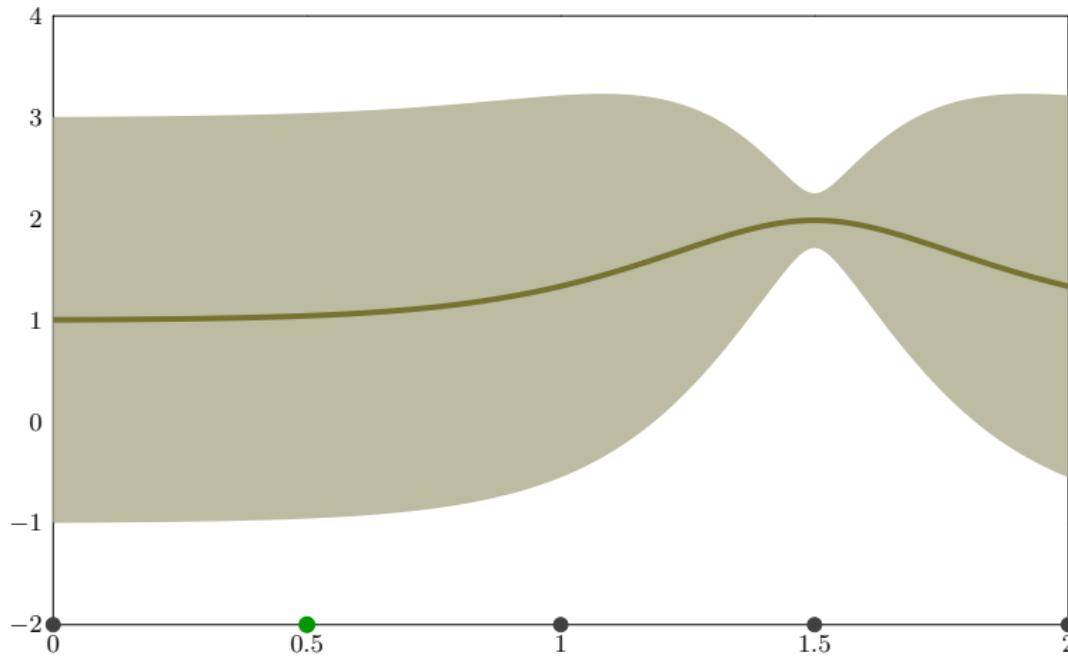
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates



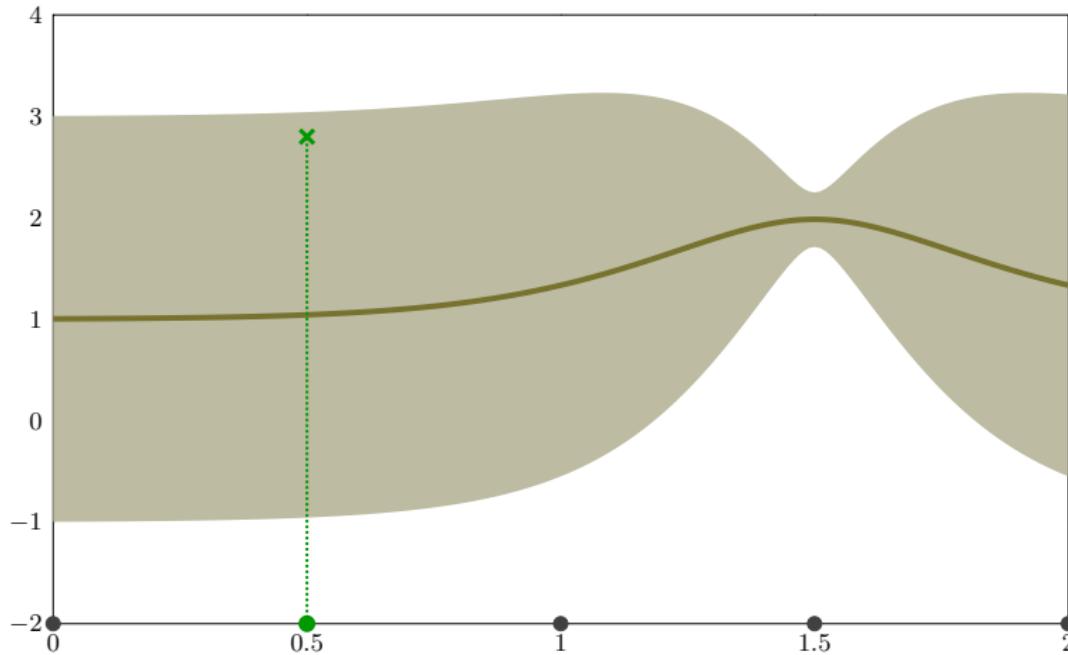
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates



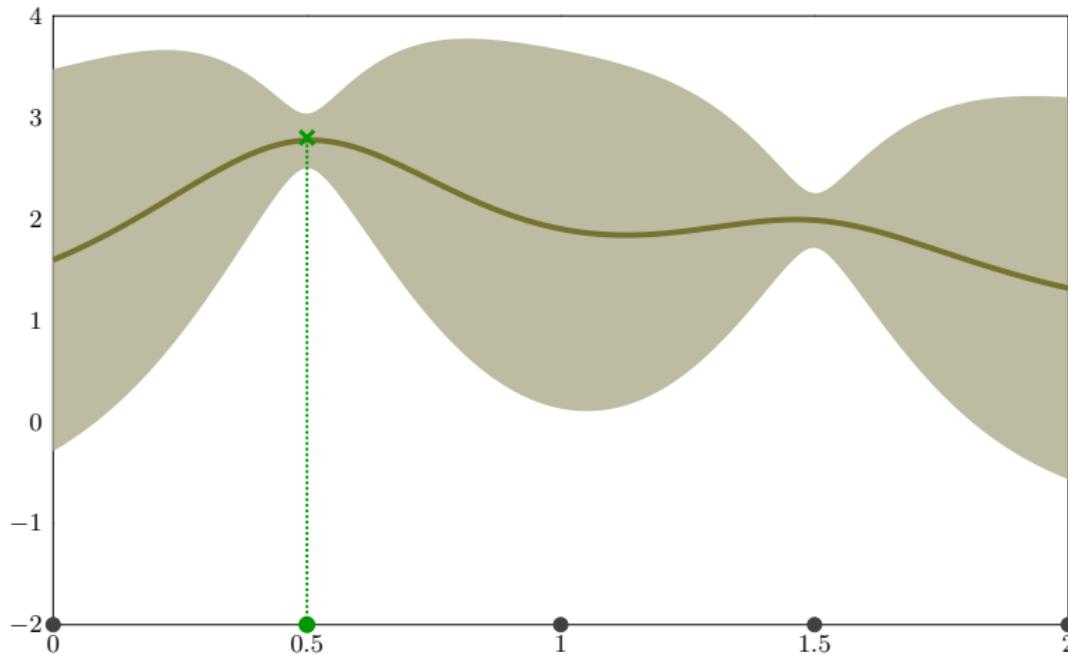
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates



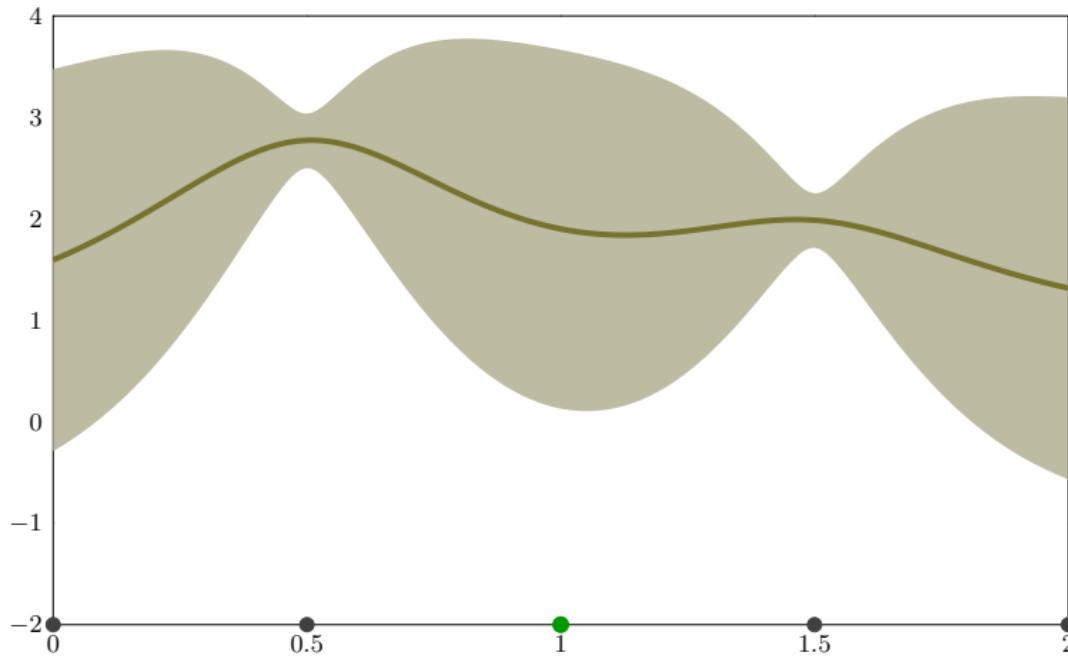
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates



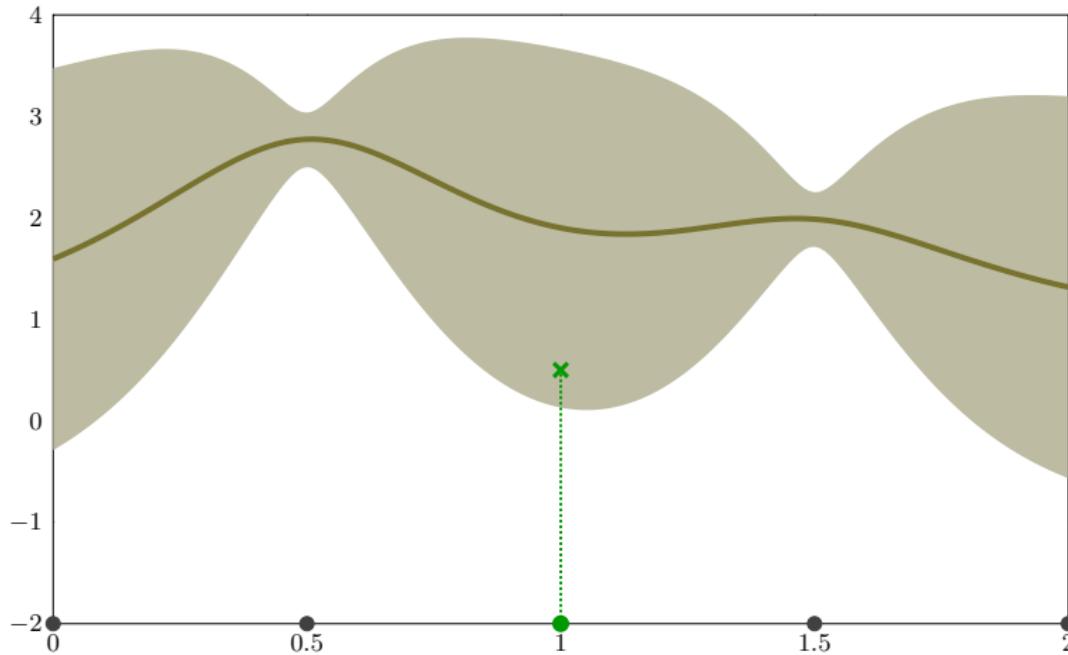
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates



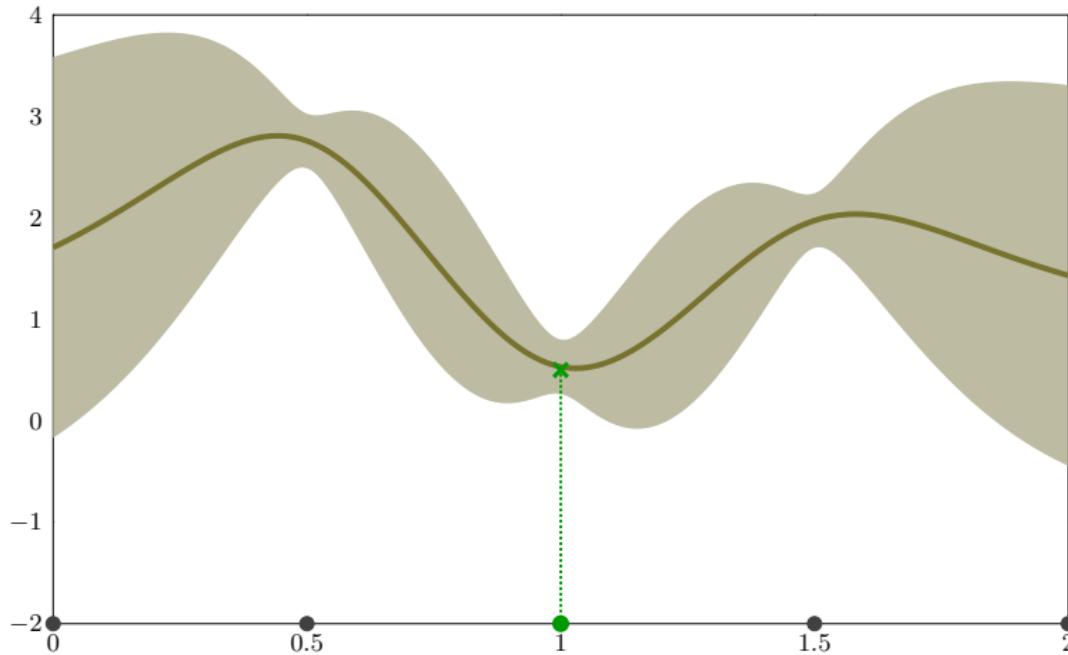
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates



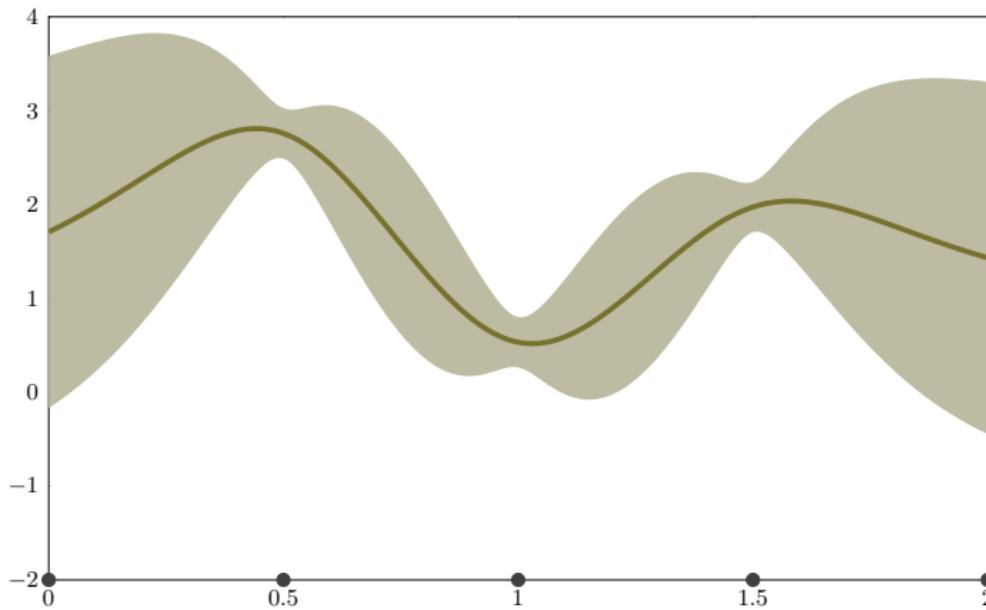
## Gaussian processes

- ▶ **Nonparametric:** impose “smoothness” assumptions via kernel  $k(\mathbf{x}, \mathbf{x}')$
- ▶ **Mean and variance** estimates: construct confidence intervals  $Q_t(\mathbf{x})$
- ▶ **Bayesian, yet efficient:** suitable for step-by-step updates

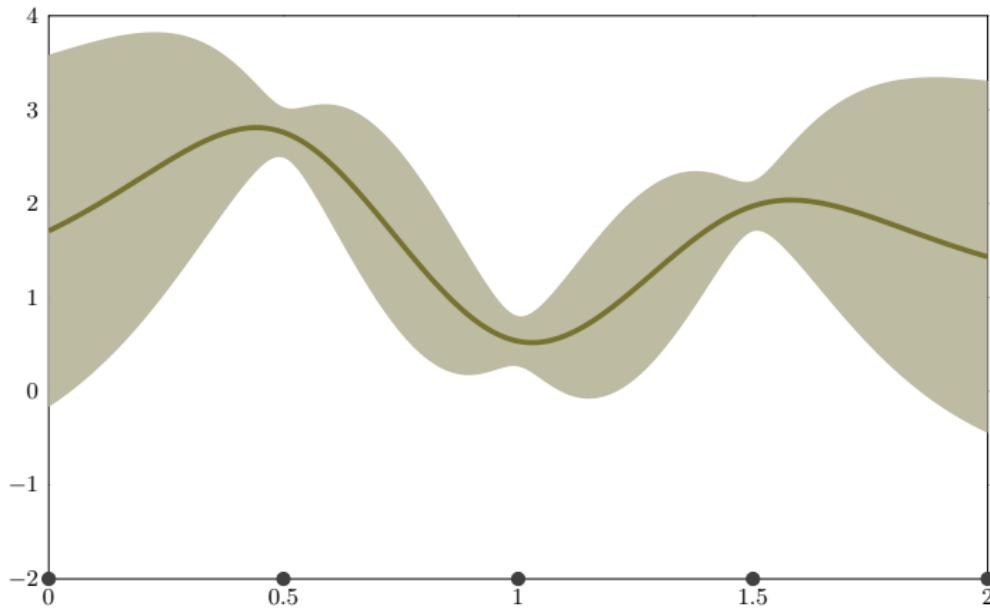


## 1. How do we **estimate** the function?

# 1. How do we **estimate** the function?

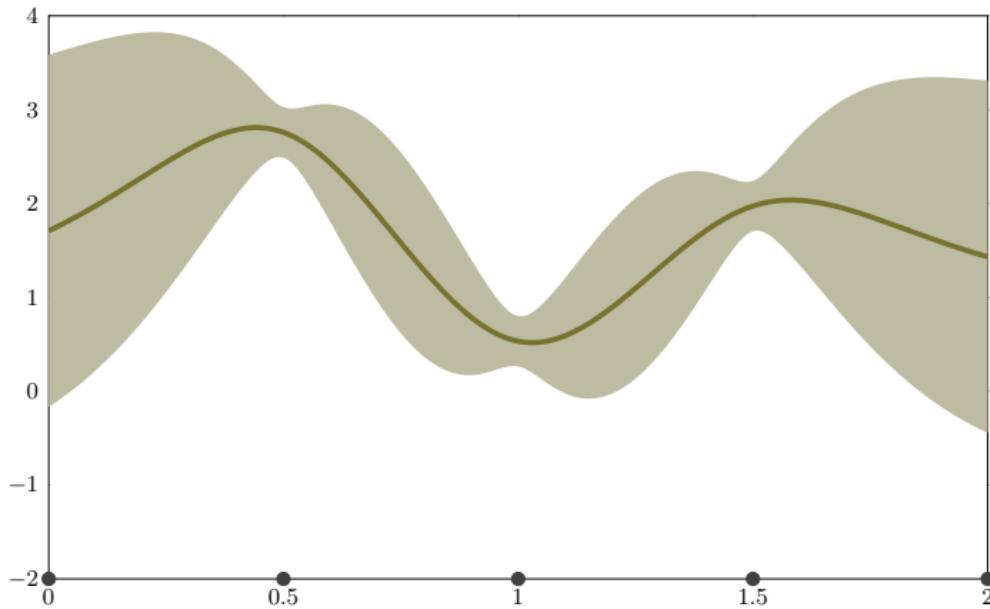


# 1. How do we **estimate** the function? ✓



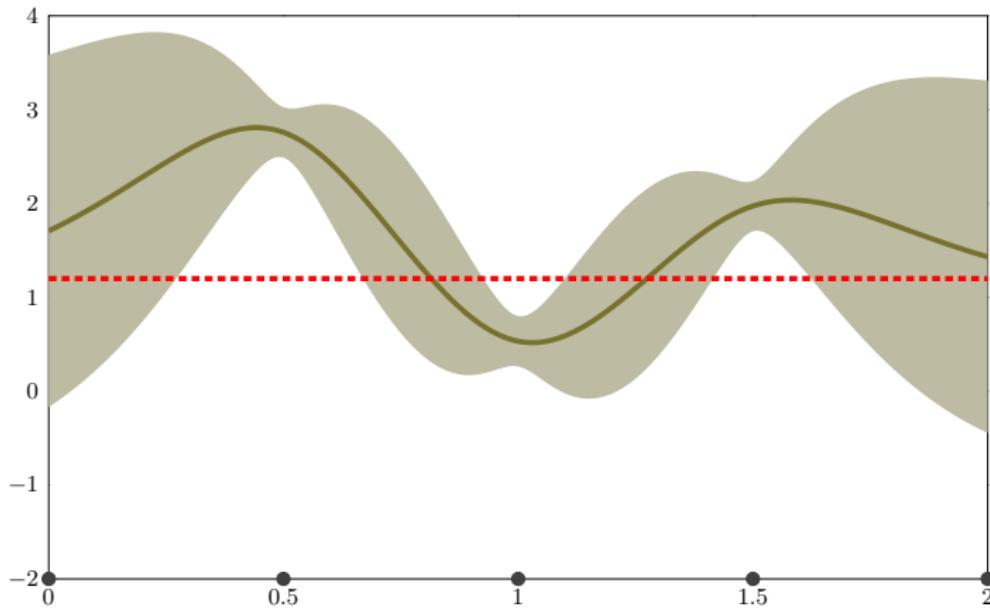
1. How do we **estimate** the function? ✓

2. How do we **classify**?



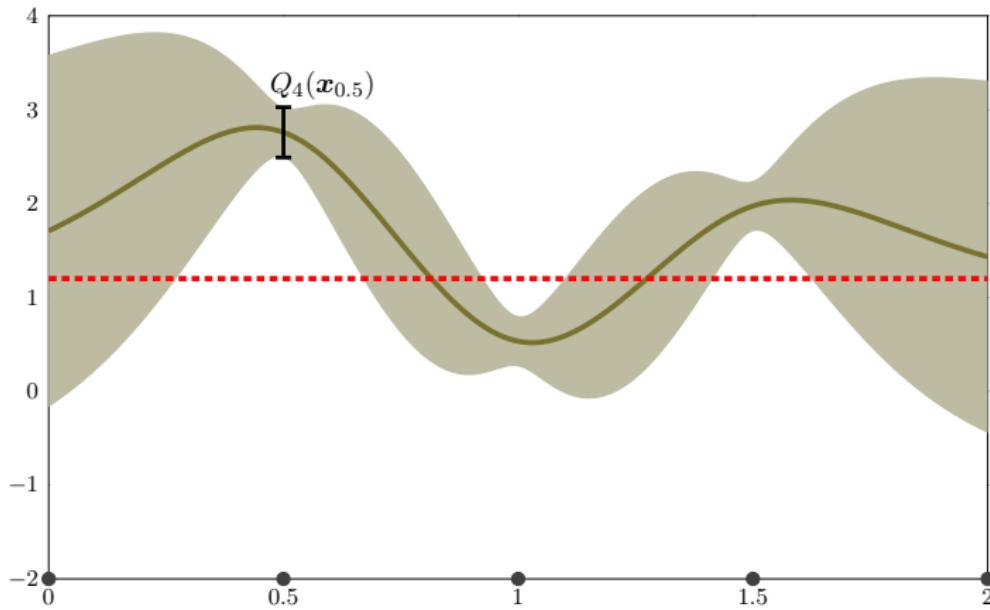
1. How do we **estimate** the function? ✓

2. How do we **classify**?



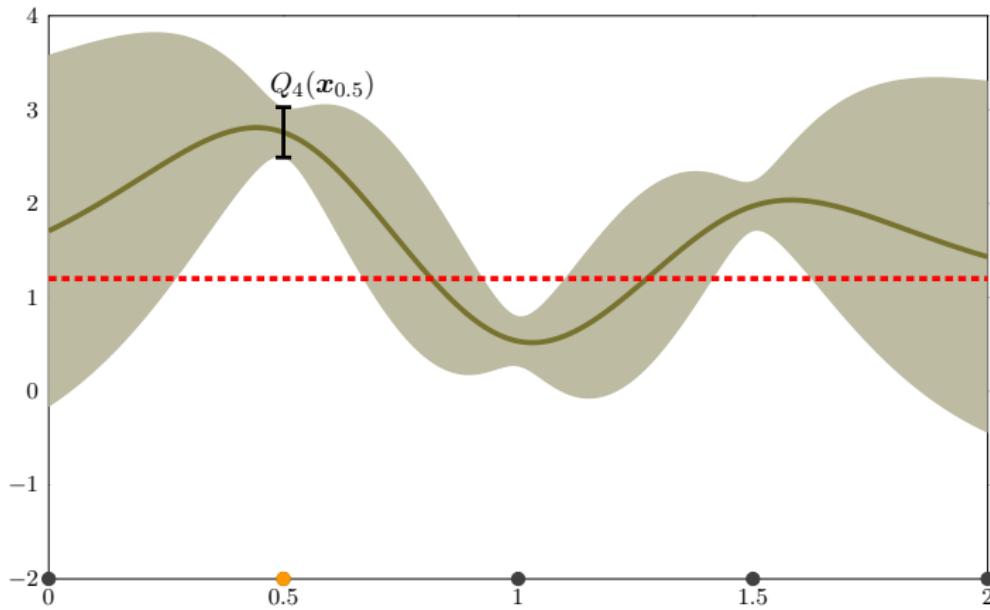
1. How do we **estimate** the function? ✓

2. How do we **classify**?



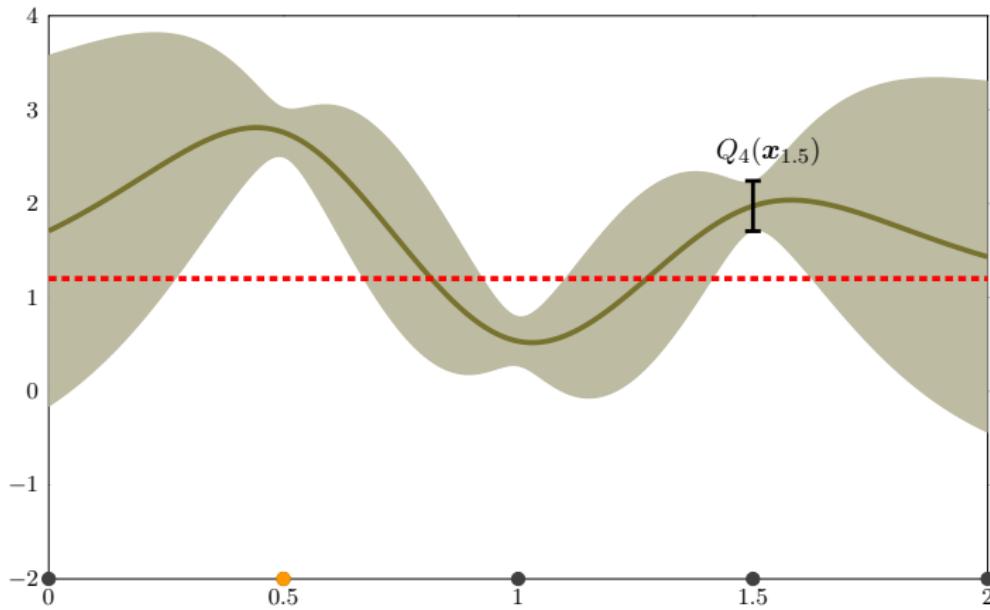
1. How do we **estimate** the function? ✓

2. How do we **classify**?



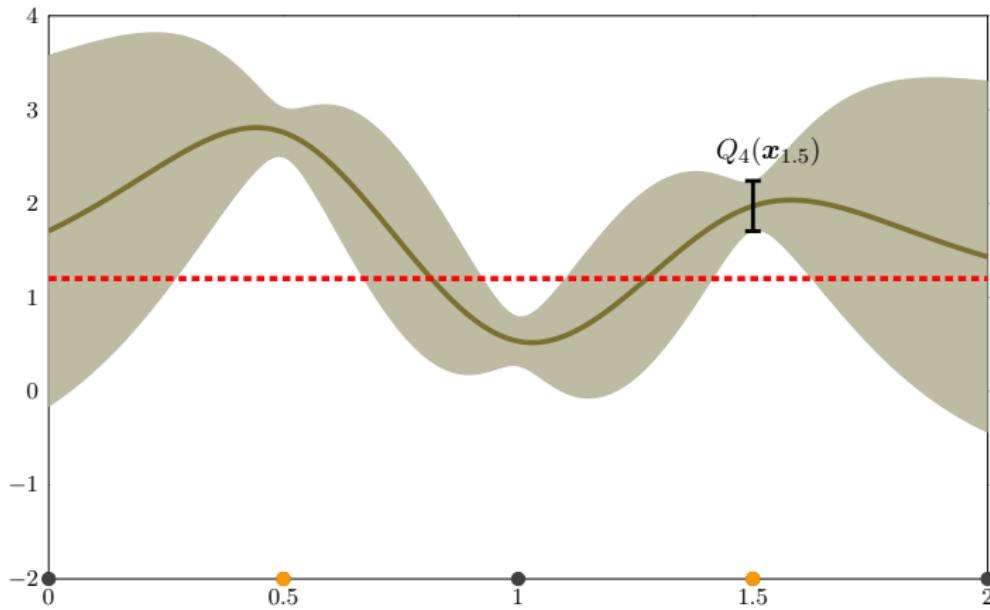
1. How do we **estimate** the function? ✓

2. How do we **classify**?



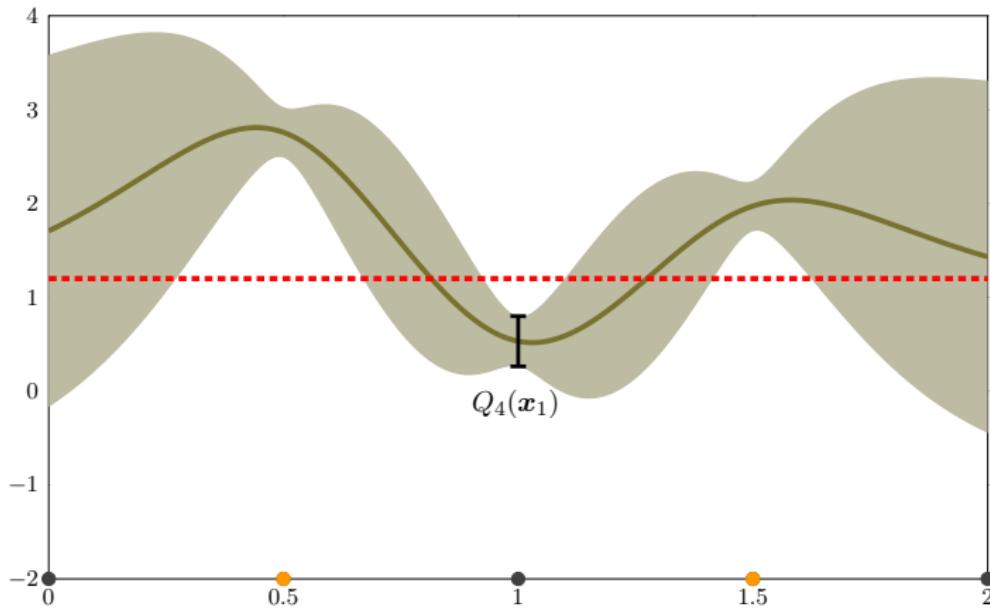
1. How do we **estimate** the function? ✓

2. How do we **classify**?

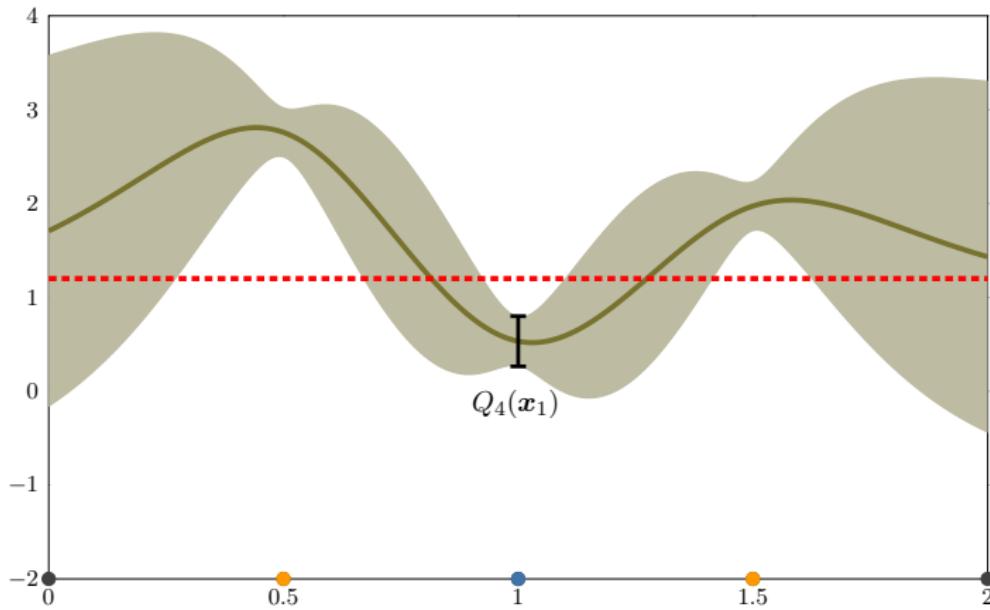


1. How do we **estimate** the function? ✓

2. How do we **classify**?

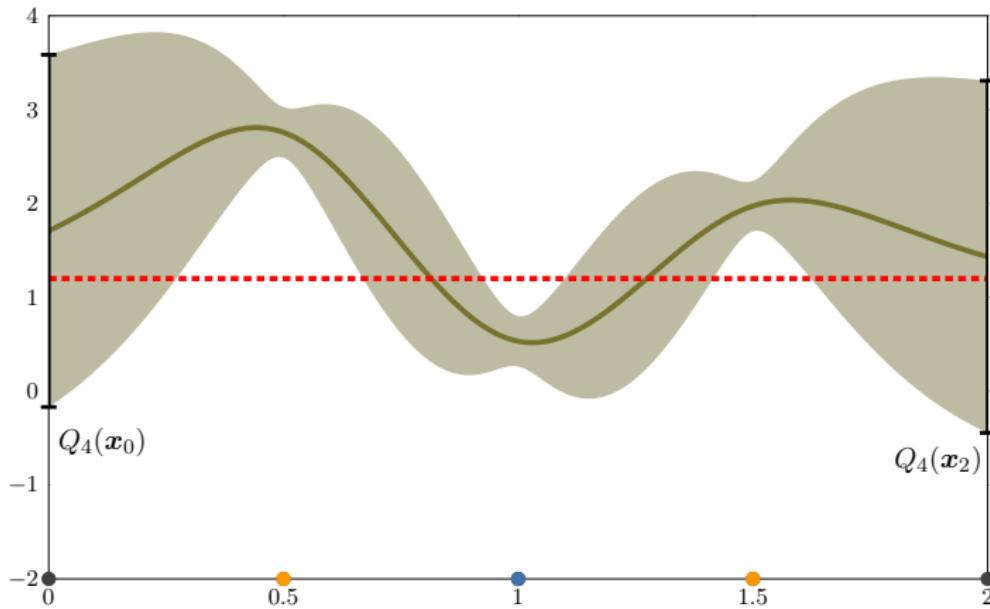


1. How do we **estimate** the function? ✓
  2. How do we **classify**?



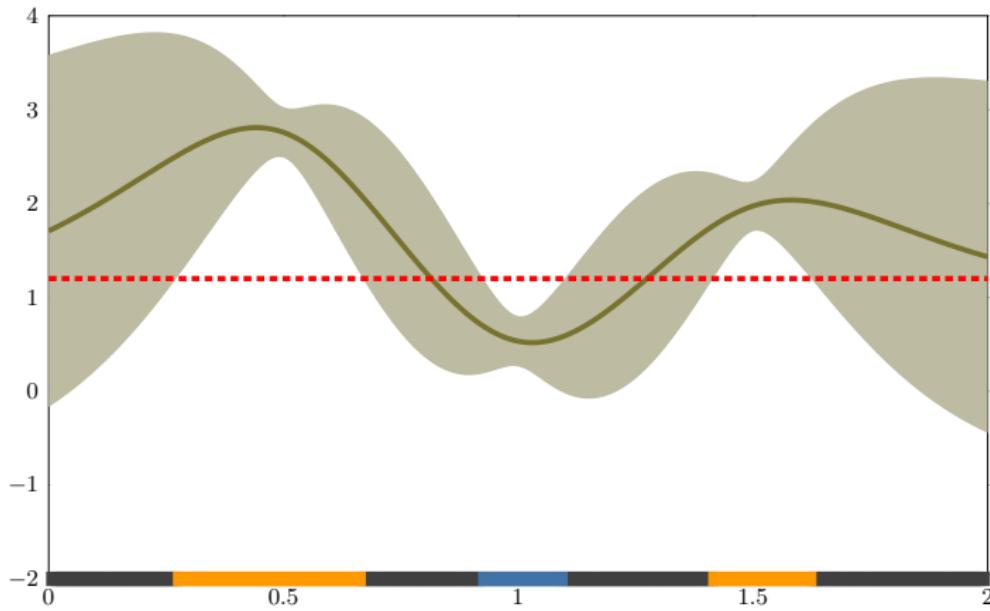
1. How do we **estimate** the function? ✓

2. How do we **classify**?



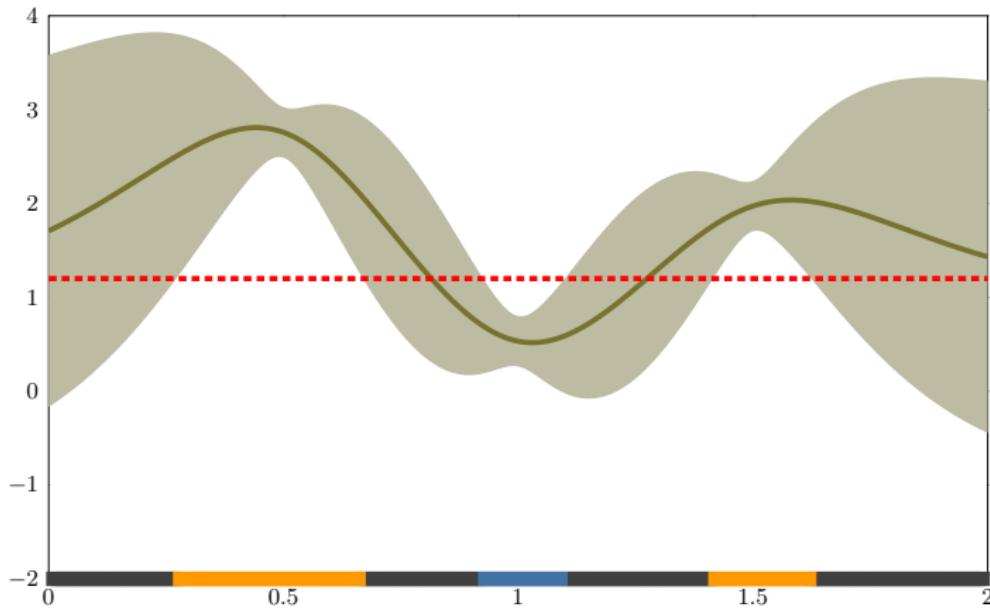
1. How do we **estimate** the function? ✓

2. How do we **classify**?

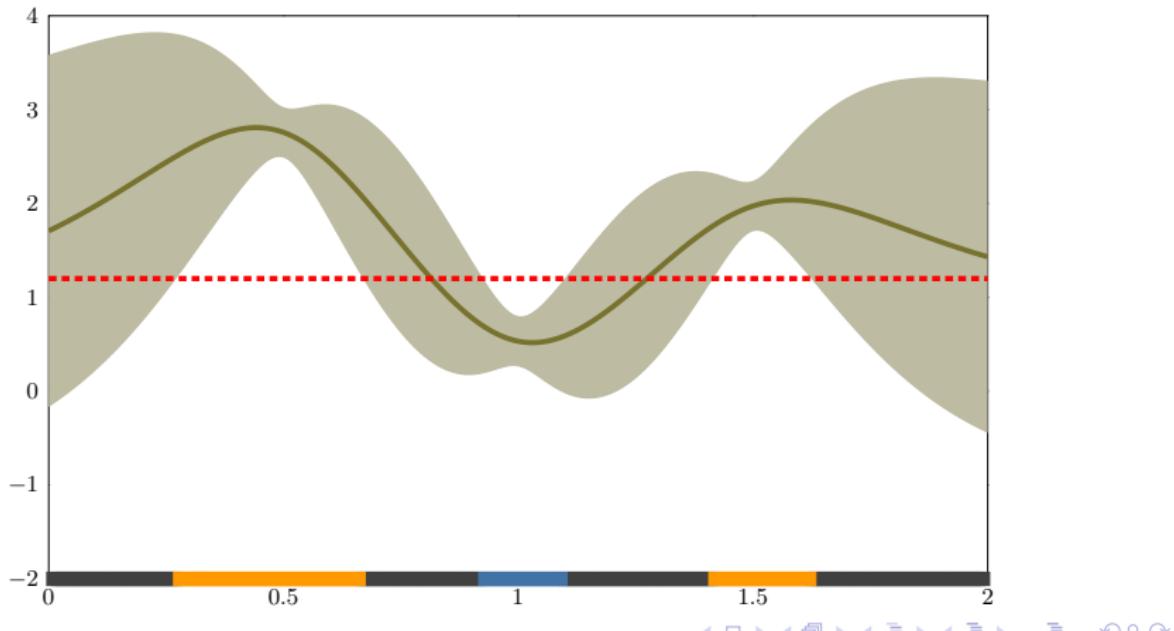


1. How do we **estimate** the function? ✓

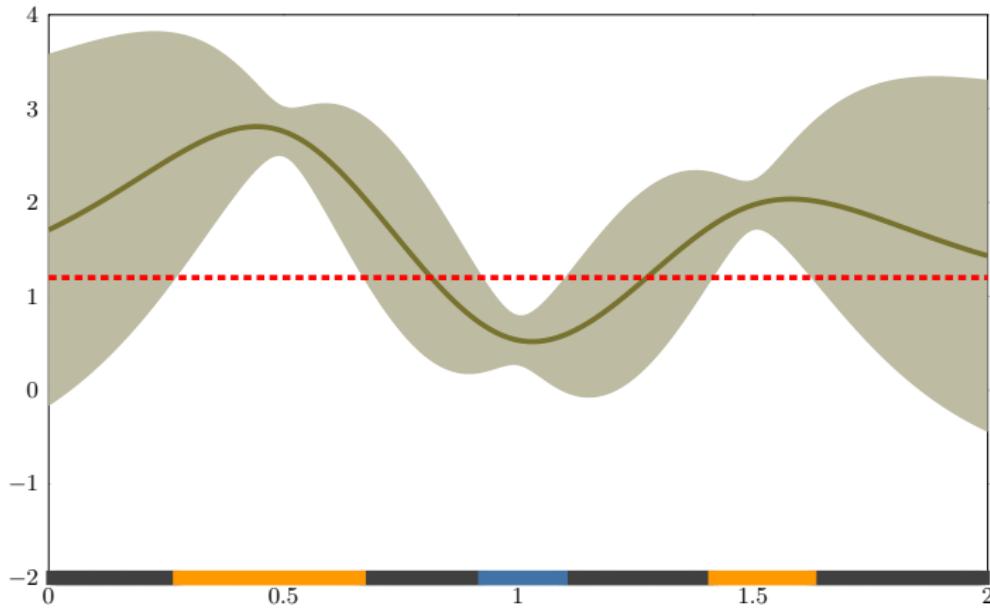
2. How do we **classify**? ✓



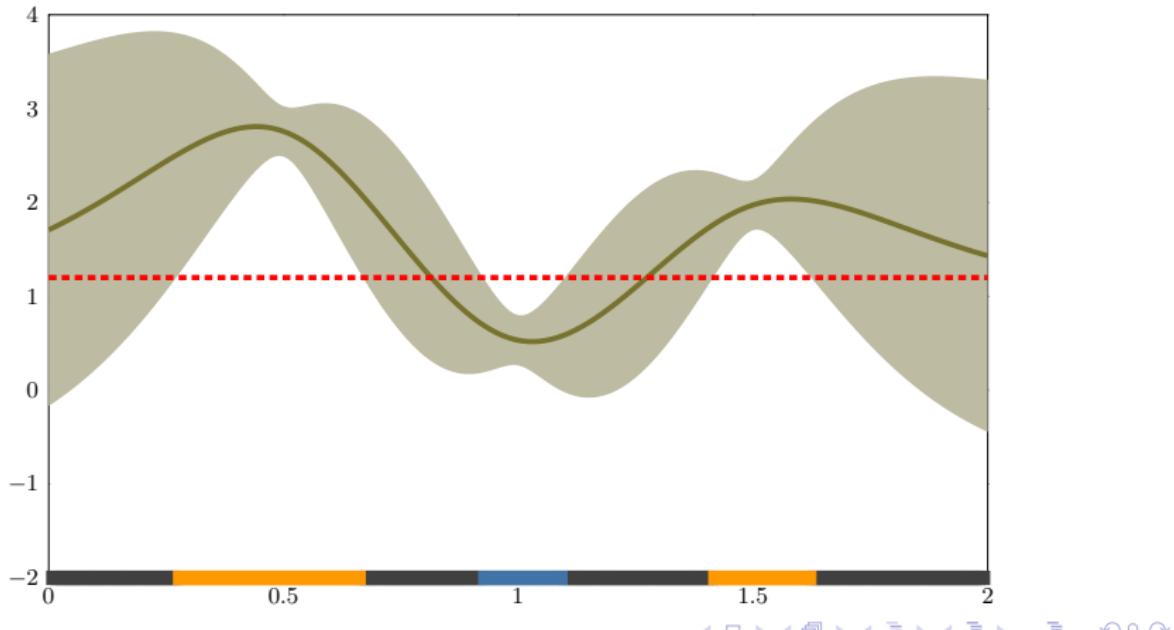
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?



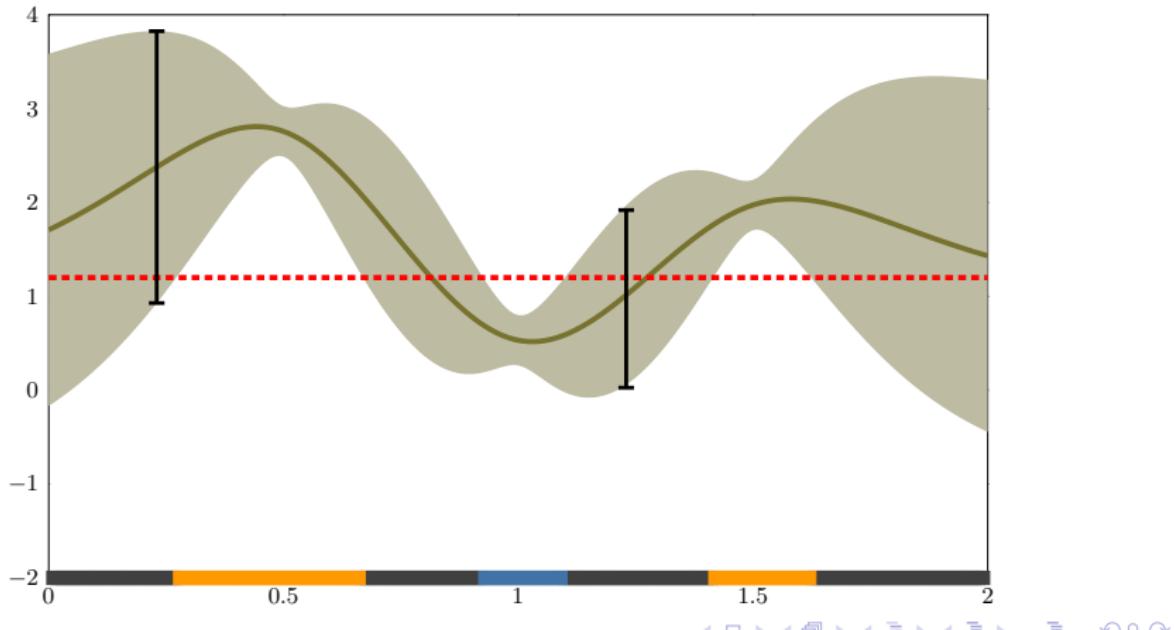
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
  - ▶ Pick among the yet unclassified...



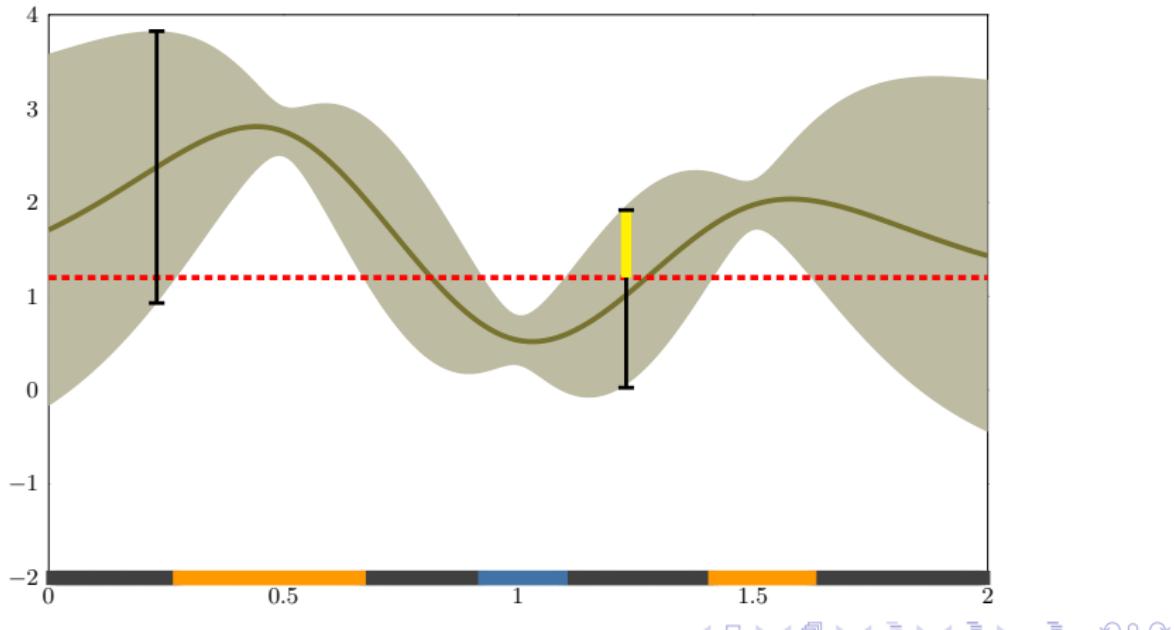
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
  - ▶ Pick among the yet unclassified...
  - ▶ ...the most “ambiguous” point



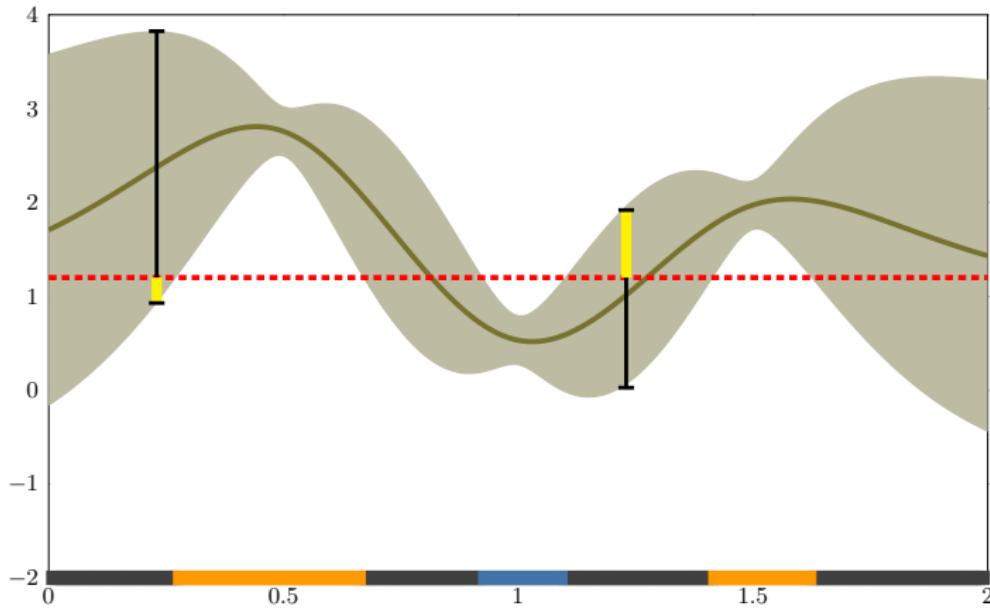
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
  - ▶ Pick among the yet unclassified...
  - ▶ ...the most “ambiguous” point



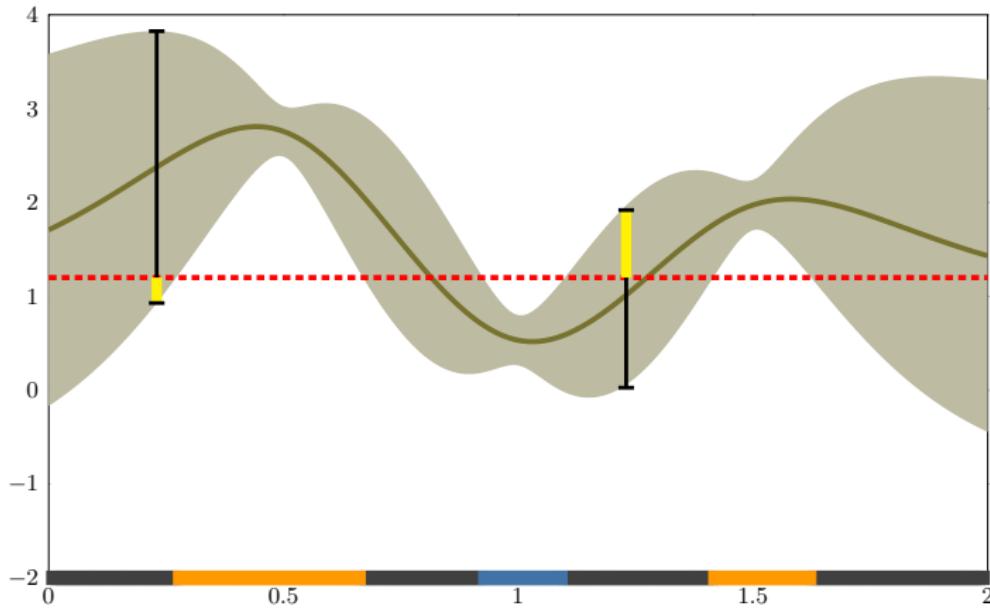
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
  - ▶ Pick among the yet unclassified...
  - ▶ ...the most “ambiguous” point



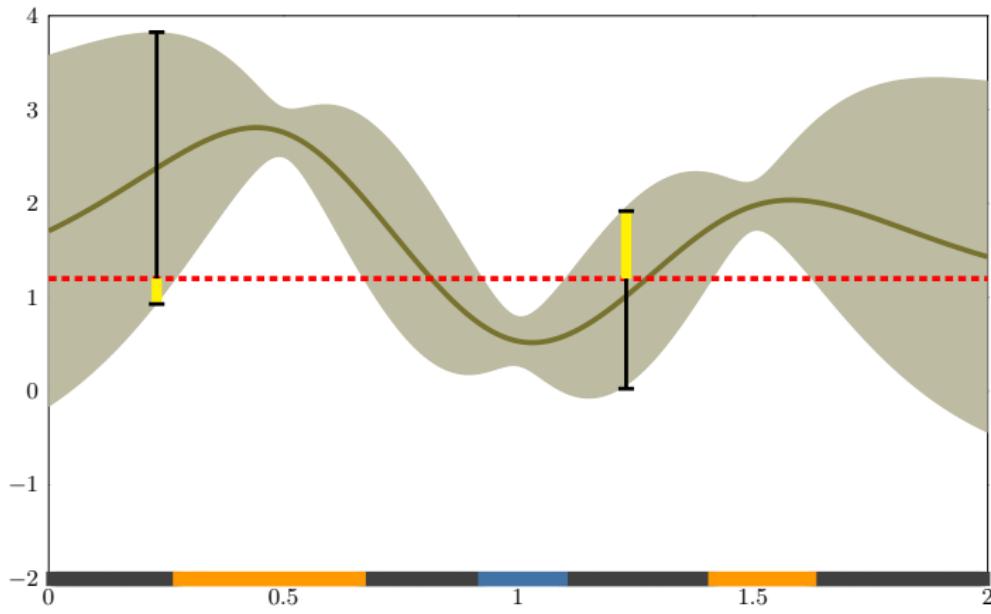
1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
  - ▶ Pick among the yet unclassified...
  - ▶ ...the most “ambiguous” point



1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements?
  - ▶ Pick among the yet unclassified...
  - ▶ ...the most “ambiguous” point (max. variance or random also work)



1. How do we **estimate** the function? ✓
2. How do we **classify**? ✓
3. How do we **select** “informative” measurements? ✓
  - ▶ Pick among the yet unclassified...
  - ▶ ...the most “ambiguous” point (max. variance or random also work)



**Input:** sample set  $D$ , GP prior  $(\mu_0, k, \sigma_0)$ ,  
thr. value  $h$ , accuracy parameter  $\epsilon$   
**Output:** predicted sets  $\hat{H}, \hat{L}$

## The Level Set Estimation (LSE) algorithm

**Input:** sample set  $D$ , GP prior  $(\mu_0, k, \sigma_0)$ ,  
thr. value  $h$ , accuracy parameter  $\epsilon$

**Output:** predicted sets  $\hat{H}, \hat{L}$

$$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$$

$$C_0(x) \leftarrow \mathbb{R}, \text{ for all } x \in D$$

$$t \leftarrow 1$$

## The Level Set Estimation (LSE) algorithm

$$\begin{aligned}\hat{H} &\leftarrow H_{t-1} \\ \hat{L} &\leftarrow L_{t-1}\end{aligned}$$

**Input:** sample set  $D$ , GP prior  $(\mu_0, k, \sigma_0)$ ,  
thr. value  $h$ , accuracy parameter  $\epsilon$

**Output:** predicted sets  $\hat{H}, \hat{L}$

$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$

$C_0(x) \leftarrow \mathbb{R}$ , for all  $x \in D$

$t \leftarrow 1$

**while**  $U_{t-1} \neq \emptyset$  **do**

$H_t \leftarrow H_{t-1}, L_t \leftarrow L_{t-1}, U_t \leftarrow U_{t-1}$

The Level Set Estimation (LSE) algorithm

← loop until all points have been classified

$t \leftarrow t + 1$

**end while**

$\hat{H} \leftarrow H_{t-1}$

$\hat{L} \leftarrow L_{t-1}$

**Input:** sample set  $D$ , GP prior  $(\mu_0, k, \sigma_0)$ ,  
thr. value  $h$ , accuracy parameter  $\epsilon$

**Output:** predicted sets  $\hat{H}, \hat{L}$

$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$

$C_0(x) \leftarrow \mathbb{R}$ , for all  $x \in D$

$t \leftarrow 1$

**while**  $U_{t-1} \neq \emptyset$  **do**

$H_t \leftarrow H_{t-1}, L_t \leftarrow L_{t-1}, U_t \leftarrow U_{t-1}$

**for all**  $x \in U_{t-1}$  **do**

$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$

**if**  $\min(C_t(x)) + \epsilon > h$  **then**

$U_t \leftarrow U_t \setminus \{x\}$

$H_t \leftarrow H_t \cup \{x\}$

**else if**  $\max(C_t(x)) - \epsilon \leq h$  **then**

$U_t \leftarrow U_t \setminus \{x\}$

$L_t \leftarrow L_t \cup \{x\}$

**end if**

**end for**

The Level Set Estimation (LSE) algorithm

← loop until all points have been classified

← classify

$t \leftarrow t + 1$

**end while**

$\hat{H} \leftarrow H_{t-1}$

$\hat{L} \leftarrow L_{t-1}$

**Input:** sample set  $D$ , GP prior  $(\mu_0, k, \sigma_0)$ ,  
thr. value  $h$ , accuracy parameter  $\epsilon$

**Output:** predicted sets  $\hat{H}, \hat{L}$

$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$

$C_0(x) \leftarrow \mathbb{R}$ , for all  $x \in D$

$t \leftarrow 1$

**while**  $U_{t-1} \neq \emptyset$  **do**

$H_t \leftarrow H_{t-1}, L_t \leftarrow L_{t-1}, U_t \leftarrow U_{t-1}$

**for all**  $x \in U_{t-1}$  **do**

$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$

**if**  $\min(C_t(x)) + \epsilon > h$  **then**

$U_t \leftarrow U_t \setminus \{x\}$

$H_t \leftarrow H_t \cup \{x\}$

**else if**  $\max(C_t(x)) - \epsilon \leq h$  **then**

$U_t \leftarrow U_t \setminus \{x\}$

$L_t \leftarrow L_t \cup \{x\}$

**end if**

**end for**

$x_t \leftarrow \operatorname{argmax}\{a_t(x) \mid x \in U_t\}$

$y_t \leftarrow f(x_t) + n_t$

The Level Set Estimation (LSE) algorithm

← loop until all points have been classified

← classify

← select max. ambiguity point

$t \leftarrow t + 1$

**end while**

$\hat{H} \leftarrow H_{t-1}$

$\hat{L} \leftarrow L_{t-1}$

**Input:** sample set  $D$ , GP prior  $(\mu_0, k, \sigma_0)$ ,  
thr. value  $h$ , accuracy parameter  $\epsilon$

**Output:** predicted sets  $\hat{H}, \hat{L}$

$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$

$C_0(x) \leftarrow \mathbb{R}$ , for all  $x \in D$

$t \leftarrow 1$

**while**  $U_{t-1} \neq \emptyset$  **do**

$H_t \leftarrow H_{t-1}, L_t \leftarrow L_{t-1}, U_t \leftarrow U_{t-1}$

**for all**  $x \in U_{t-1}$  **do**

$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$

**if**  $\min(C_t(x)) + \epsilon > h$  **then**

$U_t \leftarrow U_t \setminus \{x\}$

$H_t \leftarrow H_t \cup \{x\}$

**else if**  $\max(C_t(x)) - \epsilon \leq h$  **then**

$U_t \leftarrow U_t \setminus \{x\}$

$L_t \leftarrow L_t \cup \{x\}$

**end if**

**end for**

$x_t \leftarrow \operatorname{argmax}\{a_t(x) \mid x \in U_t\}$

$y_t \leftarrow f(x_t) + n_t$

    Compute  $\mu_t(x)$  and  $\sigma_t(x), \forall x \in U_t$

$t \leftarrow t + 1$

**end while**

$\hat{H} \leftarrow H_{t-1}$

$\hat{L} \leftarrow L_{t-1}$

## The Level Set Estimation (lse) algorithm

← loop until all points have been classified

← classify

← select max. ambiguity point

← update GP estimate

**Input:** sample set  $D$ , GP prior  $(\mu_0, k, \sigma_0)$ ,  
thr. value  $h$ , accuracy parameter  $\epsilon$

**Output:** predicted sets  $\hat{H}, \hat{L}$

$$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$$

$$C_0(x) \leftarrow \mathbb{R}, \text{ for all } x \in D$$

$$t \leftarrow 1$$

**while**  $U_{t-1} \neq \emptyset$  **do**

$$H_t \leftarrow H_{t-1}, L_t \leftarrow L_{t-1}, U_t \leftarrow U_{t-1}$$

**for all**  $x \in U_{t-1}$  **do**

$$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$$

**if**  $\min(C_t(x)) + \epsilon > h$  **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$H_t \leftarrow H_t \cup \{x\}$$

**else if**  $\max(C_t(x)) - \epsilon \leq h$  **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$L_t \leftarrow L_t \cup \{x\}$$

**end if**

**end for**

$$x_t \leftarrow \operatorname{argmax}\{a_t(x) \mid x \in U_t\}$$

$$y_t \leftarrow f(x_t) + n_t$$

Compute  $\mu_t(x)$  and  $\sigma_t(x)$ ,  $\forall x \in U_t$

$$t \leftarrow t + 1$$

**end while**

$$\hat{H} \leftarrow H_{t-1}$$

$$\hat{L} \leftarrow L_{t-1}$$

## The Level Set Estimation (lse) algorithm

- ▶ Monotonicity of
  - 1. confidence intervals
  - 2. classification

**Input:** sample set  $D$ , GP prior  $(\mu_0, k, \sigma_0)$ ,  
thr. value  $h$ , accuracy parameter  $\epsilon$

**Output:** predicted sets  $\hat{H}, \hat{L}$

$$H_0 \leftarrow \emptyset, L_0 \leftarrow \emptyset, U_0 \leftarrow D$$

$$C_0(x) \leftarrow \mathbb{R}, \text{ for all } x \in D$$

$$t \leftarrow 1$$

**while**  $U_{t-1} \neq \emptyset$  **do**

$$H_t \leftarrow H_{t-1}, L_t \leftarrow L_{t-1}, U_t \leftarrow U_{t-1}$$

**for all**  $x \in U_{t-1}$  **do**

$$C_t(x) \leftarrow C_{t-1}(x) \cap Q_t(x)$$

**if**  $\min(C_t(x)) + \epsilon > h$  **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$H_t \leftarrow H_t \cup \{x\}$$

**else if**  $\max(C_t(x)) - \epsilon \leq h$  **then**

$$U_t \leftarrow U_t \setminus \{x\}$$

$$L_t \leftarrow L_t \cup \{x\}$$

**end if**

**end for**

$$x_t \leftarrow \operatorname{argmax}\{a_t(x) \mid x \in U_t\}$$

$$y_t \leftarrow f(x_t) + n_t$$

Compute  $\mu_t(x)$  and  $\sigma_t(x)$ ,  $\forall x \in U_t$

$$t \leftarrow t + 1$$

**end while**

$$\hat{H} \leftarrow H_{t-1}$$

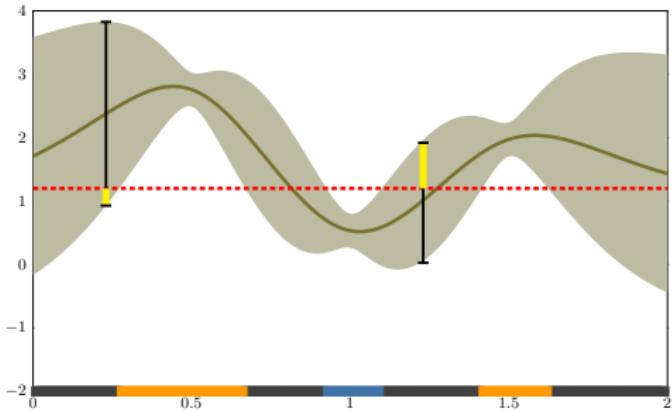
$$\hat{L} \leftarrow L_{t-1}$$

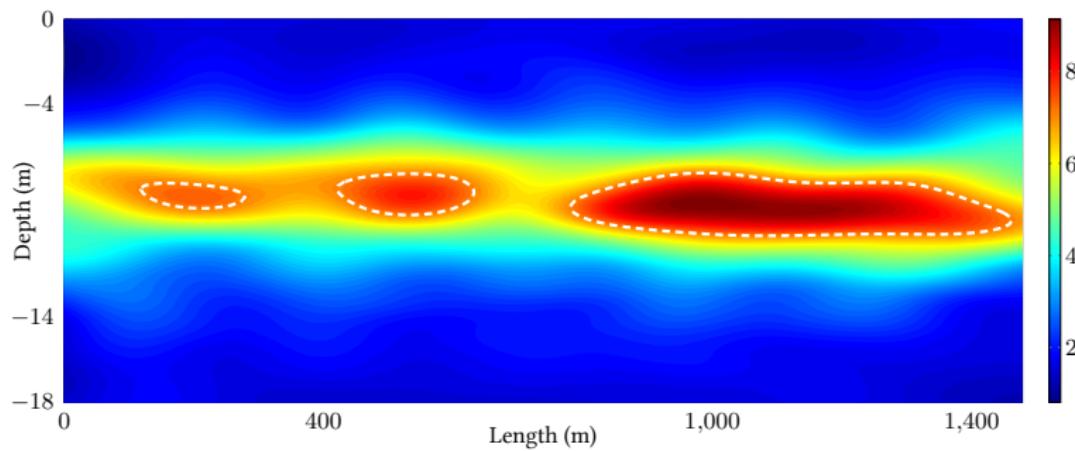
## The Level Set Estimation (LSE) algorithm

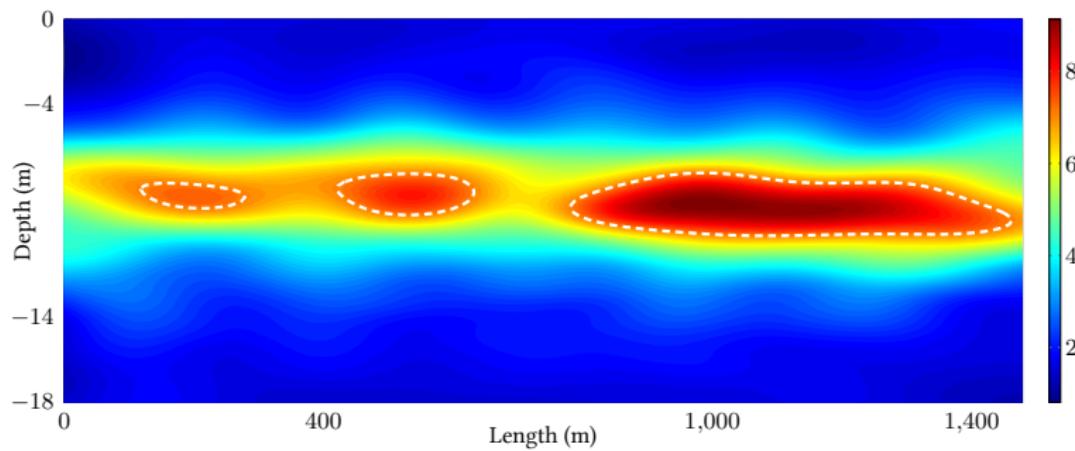
### ► Monotonicity of

1. confidence intervals
2. classification

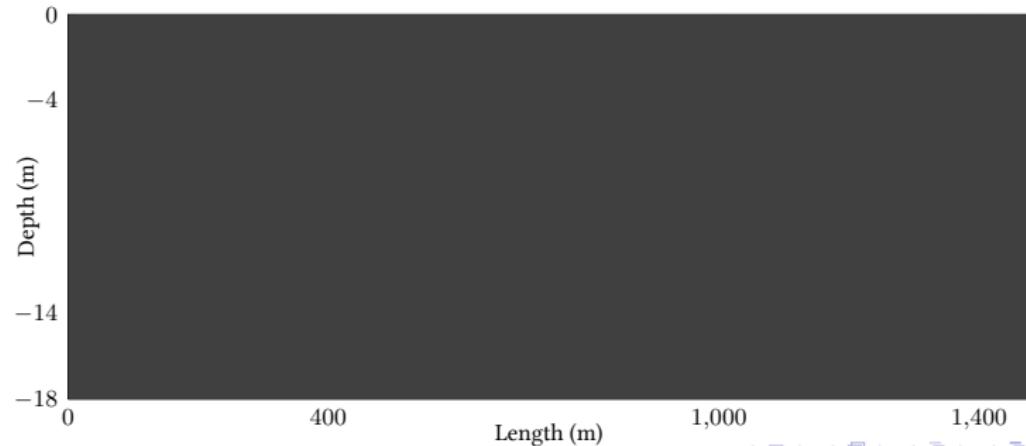
### ► Relaxed classification rules by an accuracy parameter $\epsilon$ (trades off sampling cost for accuracy)

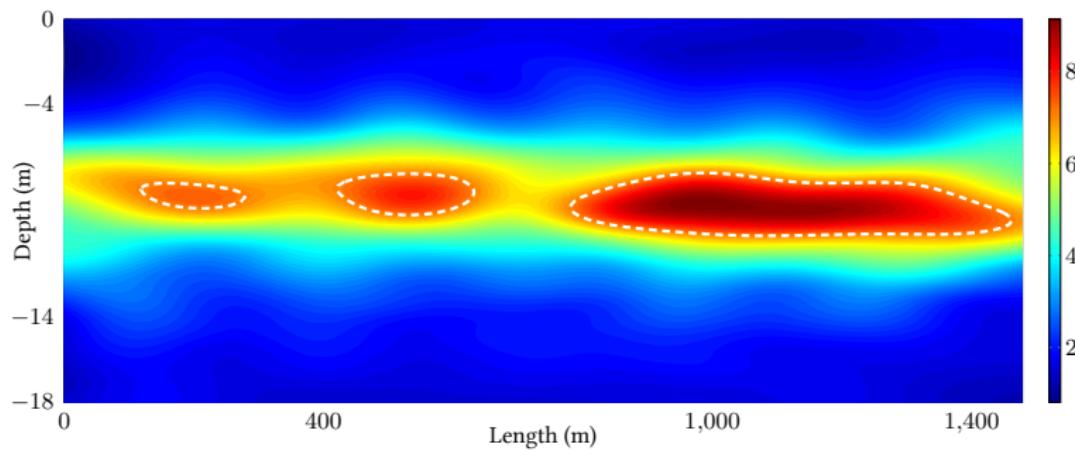




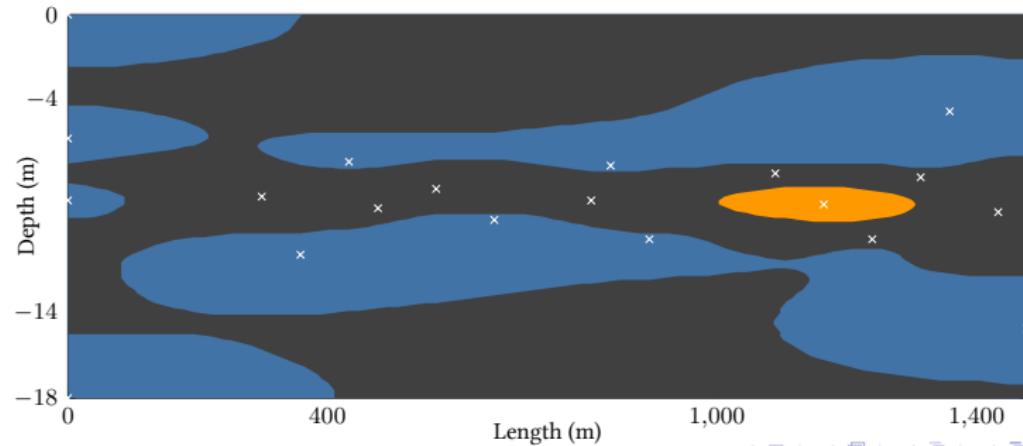


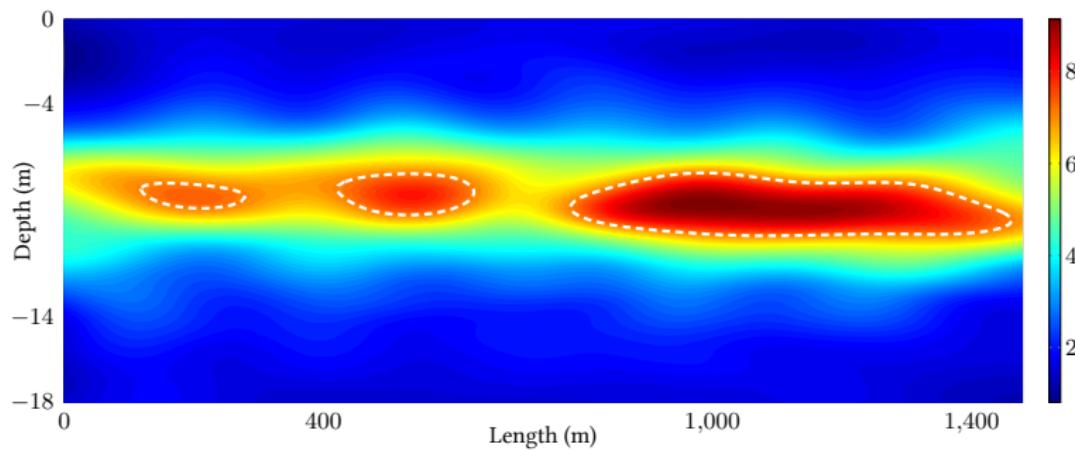
$t = 0$



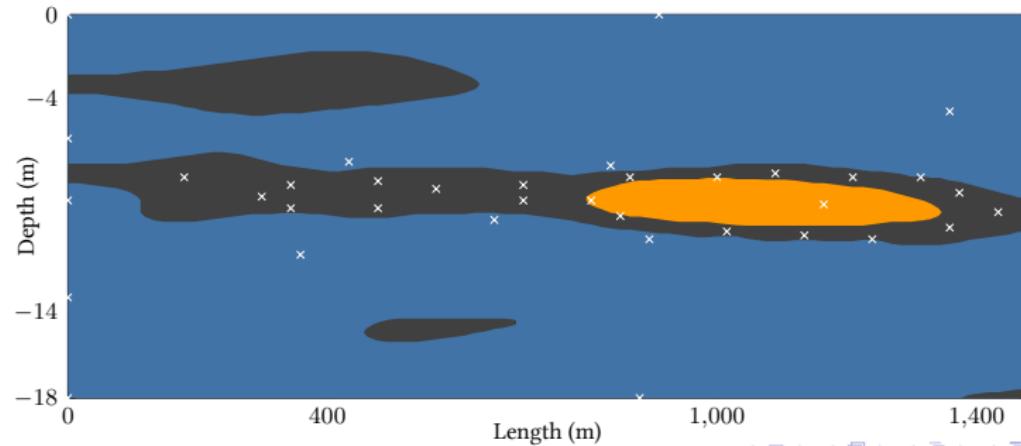


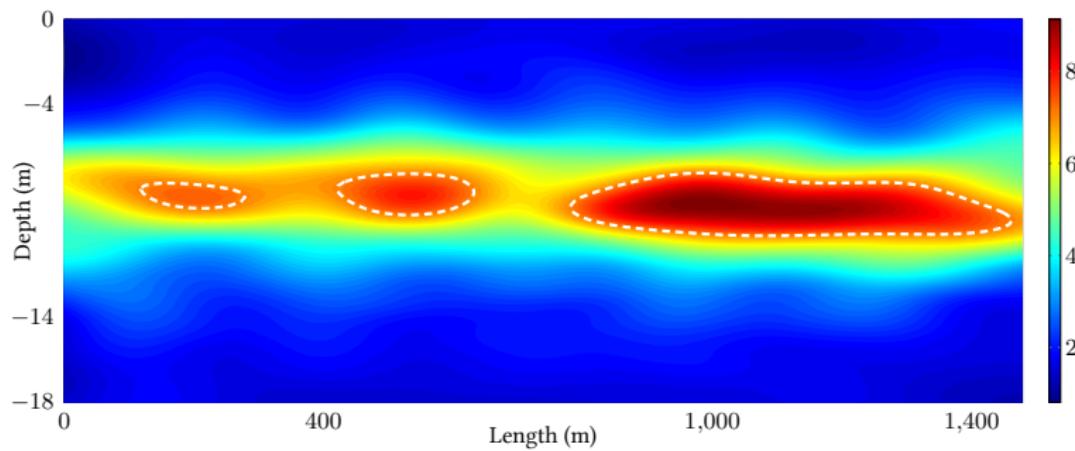
$t = 20$



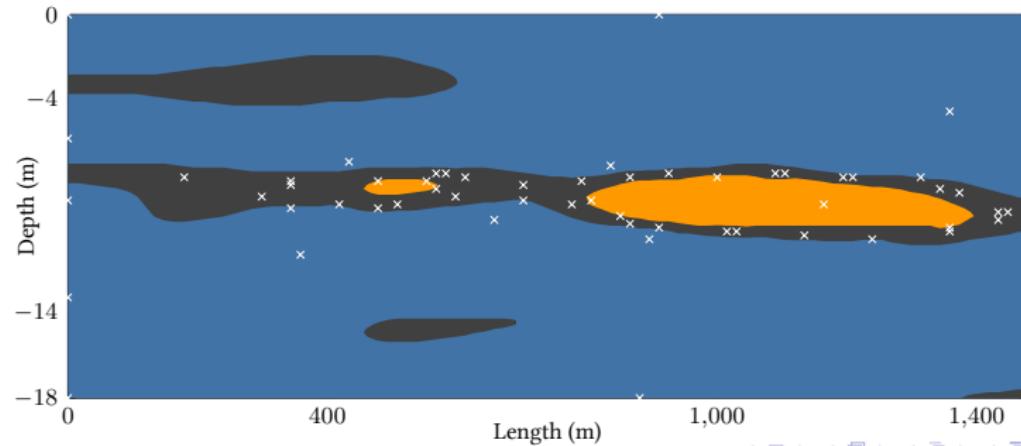


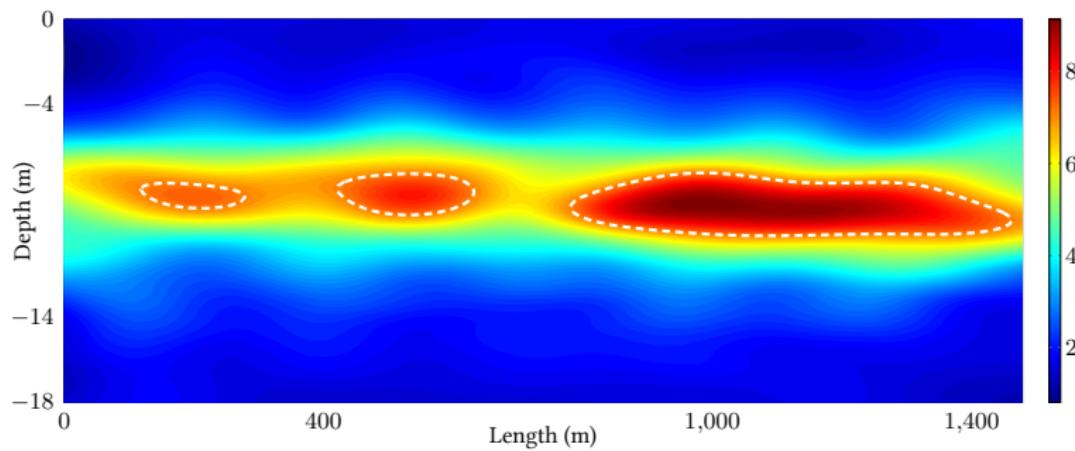
$t = 40$



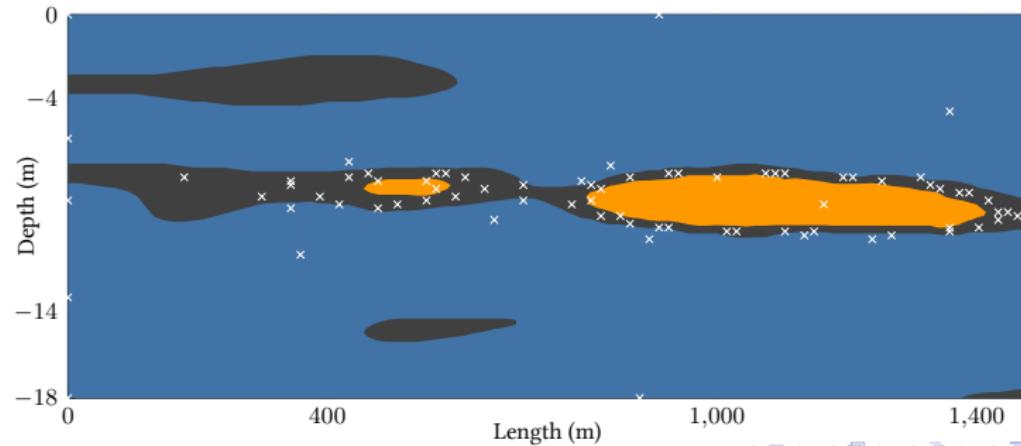


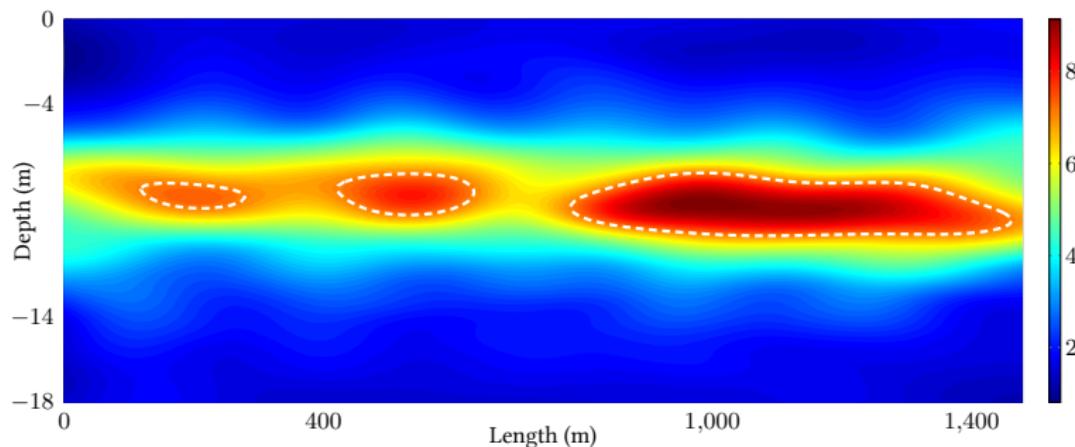
$t = 60$



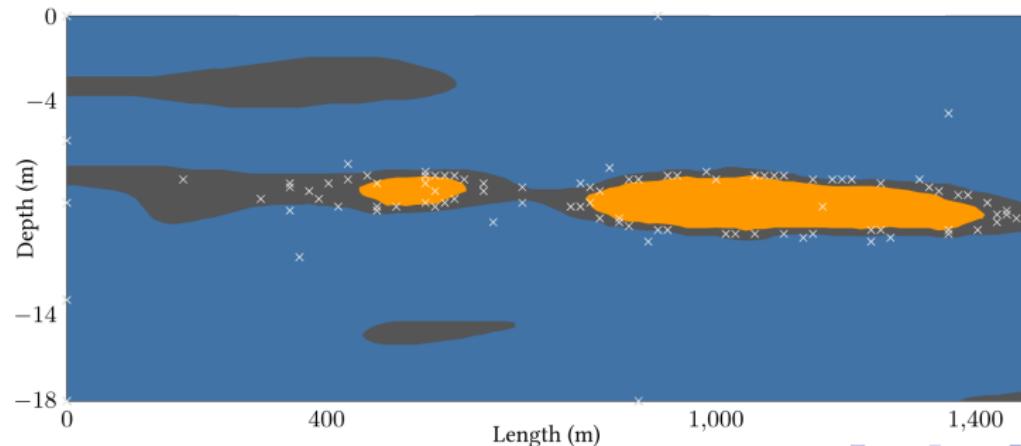


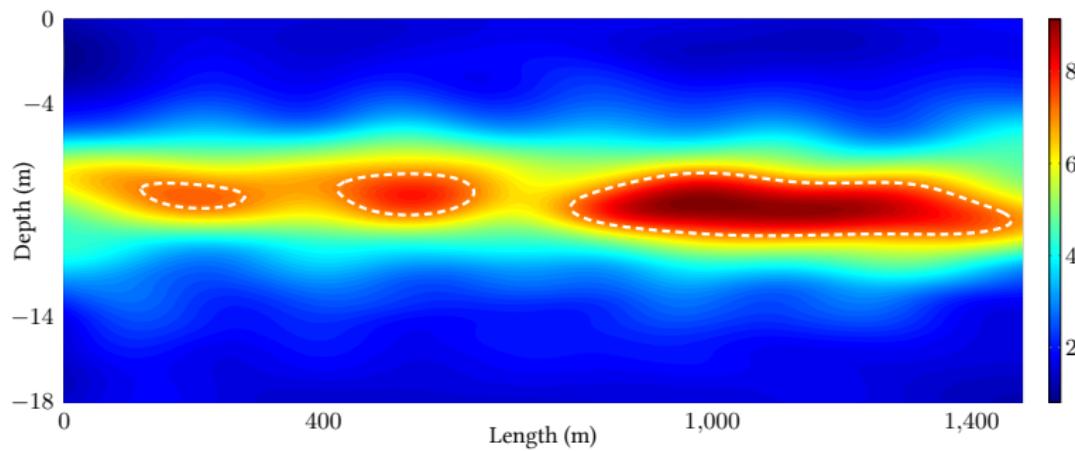
$t = 80$



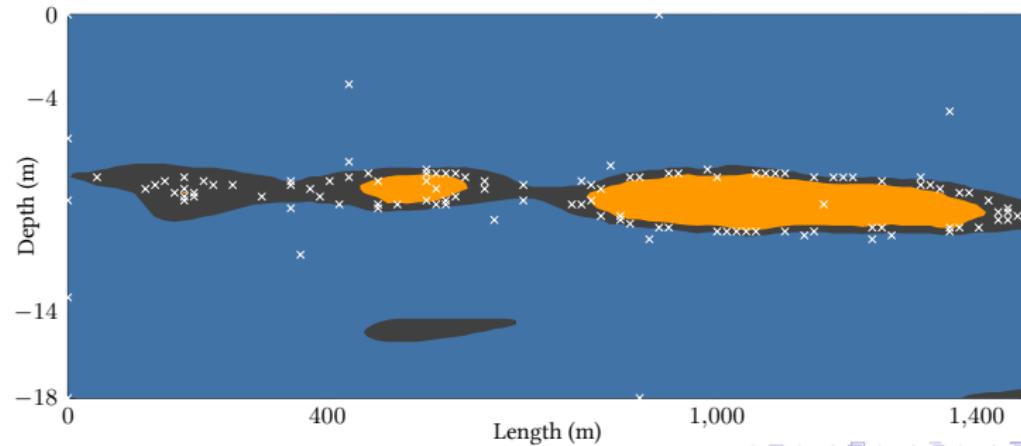


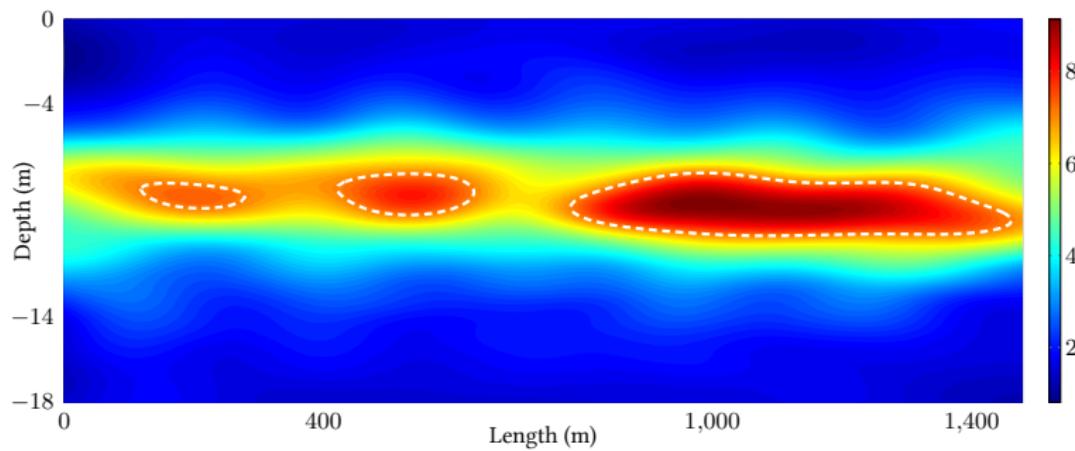
$t = 100$



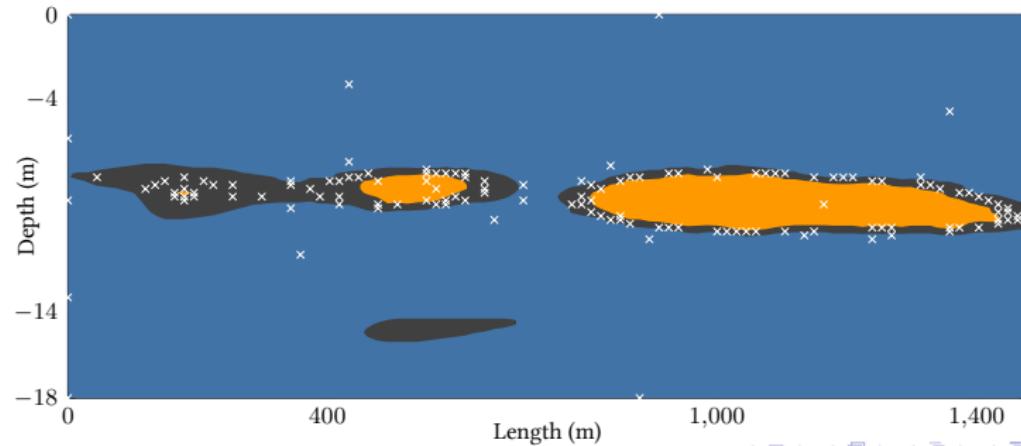


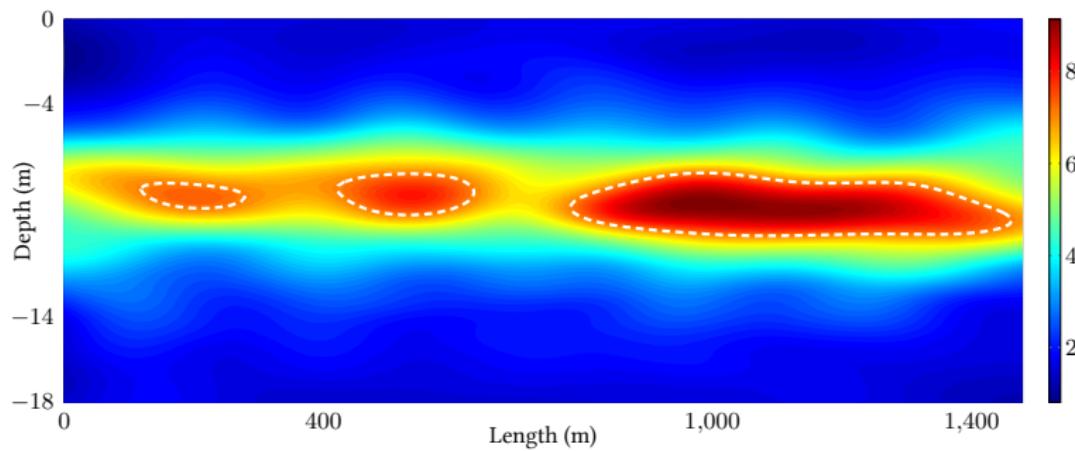
$t = 120$



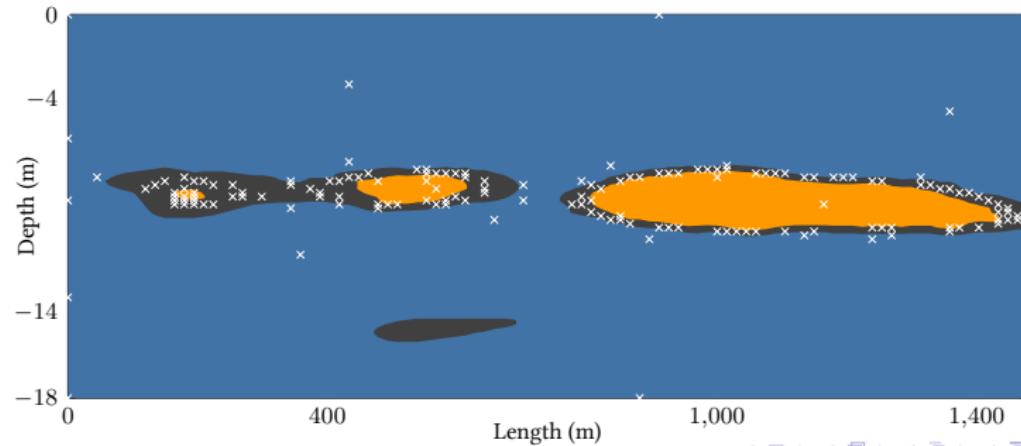


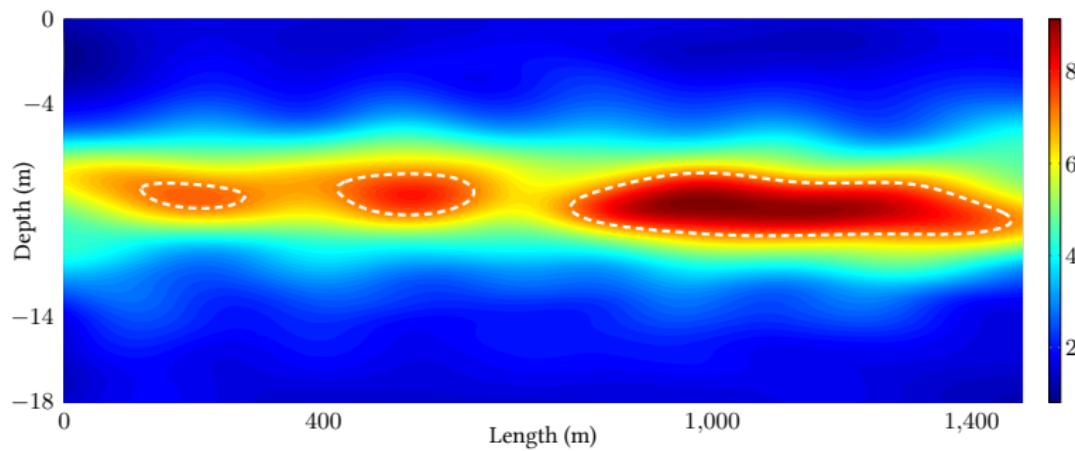
$t = 140$



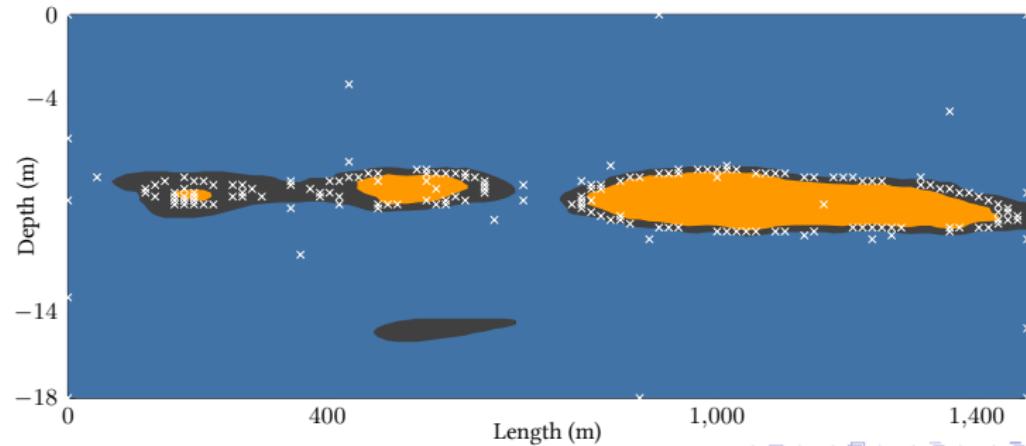


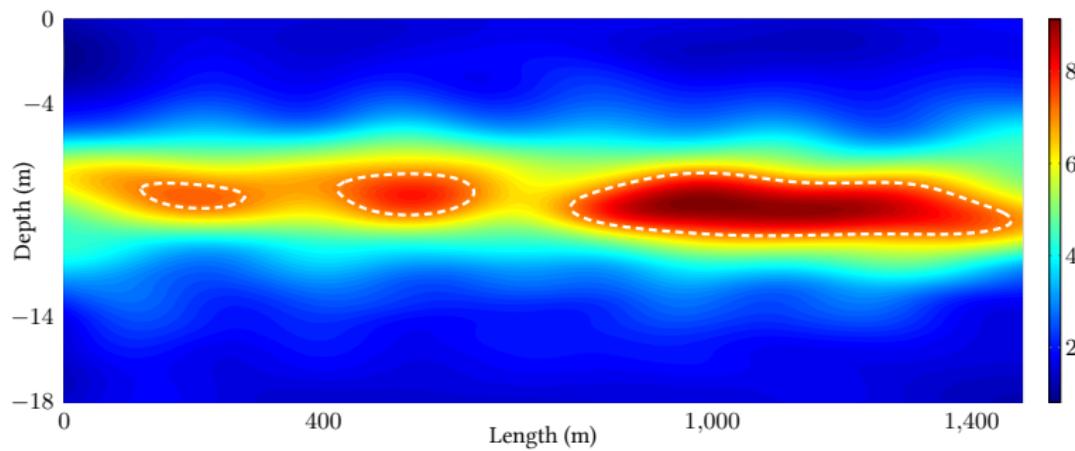
$t = 160$



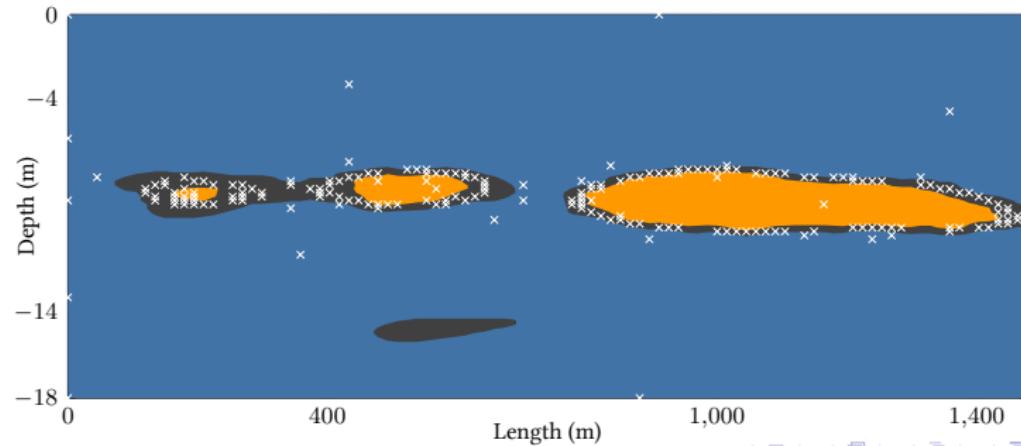


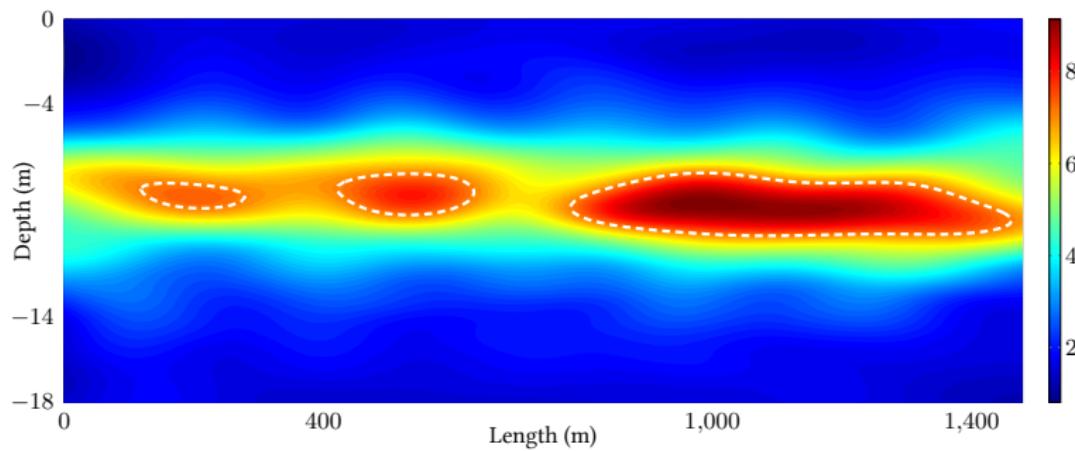
$t = 180$



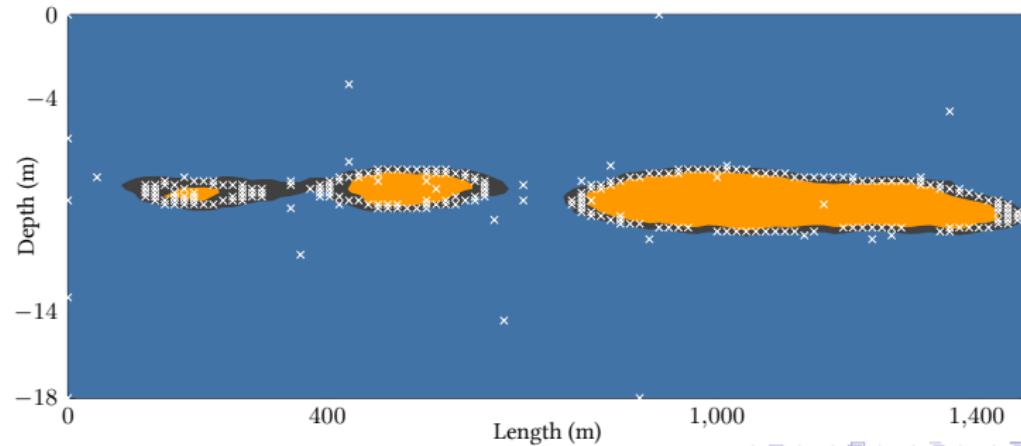


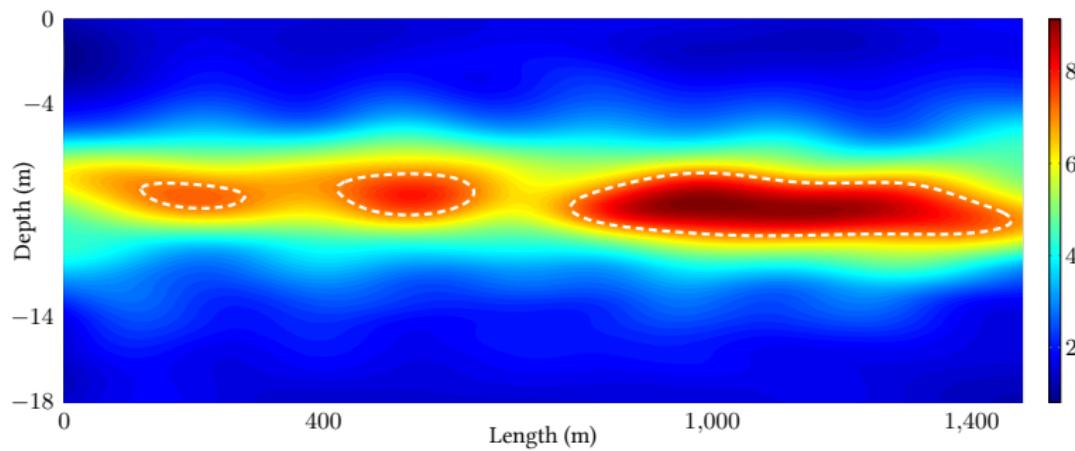
$t = 200$



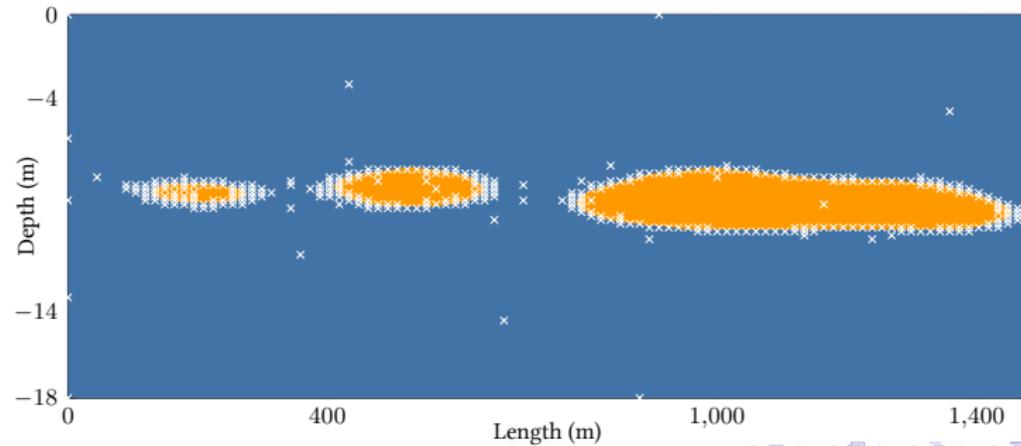


$t = 260$





$t = 354$



## Theorem (Convergence of LSE)

For any  $h \in \mathbb{R}$ ,  $\delta \in (0, 1)$ , and  $\epsilon > 0$ , if  $\beta_t = 2 \log(|D| \pi^2 t^2 / (6\delta))$ , LSE terminates after at most  $T$  iterations, where  $T$  is the smallest positive integer satisfying

$$\frac{T}{\beta_T \gamma_T} \geq \frac{C_1}{4\epsilon^2},$$

where  $C_1 = 8/\log(1 + \sigma^{-2})$ .

Furthermore, with probability at least  $1 - \delta$ , the algorithm returns an  $\epsilon$ -accurate solution, that is

$$\Pr \left\{ \max_{\mathbf{x} \in D} \ell_h(\mathbf{x}) \leq \epsilon \right\} \geq 1 - \delta.$$

## Theorem (Simplified)

*If we choose  $\beta_t$  large enough, then:*

## Theorem (Simplified)

*If we choose  $\beta_t$  large enough, then:*

- ▶ LSE terminates after a number of iterations  $T$

## Theorem (Simplified)

If we choose  $\beta_t$  large enough, then:

- ▶ LSE terminates after a number of iterations  $T$ 
  1. smoother kernel  $\Rightarrow T \downarrow$

## Theorem (Simplified)

If we choose  $\beta_t$  large enough, then:

- ▶ LSE terminates after a number of iterations  $T$ 
  1. smoother kernel  $\Rightarrow T \downarrow$
  2.  $\sigma \uparrow \Rightarrow T \uparrow$

## Theorem (Simplified)

If we choose  $\beta_t$  large enough, then:

- ▶ LSE terminates after a number of iterations  $T$ 
  1. smoother kernel  $\Rightarrow T \downarrow$
  2.  $\sigma \uparrow \Rightarrow T \uparrow$
  3.  $\epsilon \uparrow \Rightarrow T \downarrow$

## Theorem (Simplified)

If we choose  $\beta_t$  large enough, then:

- ▶ LSE terminates after a number of iterations  $T$ 
  1. smoother kernel  $\Rightarrow T \downarrow$
  2.  $\sigma \uparrow \Rightarrow T \uparrow$
  3.  $\epsilon \uparrow \Rightarrow T \downarrow$
- ▶ The solution returned is  $\epsilon$ -accurate with high probability

## Theorem (Simplified)

If we choose  $\beta_t$  large enough, then:

- ▶ LSE terminates after a number of iterations  $T$ 
  1. smoother kernel  $\Rightarrow T \downarrow$
  2.  $\sigma \uparrow \Rightarrow T \uparrow$
  3.  $\epsilon \uparrow \Rightarrow T \downarrow$
- ▶ The solution returned is  $\epsilon$ -accurate with high probability

# Experiments

## 1. LSE

## Experiments

1. LSE
2. Maximum variance sampling:

$$\mathbf{x}_t = \operatorname{argmax}_{\mathbf{x} \in D} \sigma_{t-1}(\mathbf{x})$$

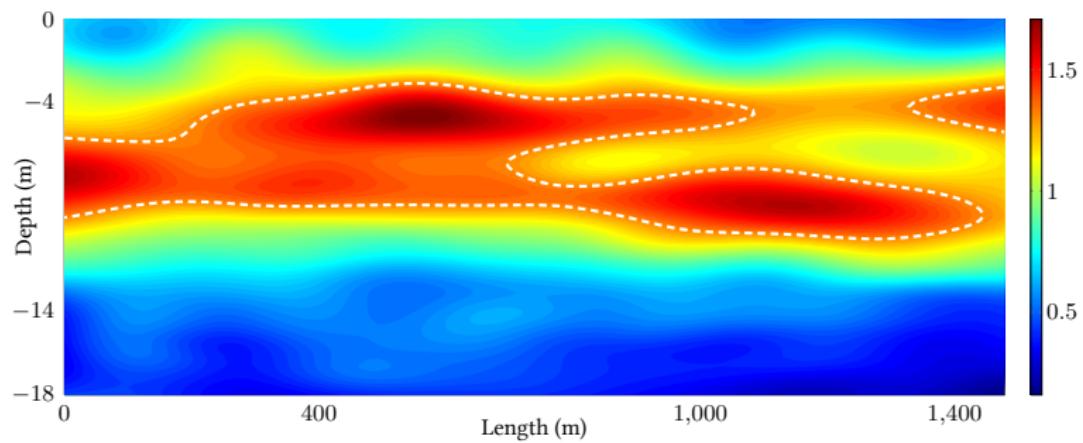
## Experiments

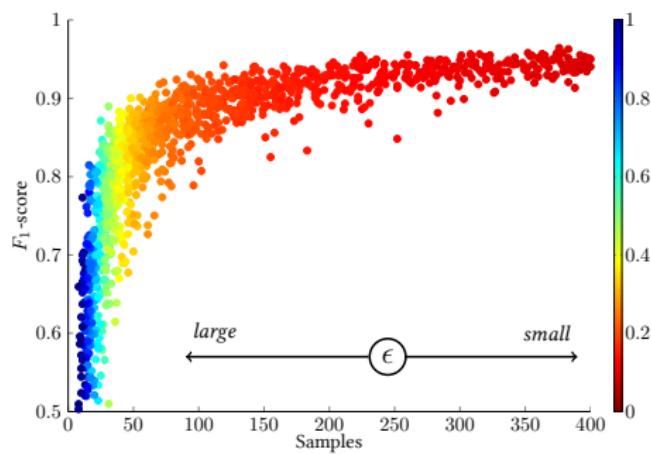
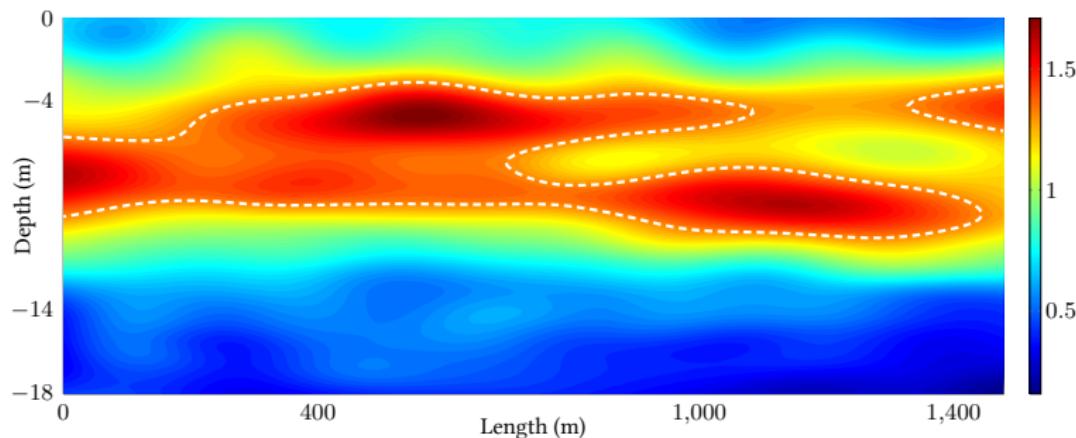
1. LSE
2. Maximum variance sampling:

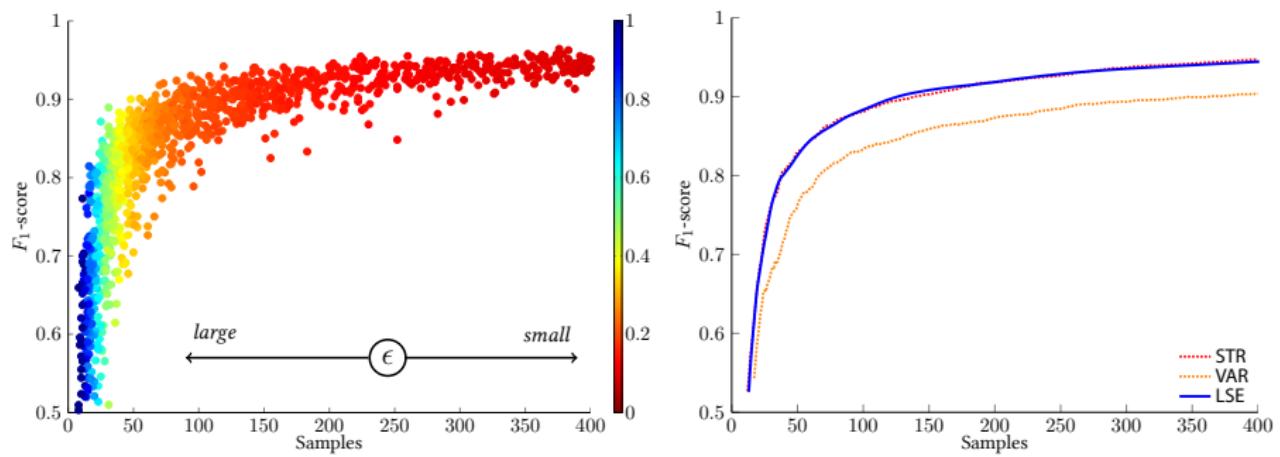
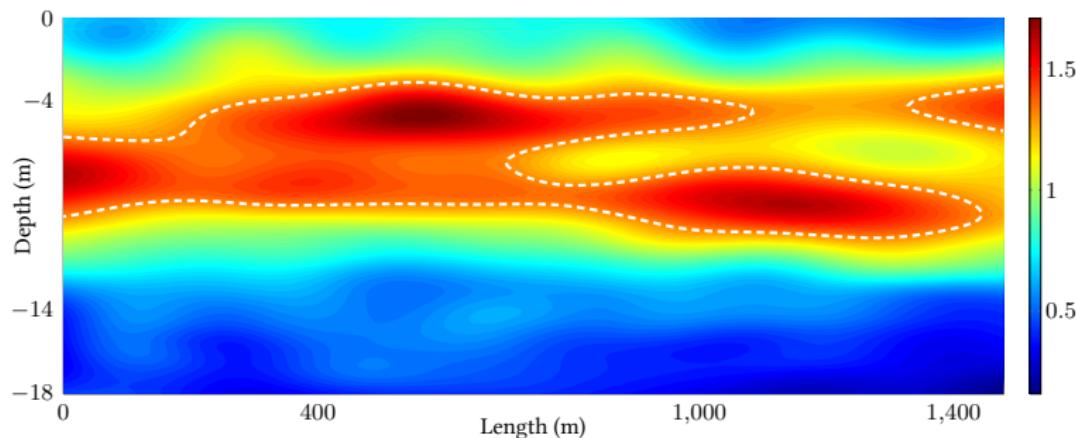
$$\mathbf{x}_t = \operatorname{argmax}_{\mathbf{x} \in D} \sigma_{t-1}(\mathbf{x})$$

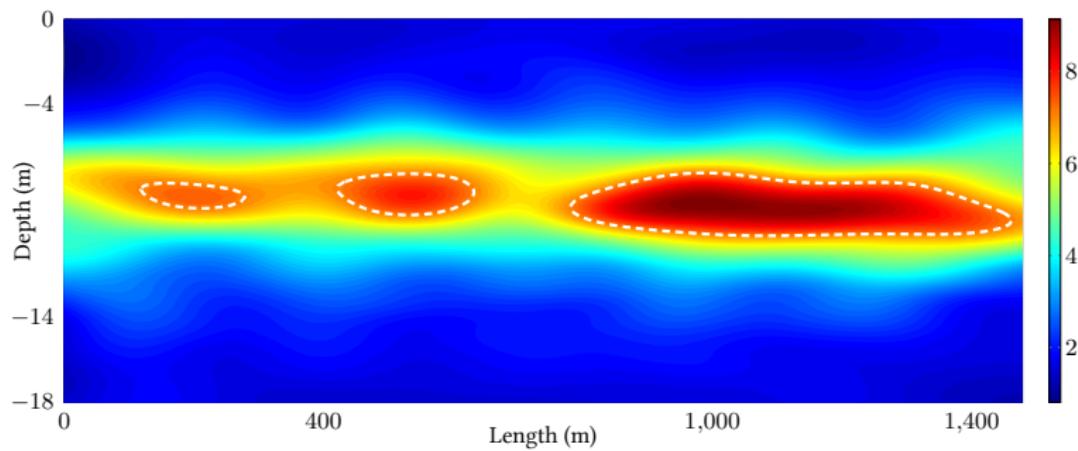
3. State of the art “straddle” heuristic (Bryan *et al.*, 2005):

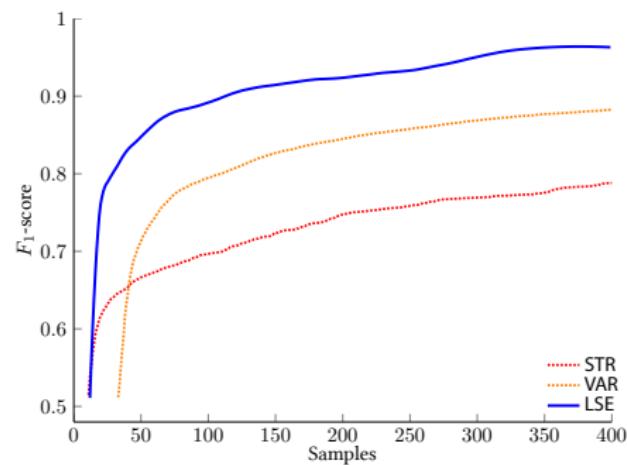
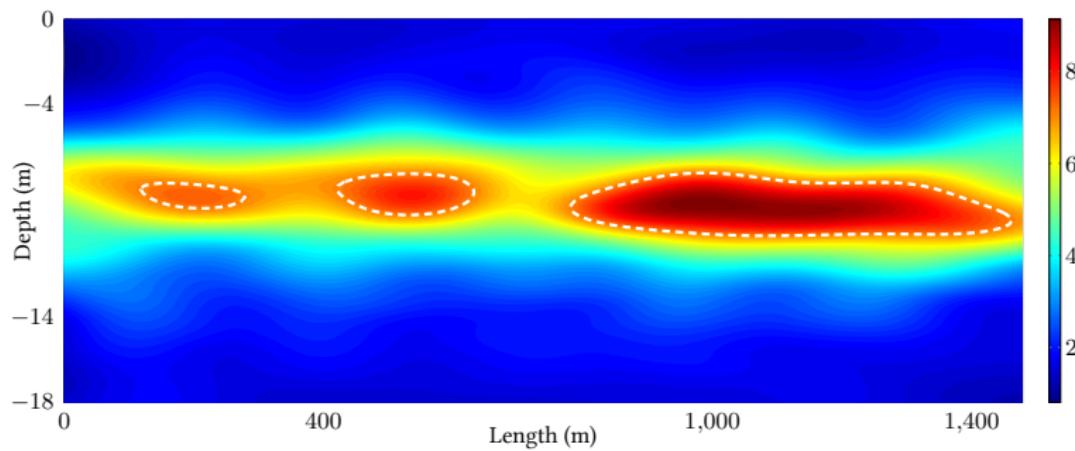
$$\mathbf{x}_t \approx \operatorname{argmax}_{\mathbf{x} \in D} a_{t-1}(\mathbf{x}) \quad (\text{for } \beta_t^{1/2} = 1.96)$$











- ▶ What if we don't know the threshold level  $h$ ?

- ▶ What if we don't know the threshold level  $h$ ?
- ▶ Implicitly define  $h = \omega \max_{x \in D} f(x)$ ,  $0 < \omega < 1$

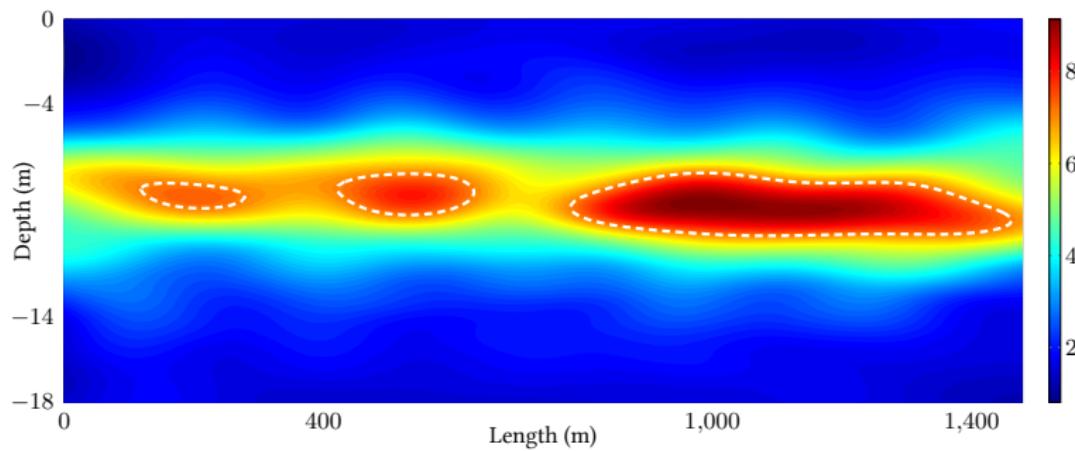
- ▶ What if we don't know the threshold level  $h$ ?
- ▶ Implicitly define  $h = \omega \max_{x \in D} f(x)$ ,  $0 < \omega < 1$
- ▶ No existing algorithms for this problem (to our knowledge)

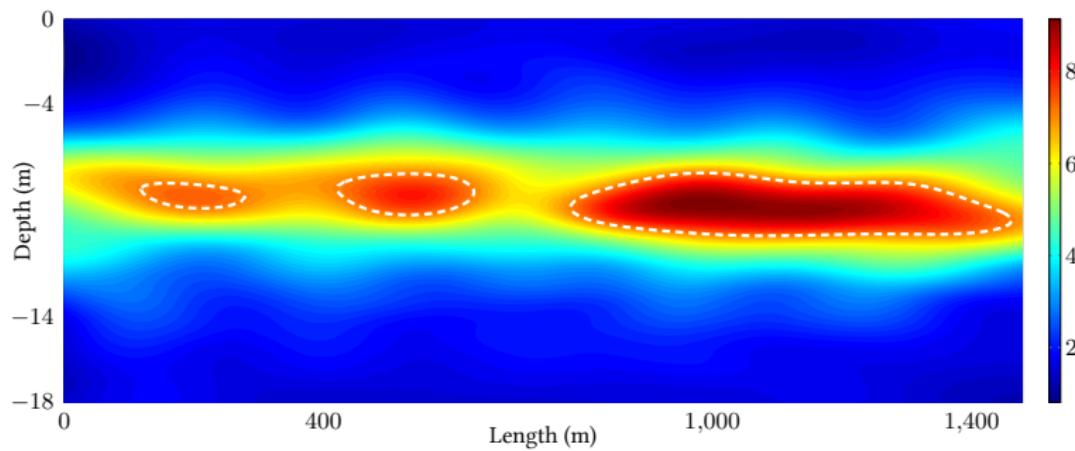
- ▶ What if we don't know the threshold level  $h$ ?
- ▶ Implicitly define  $h = \omega \max_{x \in D} f(x)$ ,  $0 < \omega < 1$
- ▶ No existing algorithms for this problem (to our knowledge)
- ▶ We extend LSE (and its theory!) to this setting

- ▶ What if we don't know the threshold level  $h$ ?
- ▶ Implicitly define  $h = \omega \max_{x \in D} f(x)$ ,  $0 < \omega < 1$
- ▶ No existing algorithms for this problem (to our knowledge)
- ▶ We extend LSE (and its theory!) to this setting
- ▶ LSE<sub>imp</sub> algorithm:

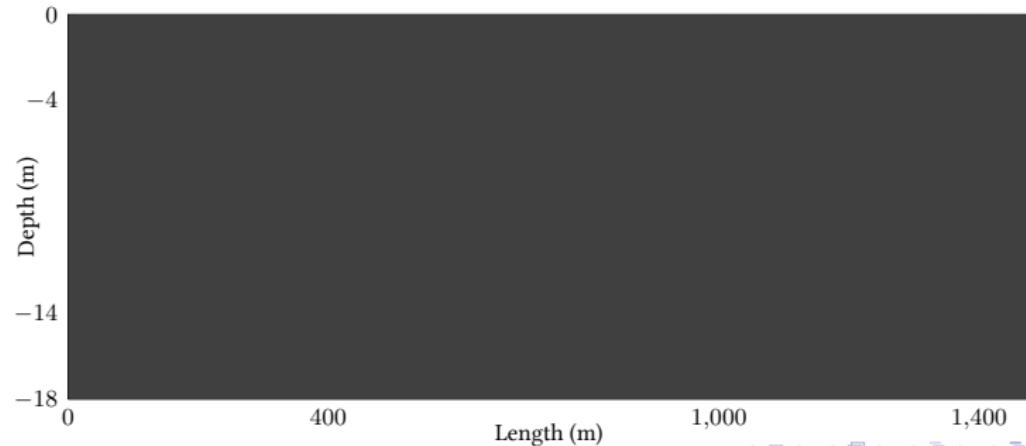
- ▶ What if we don't know the threshold level  $h$ ?
- ▶ Implicitly define  $h = \omega \max_{x \in D} f(x)$ ,  $0 < \omega < 1$
- ▶ No existing algorithms for this problem (to our knowledge)
- ▶ We extend LSE (and its theory!) to this setting
- ▶ LSE<sub>imp</sub> algorithm:
  - ▶ Need to accurately estimate the maximum → modified selection rule

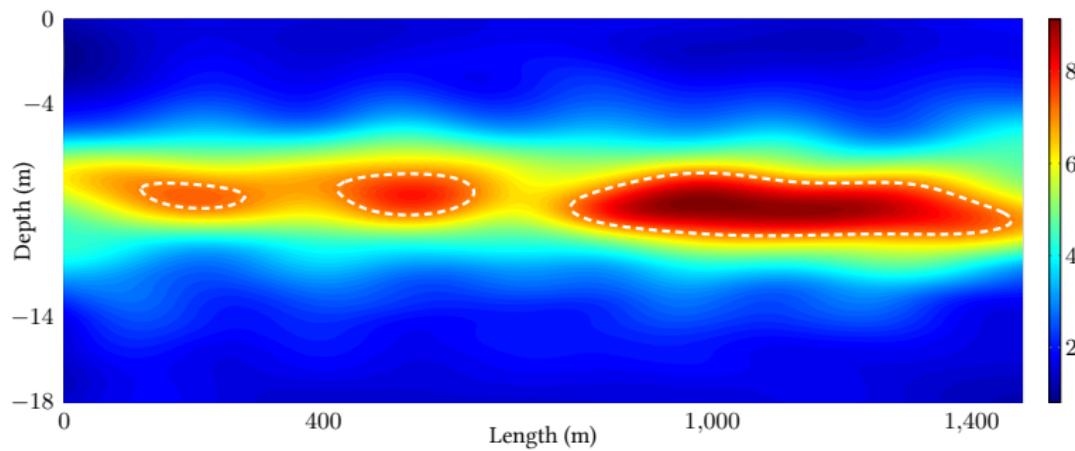
- ▶ What if we don't know the threshold level  $h$ ?
- ▶ Implicitly define  $h = \omega \max_{x \in D} f(x)$ ,  $0 < \omega < 1$
- ▶ No existing algorithms for this problem (to our knowledge)
- ▶ We extend LSE (and its theory!) to this setting
- ▶ LSE<sub>imp</sub> algorithm:
  - ▶ Need to accurately estimate the maximum → modified selection rule
  - ▶  $h$  is now an estimated quantity → modified classification rules



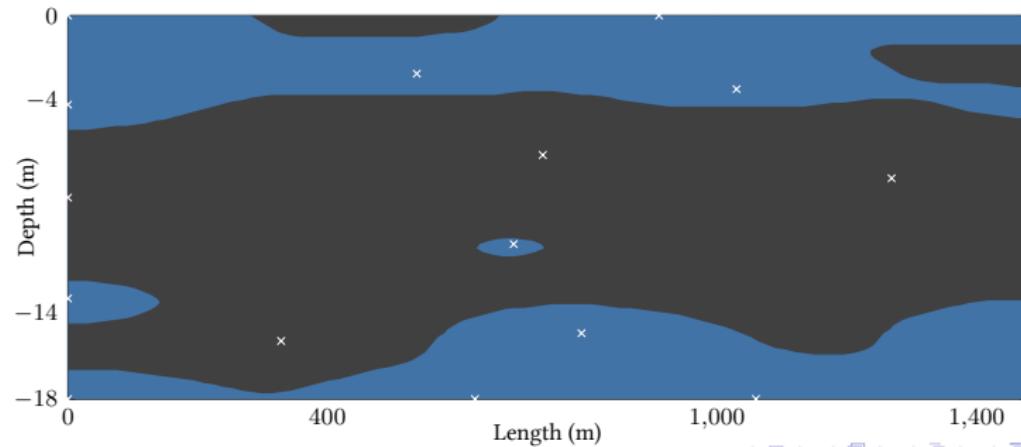


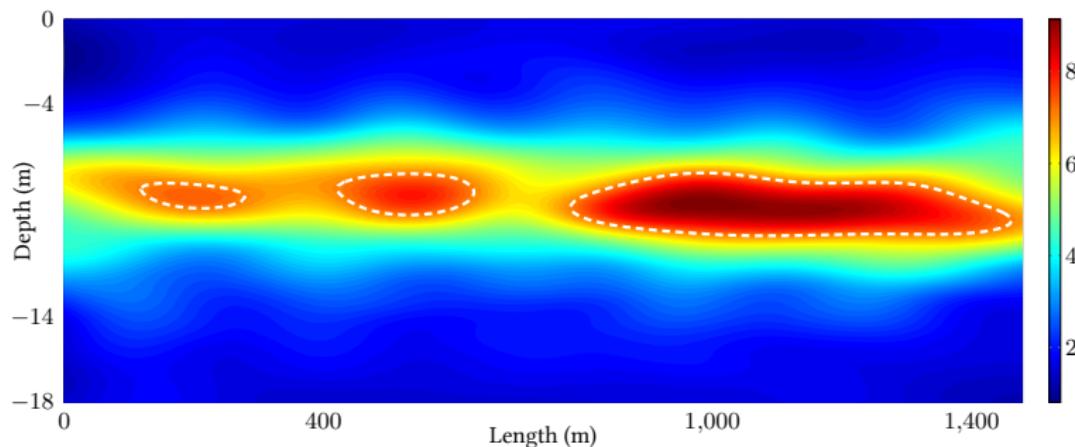
$t = 0$



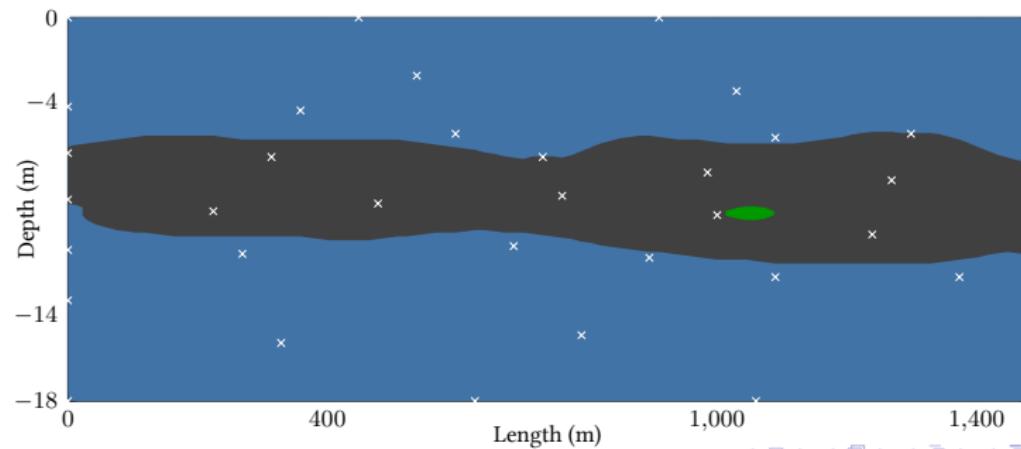


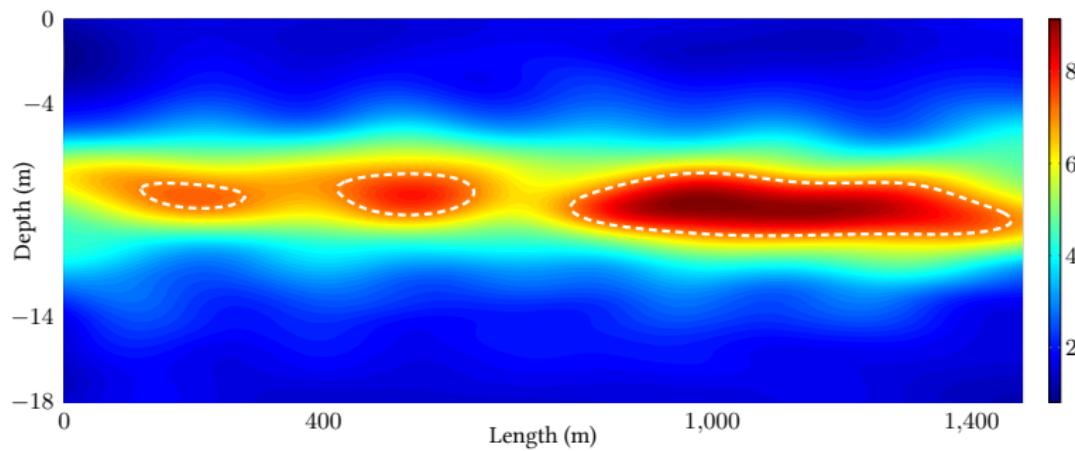
$t = 20$



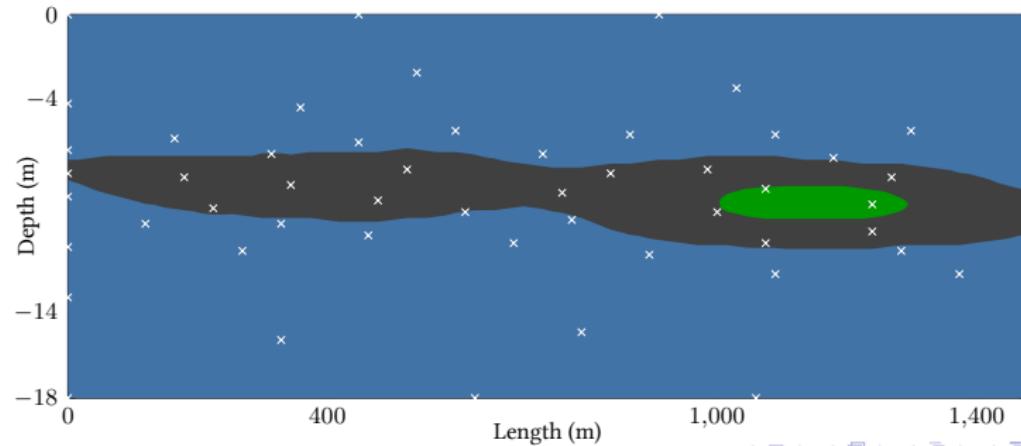


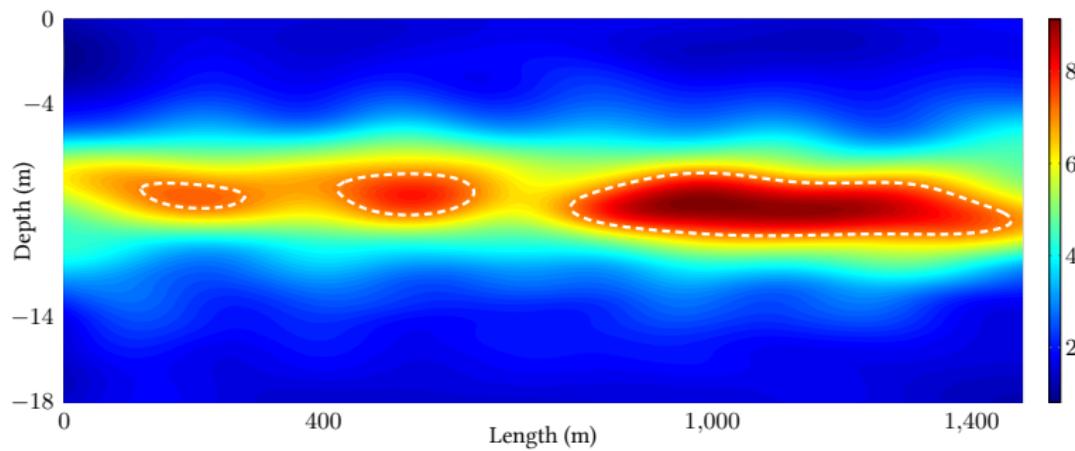
$t = 40$



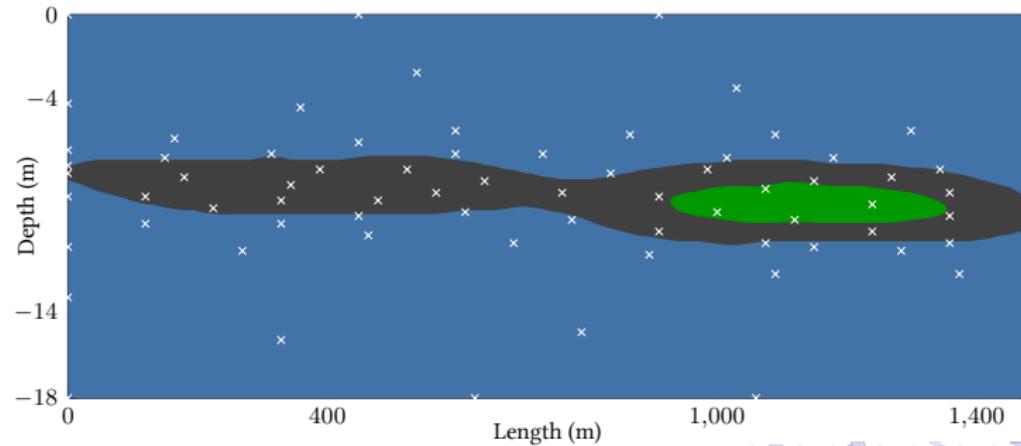


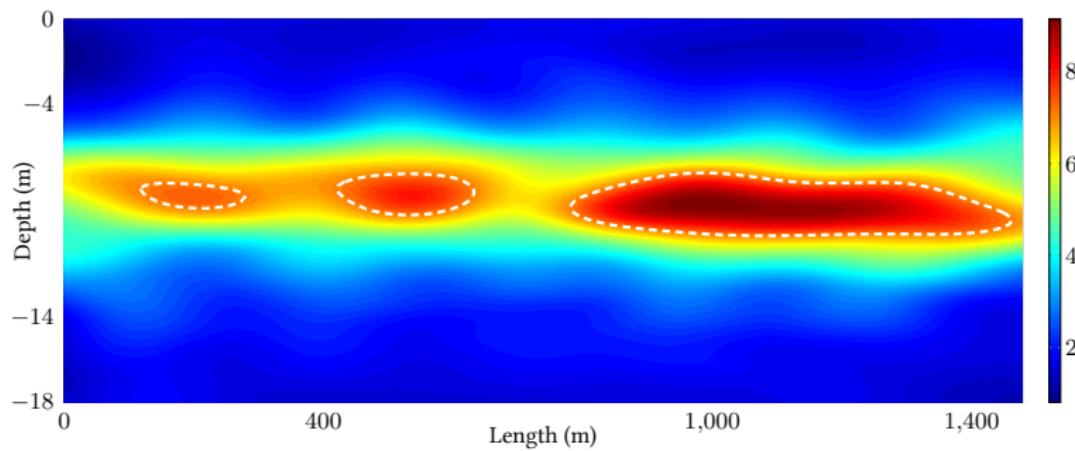
$t = 60$



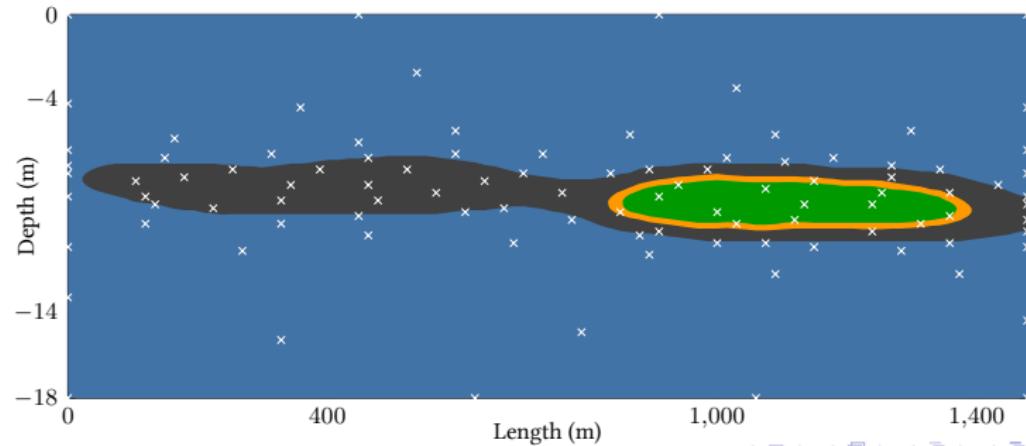


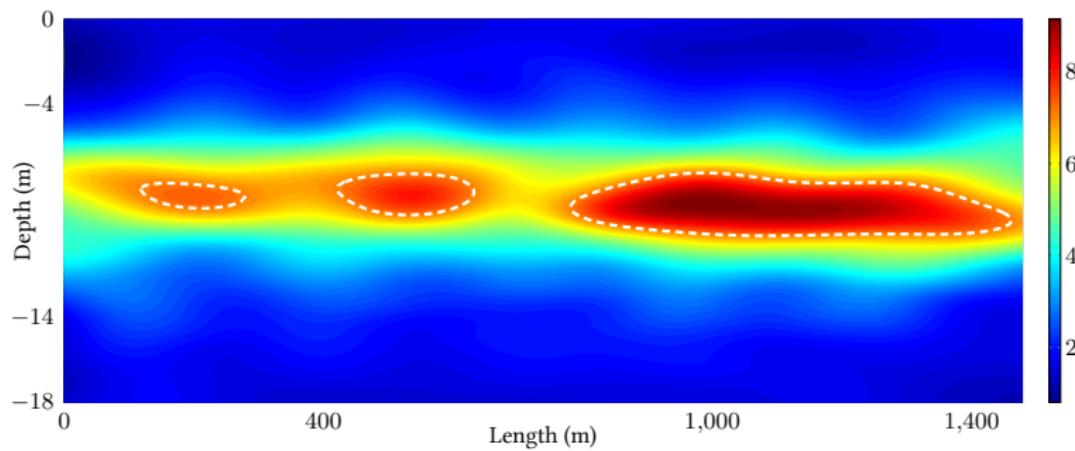
$t = 80$



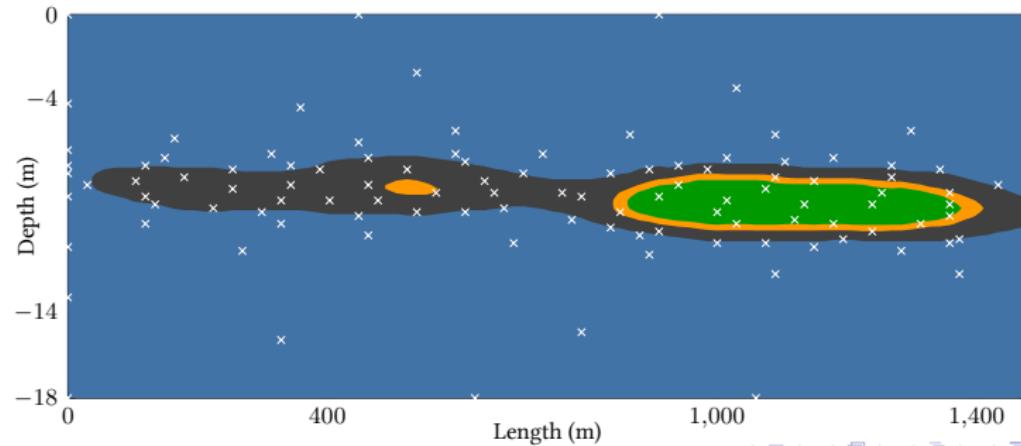


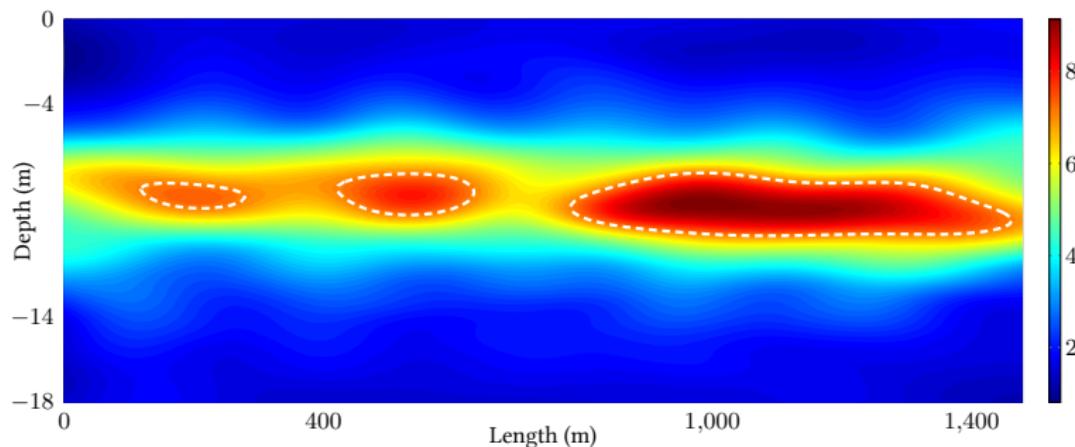
$t = 100$



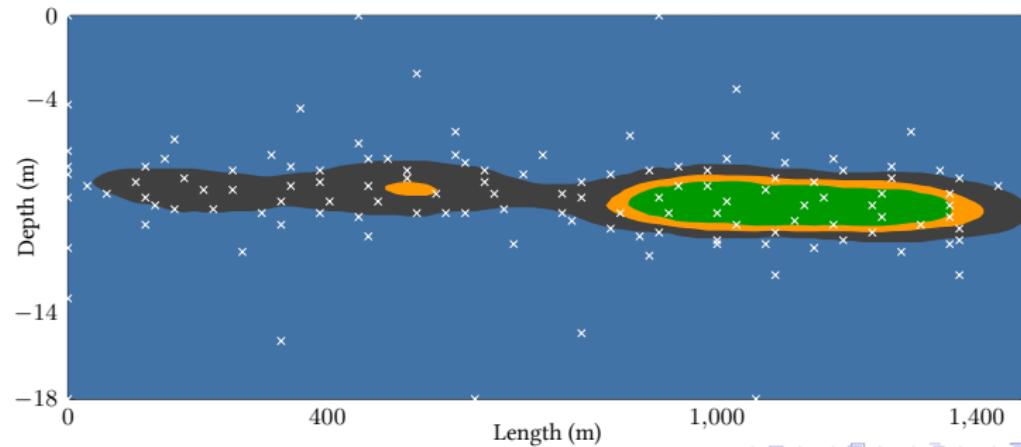


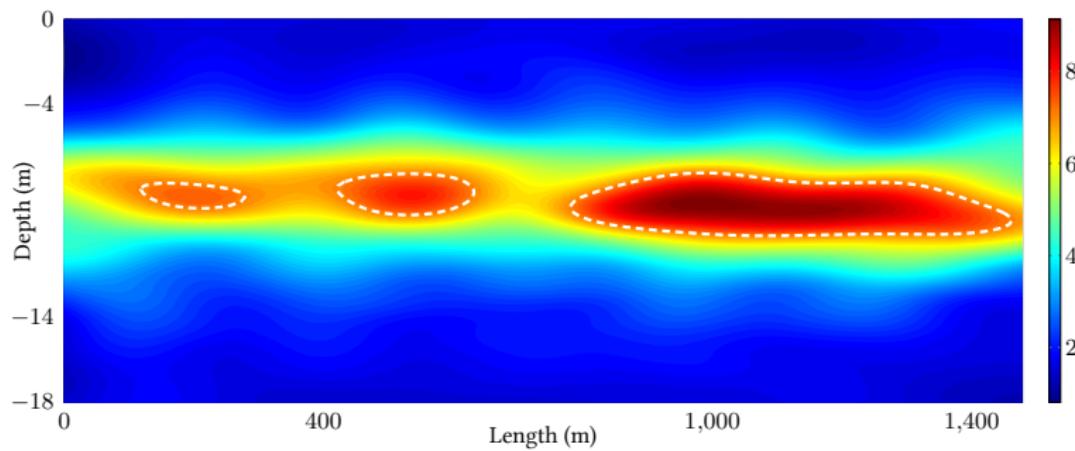
$t = 120$



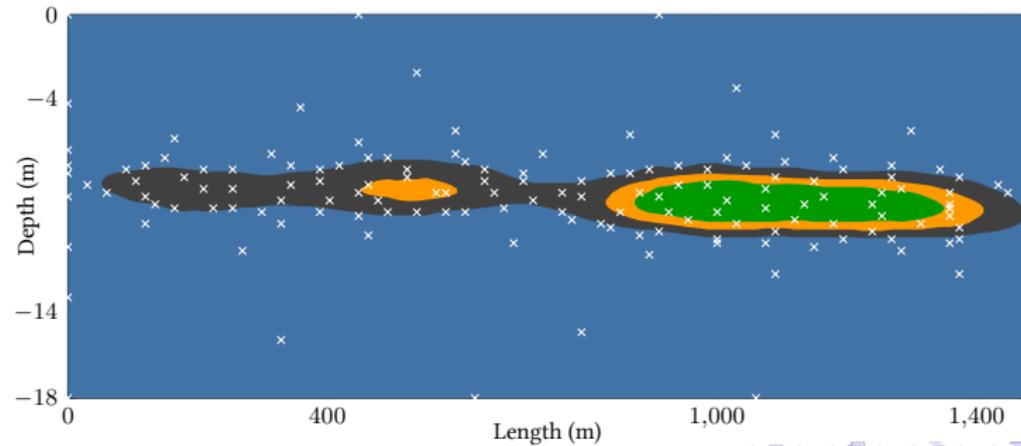


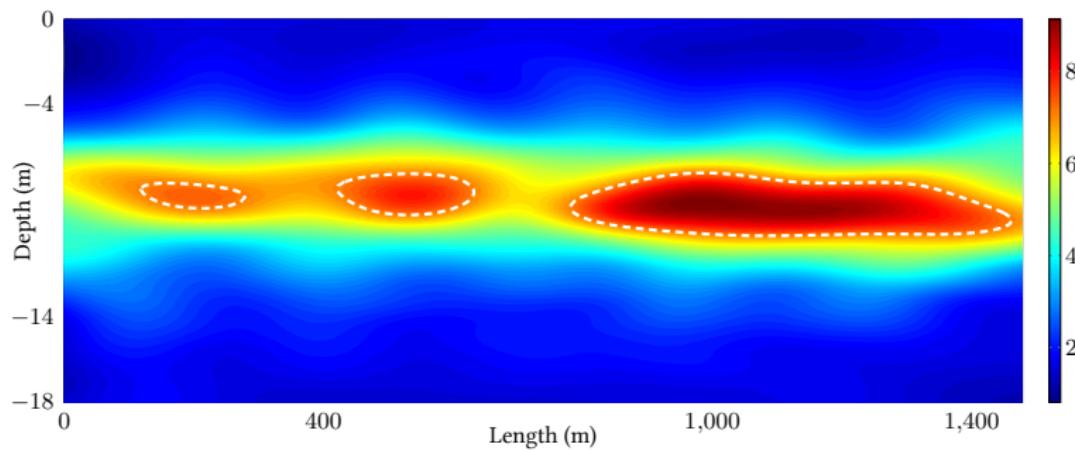
$t = 140$



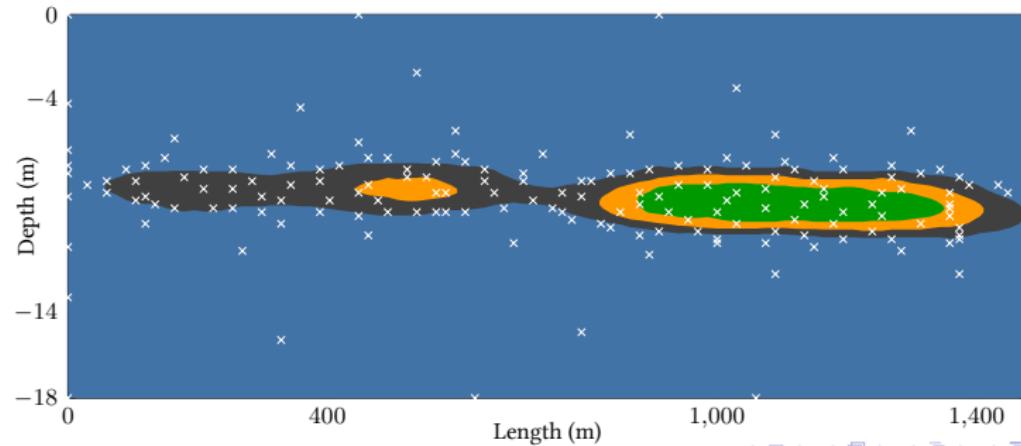


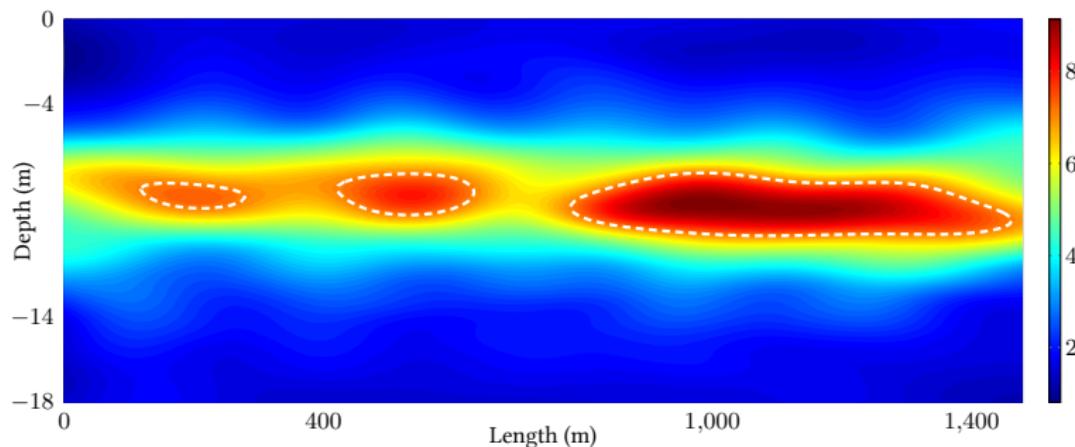
$t = 160$



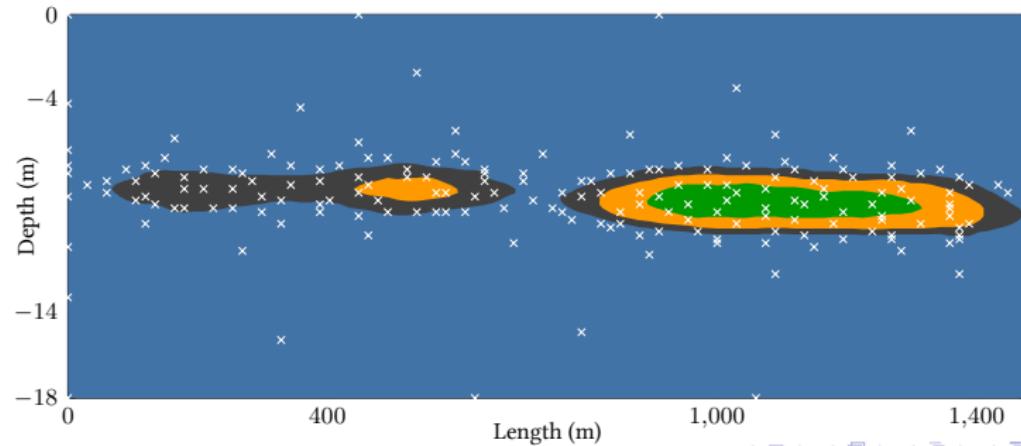


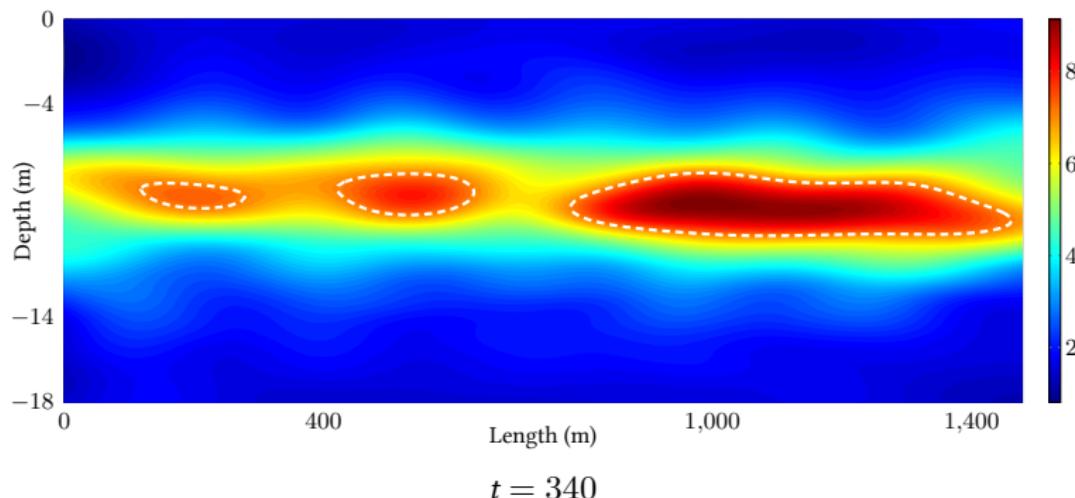
$t = 180$



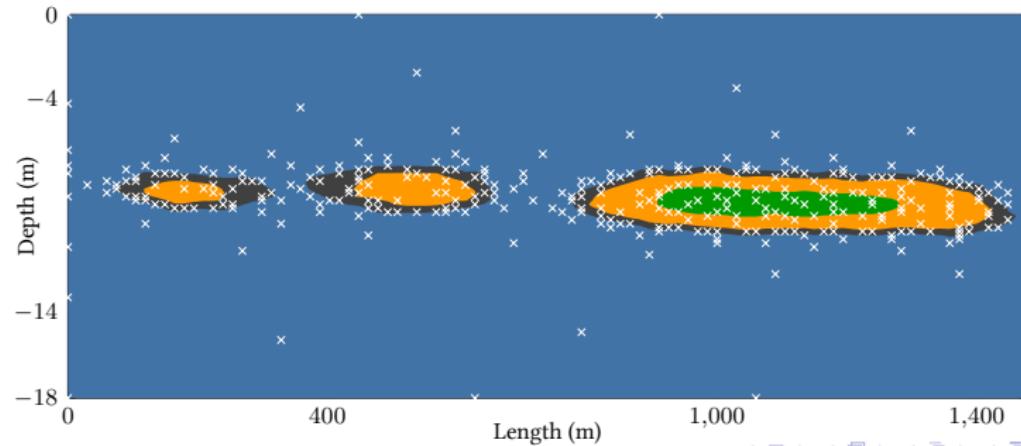


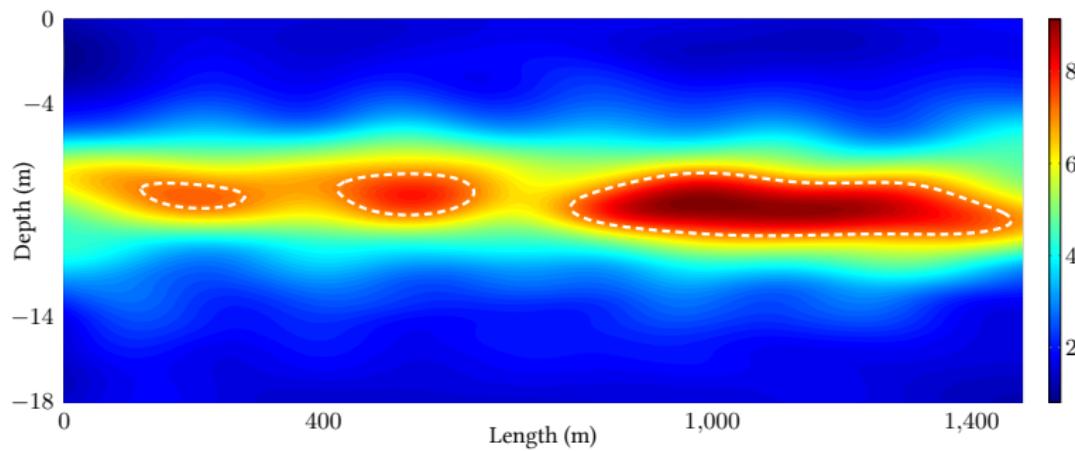
$t = 200$



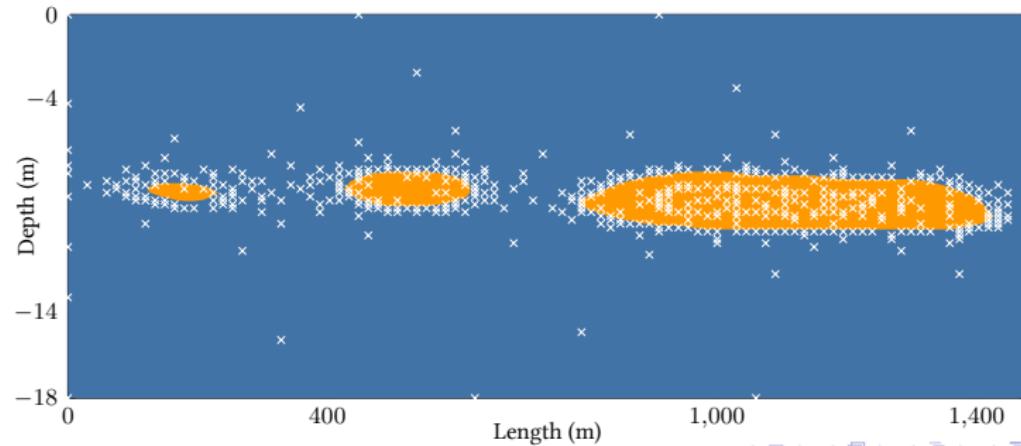


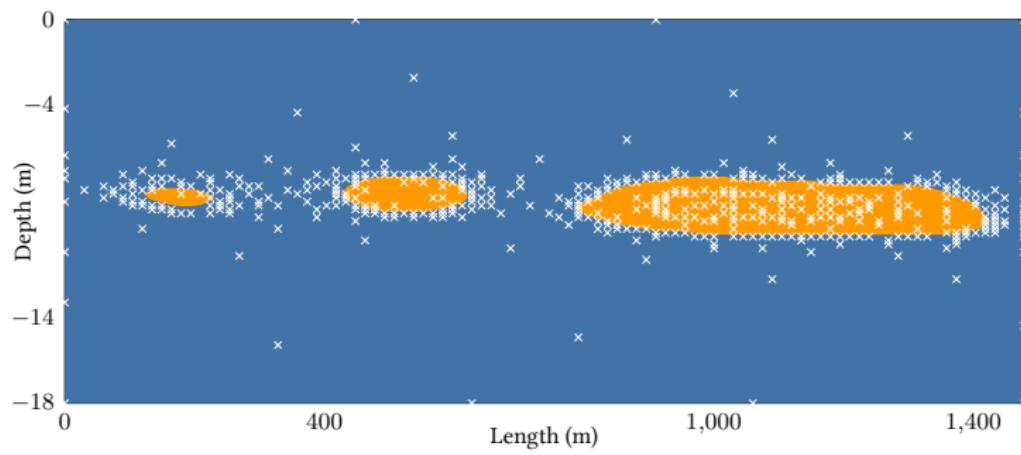
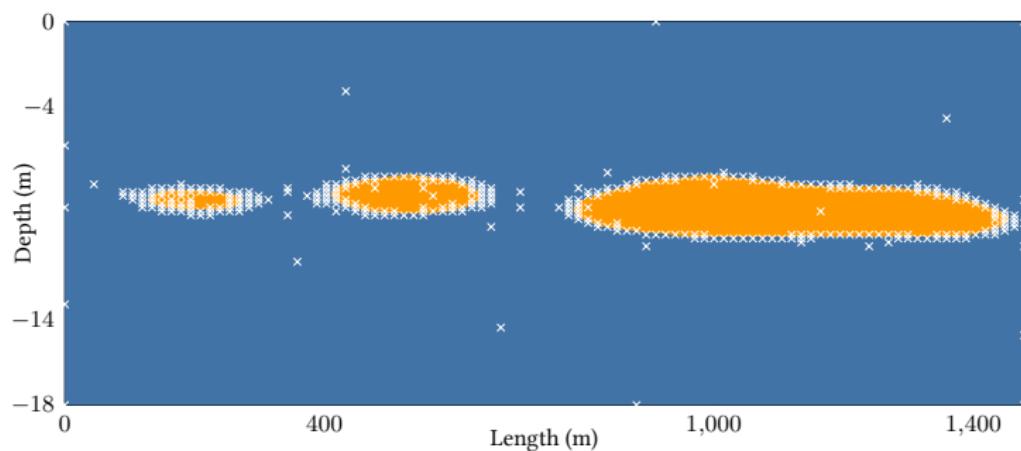
$t = 340$





$t = 486$



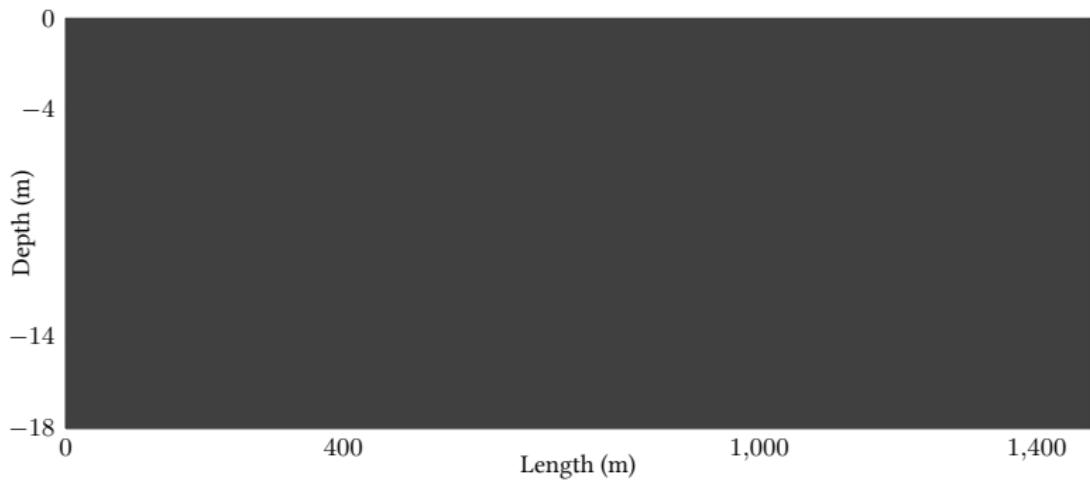


- ▶ Up to this point we have assumed a fixed cost per sample

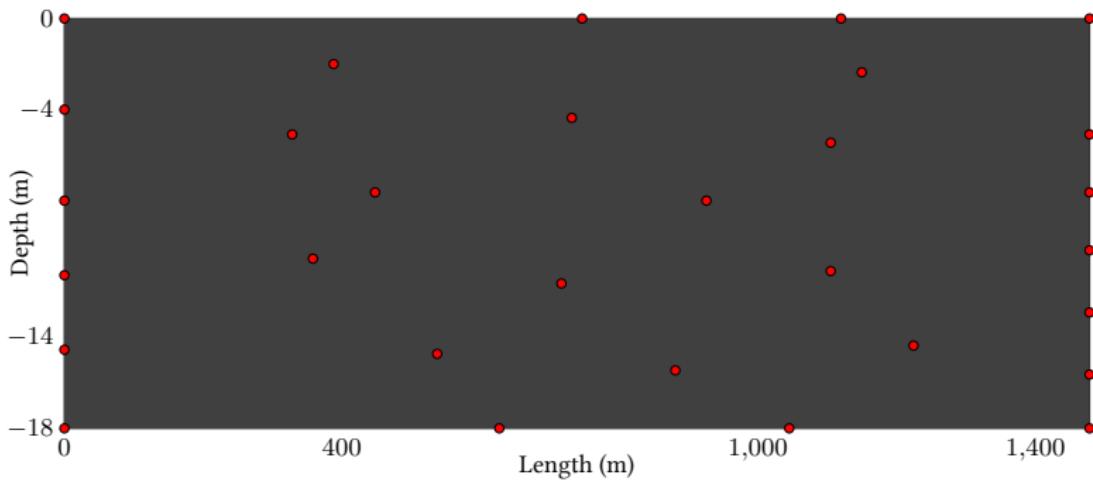
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?

- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $LSE_{batch}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements

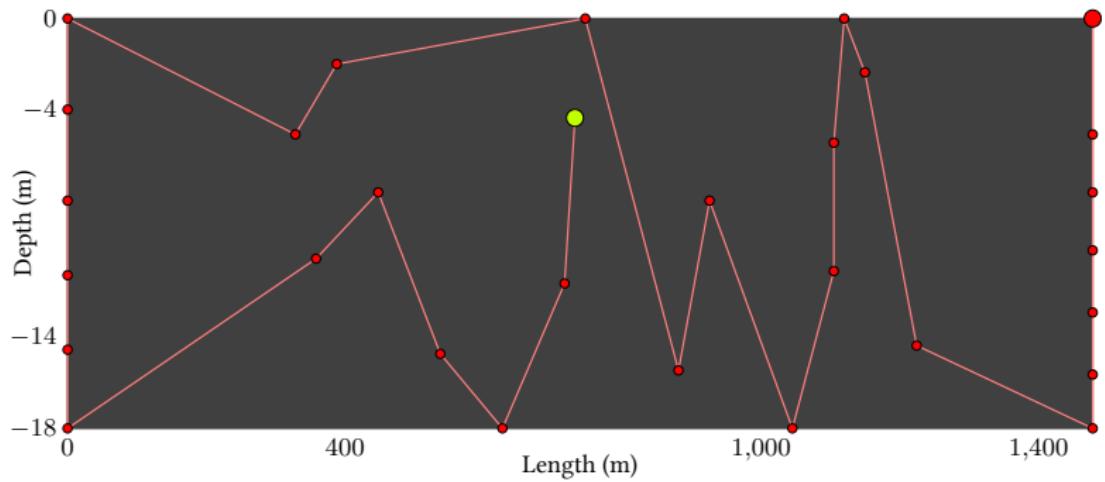
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



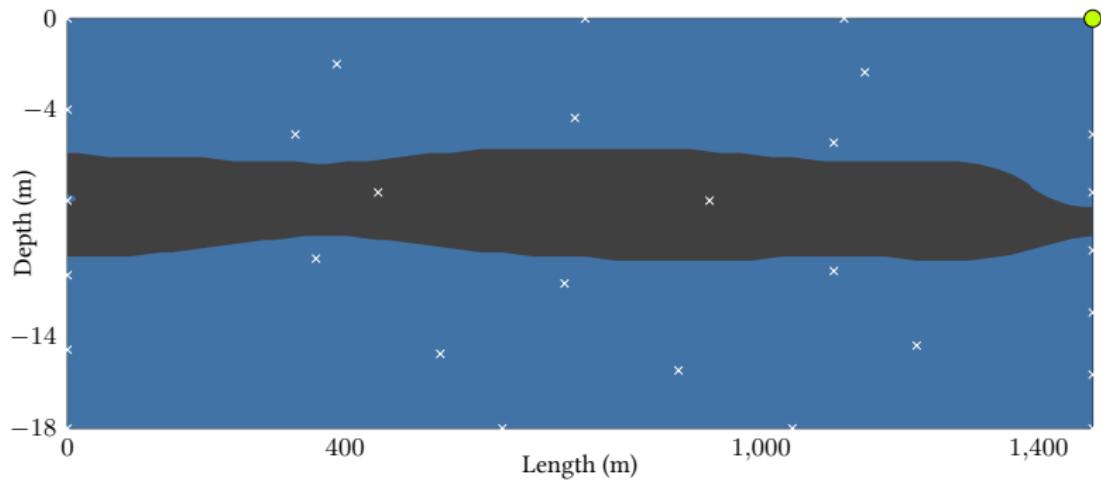
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



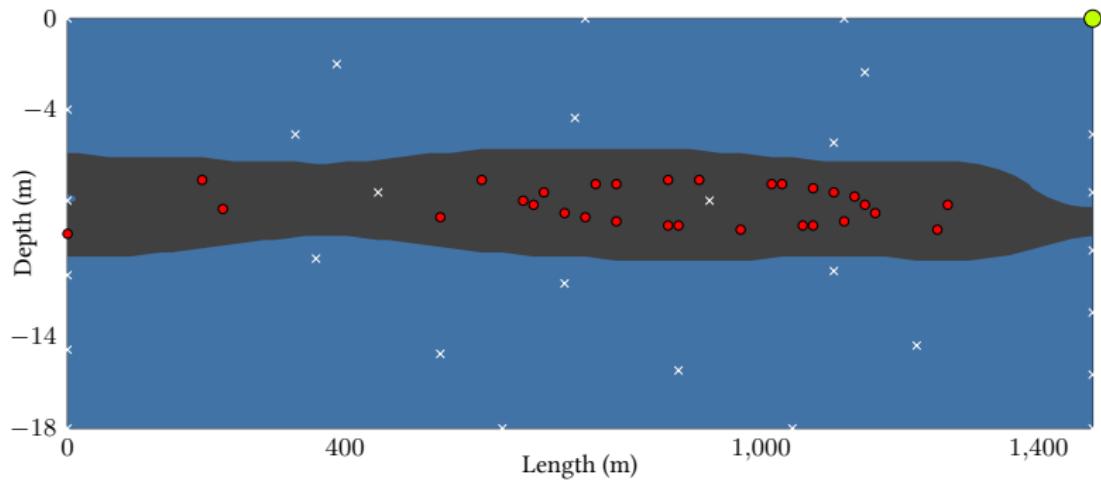
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



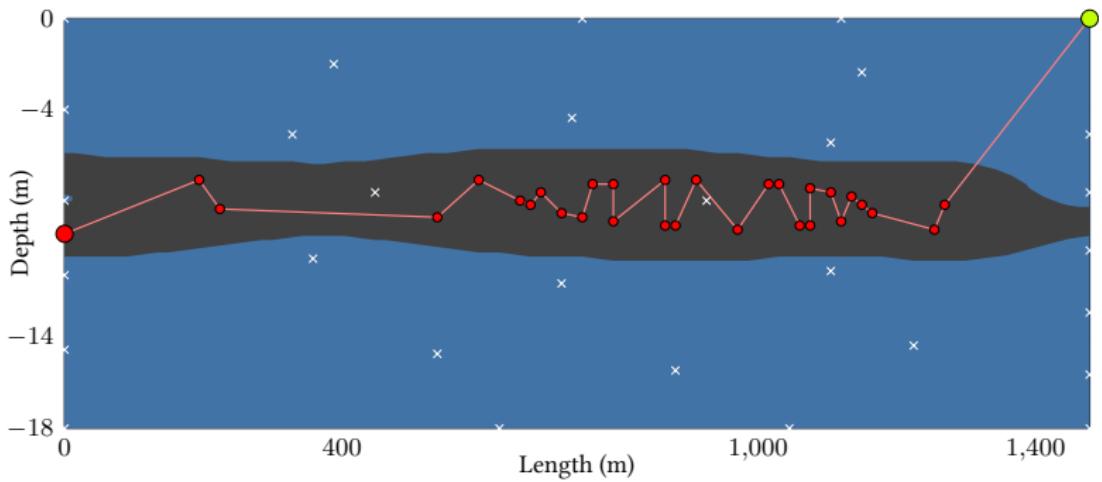
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



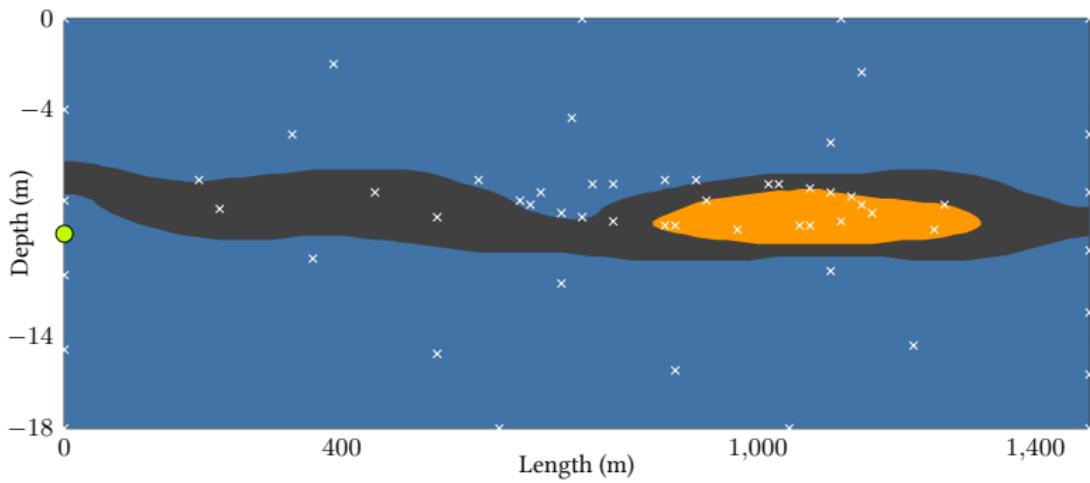
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



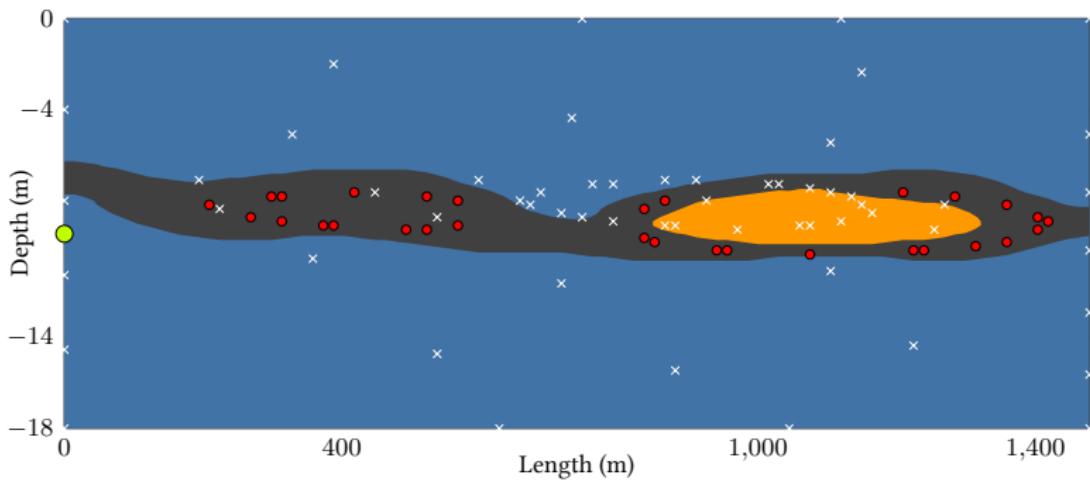
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



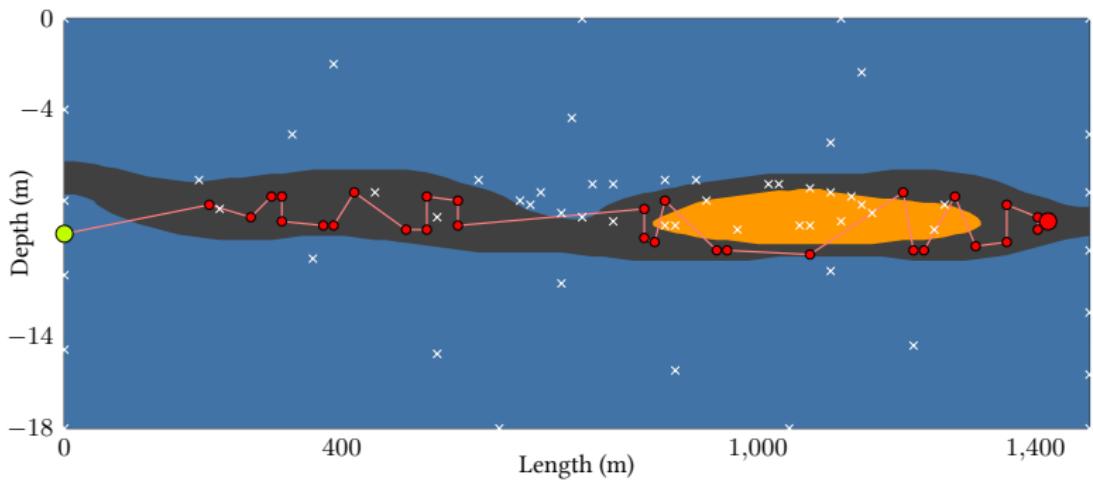
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



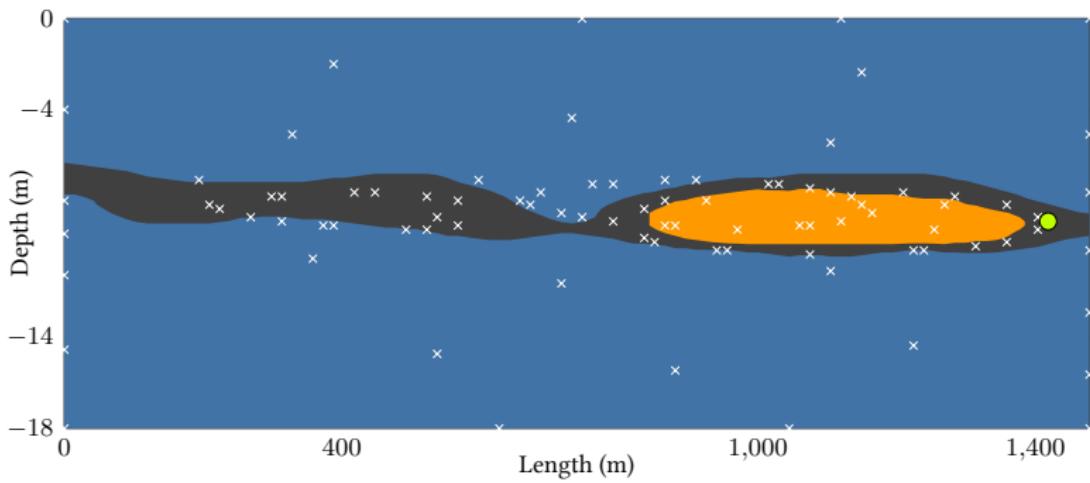
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



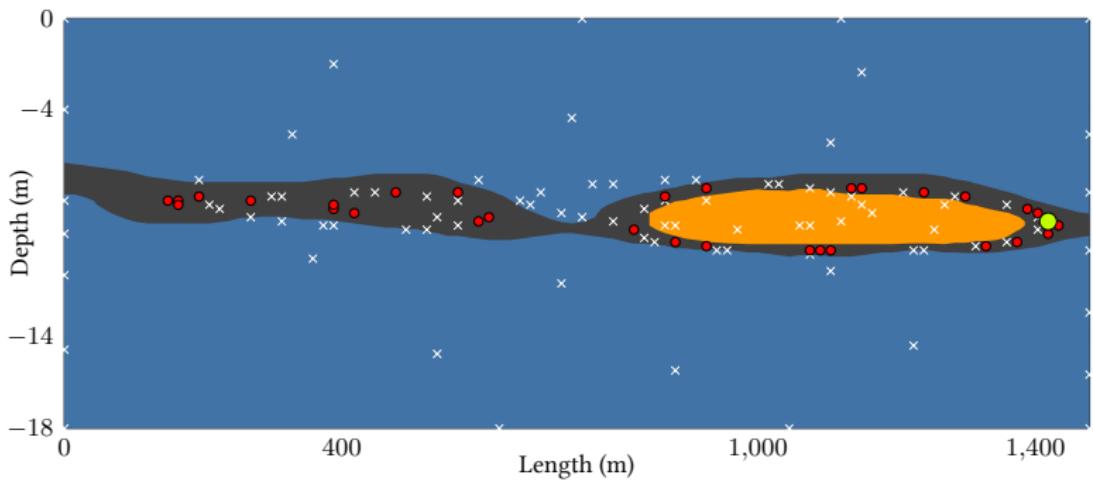
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



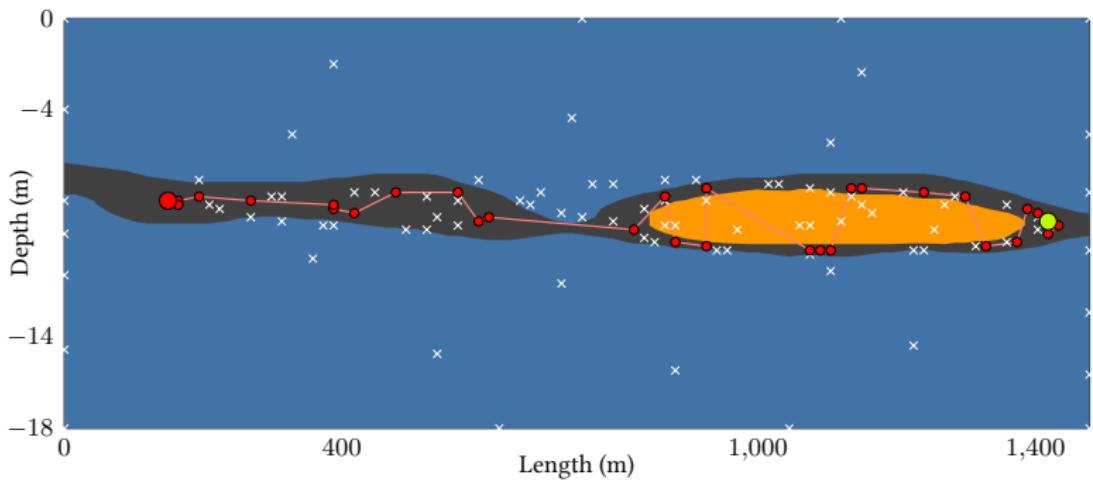
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



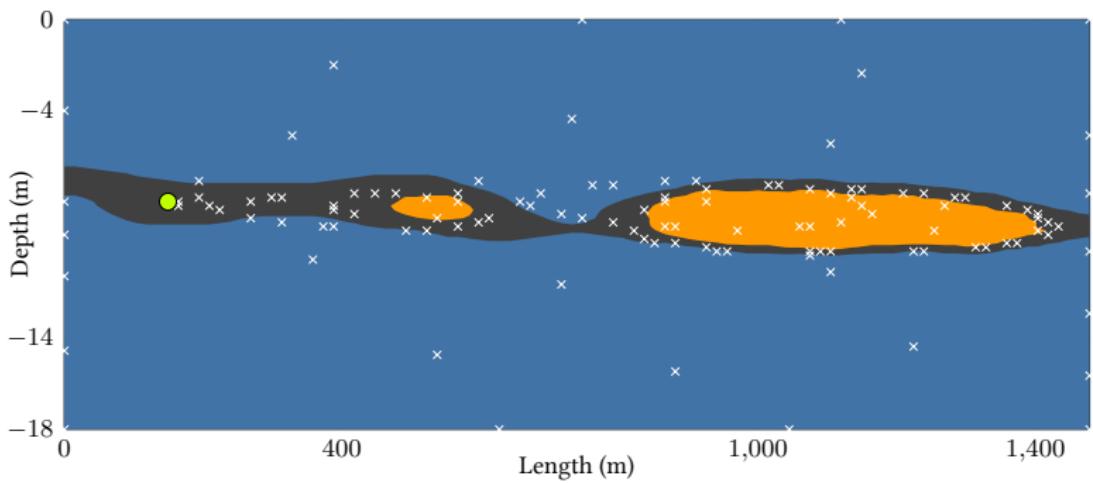
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



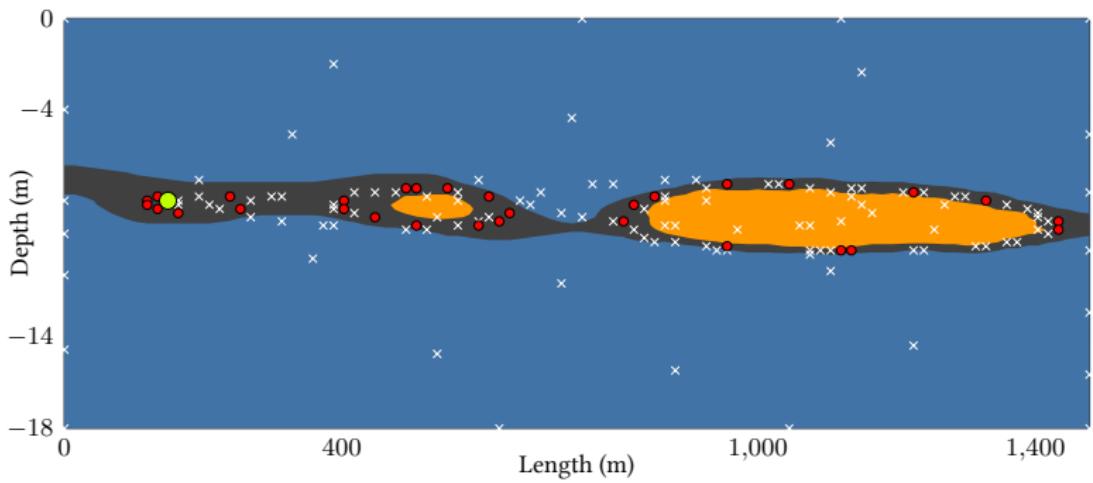
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



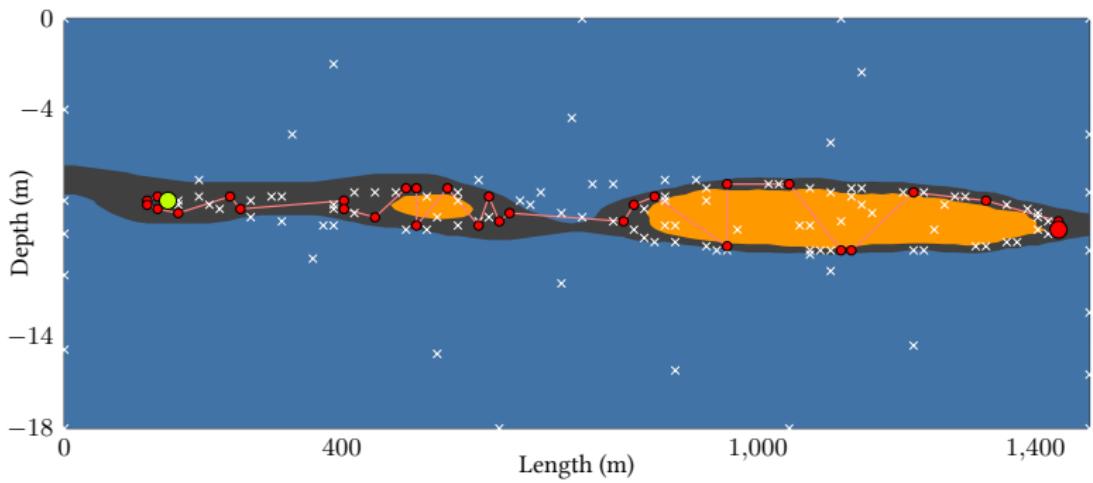
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



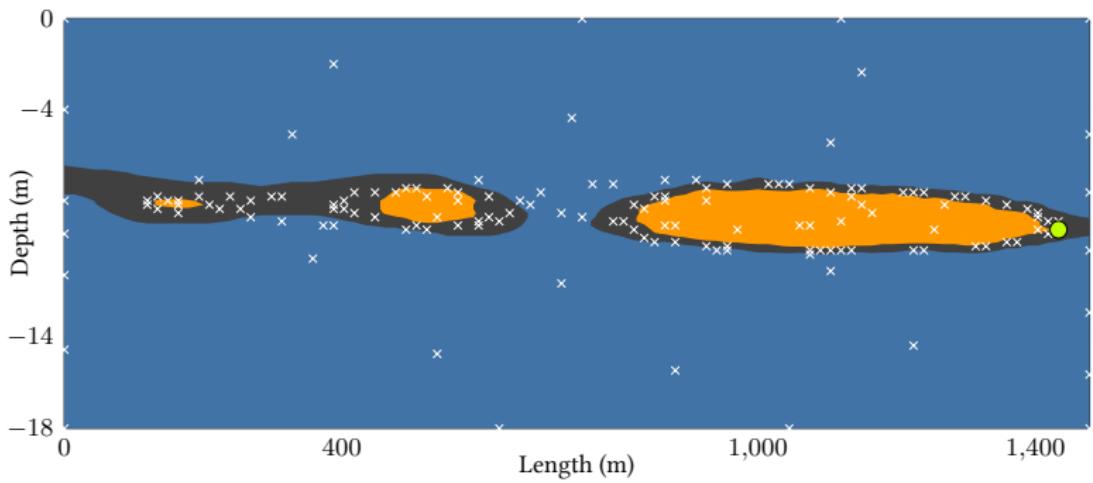
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



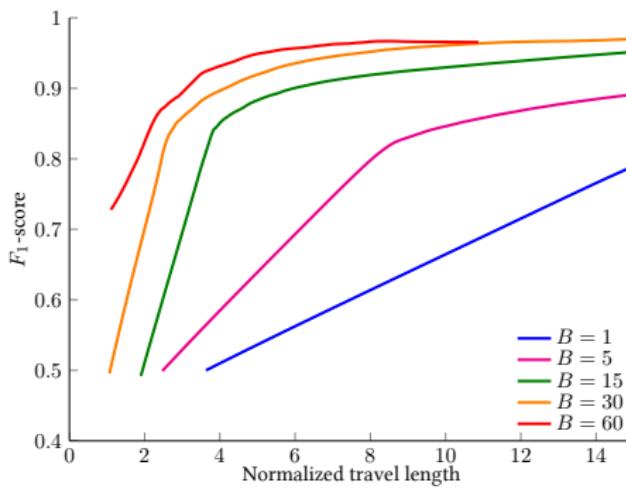
- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



- ▶ Up to this point we have assumed a fixed cost per sample
- ▶ What about the traveling distance between measurements?
- ▶ We extend LSE to select a *batch* of sampling locations at each step ( $\text{LSE}_{\text{batch}}$ )
- ▶ Use this to plan ahead:
  - ▶ Select a batch of sampling locations
  - ▶ Connect them using a Euclidean TSP path
  - ▶ Traverse path and collect measurements



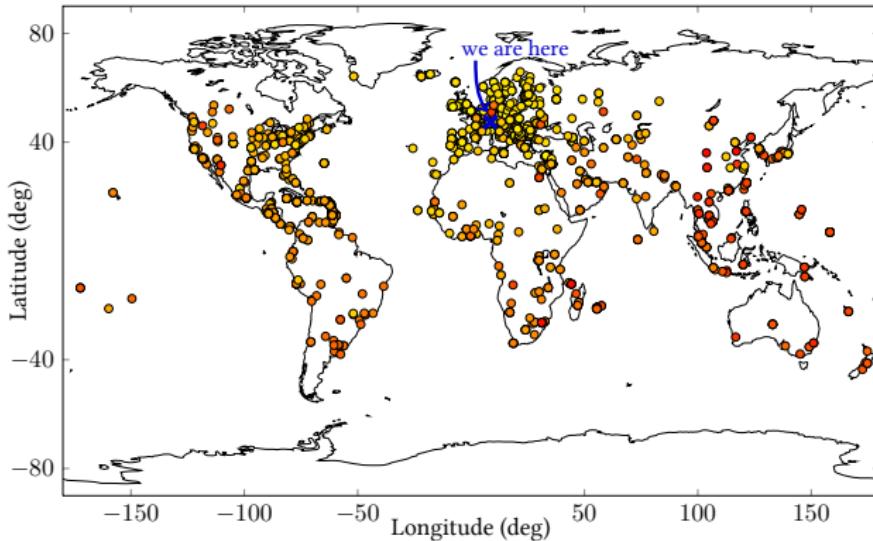
Come to the poster session for more...

Come to the poster session for more...

- ▶ ...theory: sample complexity bounds in more detail

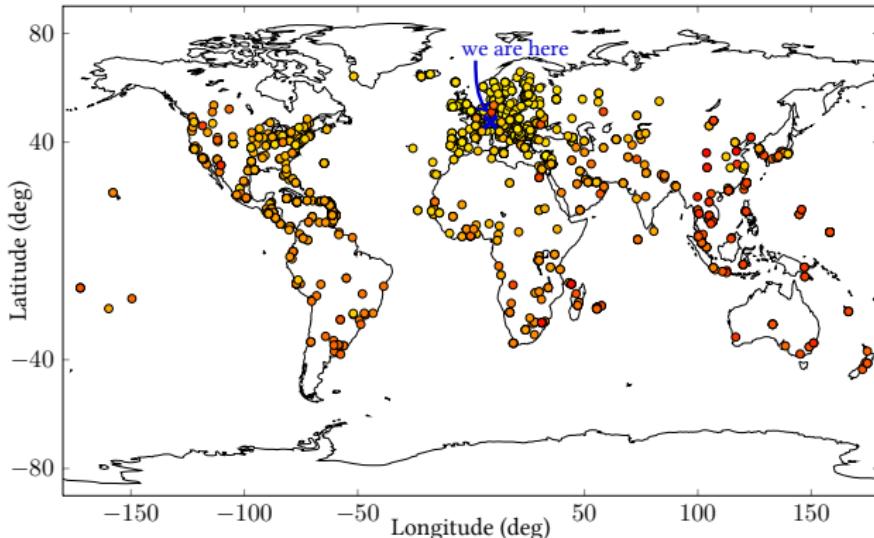
Come to the poster session for more...

- ▶ ...theory: sample complexity bounds in more detail
- ▶ ...applications: estimate world regions of low internet latency



Come to the poster session for more...

- ▶ ...theory: sample complexity bounds in more detail
- ▶ ...applications: estimate world regions of low internet latency



- ▶ ...results: why does LSE greatly outperform “straddle” in some cases?

# Summary

- ▶ LSE algorithm:

# Summary

## ► LSE algorithm:

### ► Theoretical guarantees

#### Theorem (Convergence of LSE)

For any  $\hbar \in \mathbb{R}$ ,  $\delta \in (0, 1)$ , and  $\epsilon > 0$ , if  $\beta_0 = 2 \log(D\pi^2)^2/(6\delta)$ , LSE terminates after at most  $T$  iterations, where  $T$  is the smallest positive integer satisfying

$$\frac{T}{\beta_0 \gamma} \geq \frac{C_1}{4\epsilon^2}$$

where  $C_1 = 8/\log(1 + \sigma^{-2})$ .

Furthermore, with probability at least  $1 - \delta$ , the algorithm returns an  $\epsilon$ -accurate solution, that is

$$\Pr \left\{ \max_{\mathbf{x} \in \mathcal{D}} \ell_k(\mathbf{x}) \leq \epsilon \right\} \geq 1 - \delta.$$

# Summary

- ▶ LSE algorithm:

- ▶ Theoretical guarantees

- ▶ Competitive with the state of the art in practice (sometimes considerably superior)

Theorem (Convergence of LSE)

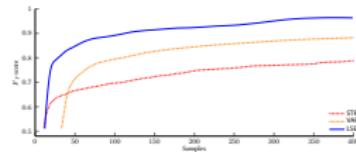
For any  $h \in \mathbb{R}$ ,  $\delta \in (0, 1)$ , and  $r > 0$ , if  $\beta_0 = 2 \log(D\pi^2 r^2 / (6\delta))$ , LSE terminates after at most  $T$  iterations, where  $T$  is the smallest positive integer satisfying

$$\frac{T}{\beta_0 T \gamma} \geq \frac{C_1}{4e^2},$$

where  $C_1 = 8/\log(1 + \sigma^{-2})$ .

Furthermore, with probability at least  $1 - \delta$ , the algorithm returns an  $\epsilon$ -accurate solution, that is

$$\Pr \left\{ \max_{x \in \mathcal{D}} \ell_h(x) \leq \epsilon \right\} \geq 1 - \delta.$$



# Summary

- ▶ LSE algorithm:

- ▶ Theoretical guarantees

- ▶ Competitive with the state of the art in practice (sometimes considerably superior)

- ▶ Two useful extensions:

Theorem (Convergence of LSE)

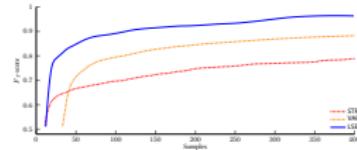
For any  $h \in \mathbb{R}$ ,  $\delta \in (0, 1)$ , and  $r > 0$ , if  $\beta_0 = 2 \log(|D(\pi^0)^2|/(6\delta))$ , LSE terminates after at most  $T$  iterations, where  $T$  is the smallest positive integer satisfying

$$\frac{T}{\beta_0 \gamma T} \geq \frac{C_1}{4e^2}$$

where  $C_1 = 8/\log(1 + \sigma^{-2})$ .

Furthermore, with probability at least  $1 - \delta$ , the algorithm returns an  $\epsilon$ -accurate solution, that is

$$\Pr \left\{ \max_{x \in D} \ell_h(x) \leq \epsilon \right\} \geq 1 - \delta.$$



# Summary

- ▶ LSE algorithm:

- ▶ Theoretical guarantees

- ▶ Competitive with the state of the art in practice (sometimes considerably superior)

- ▶ Two useful extensions:

- ▶ Implicit threshold level

Theorem (Convergence of LSE)

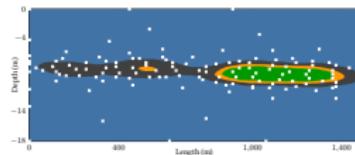
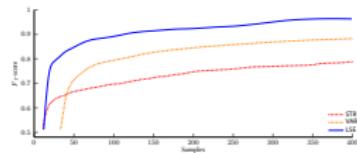
For any  $\hbar \in \mathbb{R}$ ,  $\delta \in (0, 1)$ , and  $c > 0$ , if  $\beta_0 = 2 \log(|D(\pi^0)^2|/(6\delta))$ , LSE terminates after at most  $T$  iterations, where  $T$  is the smallest positive integer satisfying

$$\frac{T}{\beta_0 T \gamma} \geq \frac{C_1}{4c^2}$$

where  $C_1 = 8/\log(1 + \sigma^{-2})$ .

Furthermore, with probability at least  $1 - \delta$ , the algorithm returns an  $\epsilon$ -accurate solution, that is

$$\Pr \left\{ \max_{x \in D} \ell_t(x) \leq \epsilon \right\} \geq 1 - \delta.$$



# Summary

## ► LSE algorithm:

### ► Theoretical guarantees

### ► Competitive with the state of the art in practice (sometimes considerably superior)

## ► Two useful extensions:

### ► Implicit threshold level

### ► Batch sampling

#### Theorem (Convergence of LSE)

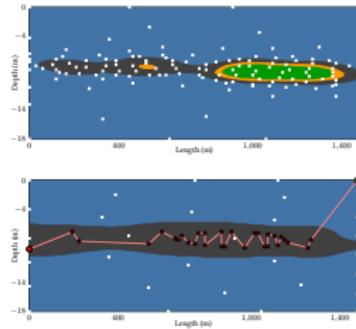
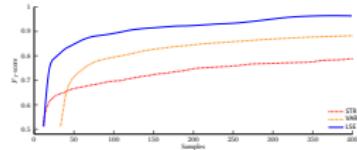
For any  $\bar{h} \in \mathbb{R}$ ,  $\delta \in (0, 1)$ , and  $c > 0$ , if  $\beta_0 = 2 \log(|D(\pi^0)^2|/(6\delta))$ , LSE terminates after at most  $T$  iterations, where  $T$  is the smallest positive integer satisfying

$$\frac{T}{\beta_0 \gamma \tau} \geq \frac{C_1}{4c^2}$$

where  $C_1 = 8/\log(1 + \sigma^{-2})$ .

Furthermore, with probability at least  $1 - \delta$ , the algorithm returns an  $\epsilon$ -accurate solution, that is

$$\Pr \left\{ \max_{x \in D} \ell_t(x) \leq \epsilon \right\} \geq 1 - \delta.$$



# Summary

## ► LSE algorithm:

### ► Theoretical guarantees

### ► Competitive with the state of the art in practice (sometimes considerably superior)

## ► Two useful extensions:

### ► Implicit threshold level

### ► Batch sampling

## ► Look out for algae when swimming in Lake Zurich! 😊

### Theorem (Convergence of LSE)

For any  $\hbar \in \mathbb{R}$ ,  $\delta \in (0, 1)$ , and  $c > 0$ , if  $\beta_1 = 2 \log(|D(\pi^T)^2|/(6\delta))$ , LSE terminates after at most  $T$  iterations, where  $T$  is the smallest positive integer satisfying

$$\frac{T}{\beta_1 \gamma \tau} \geq \frac{C_1}{4c^2},$$

where  $C_1 = 8/\log(1 + \sigma^{-2})$ .

Furthermore, with probability at least  $1 - \delta$ , the algorithm returns an  $\epsilon$ -accurate solution, that is

$$\Pr \left\{ \max_{x \in D} \ell_1(x) \leq \epsilon \right\} \geq 1 - \delta.$$

