Bayesian optimization of ocean mixed layer parameterizations

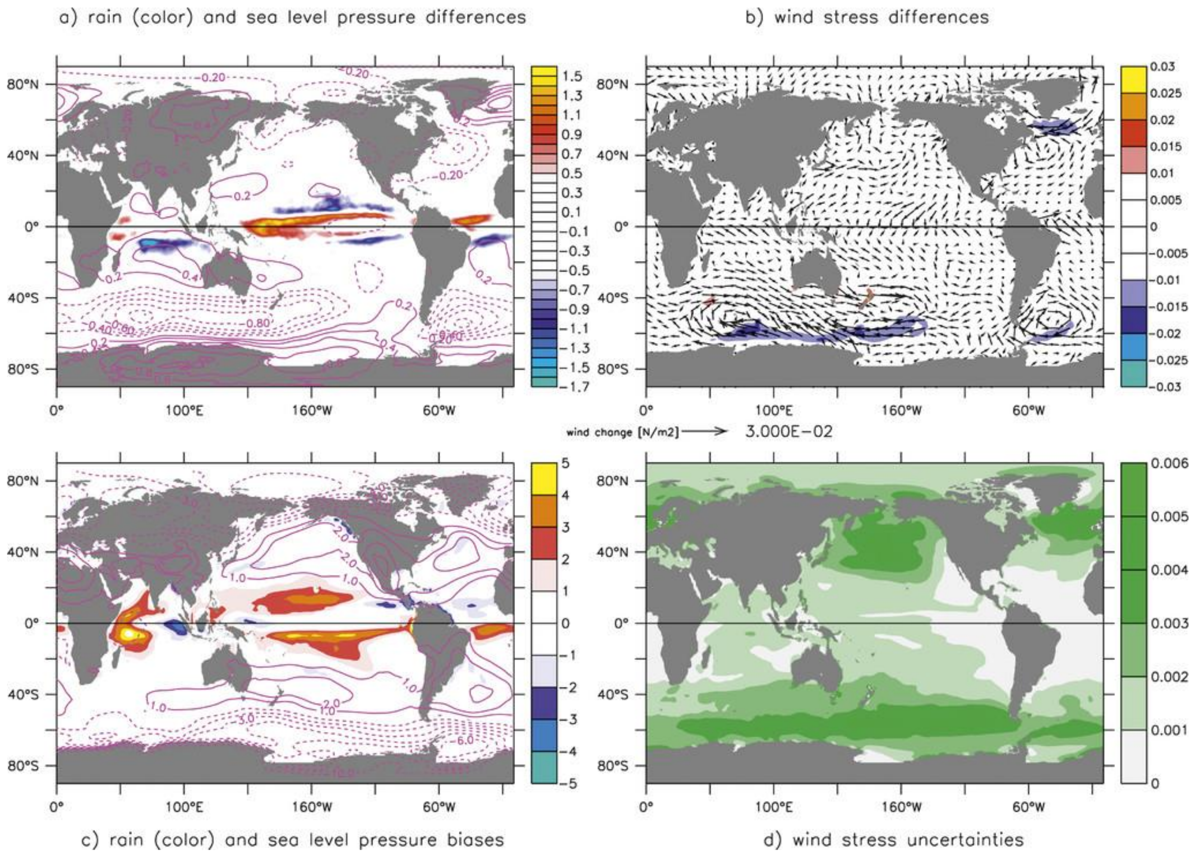
Marta Mrozowska,
Markus Jochum,
James Avery,
Ida Stoustrup,
Roman Nuterman and
Carl-Johannes Johnsen

20/08/2024

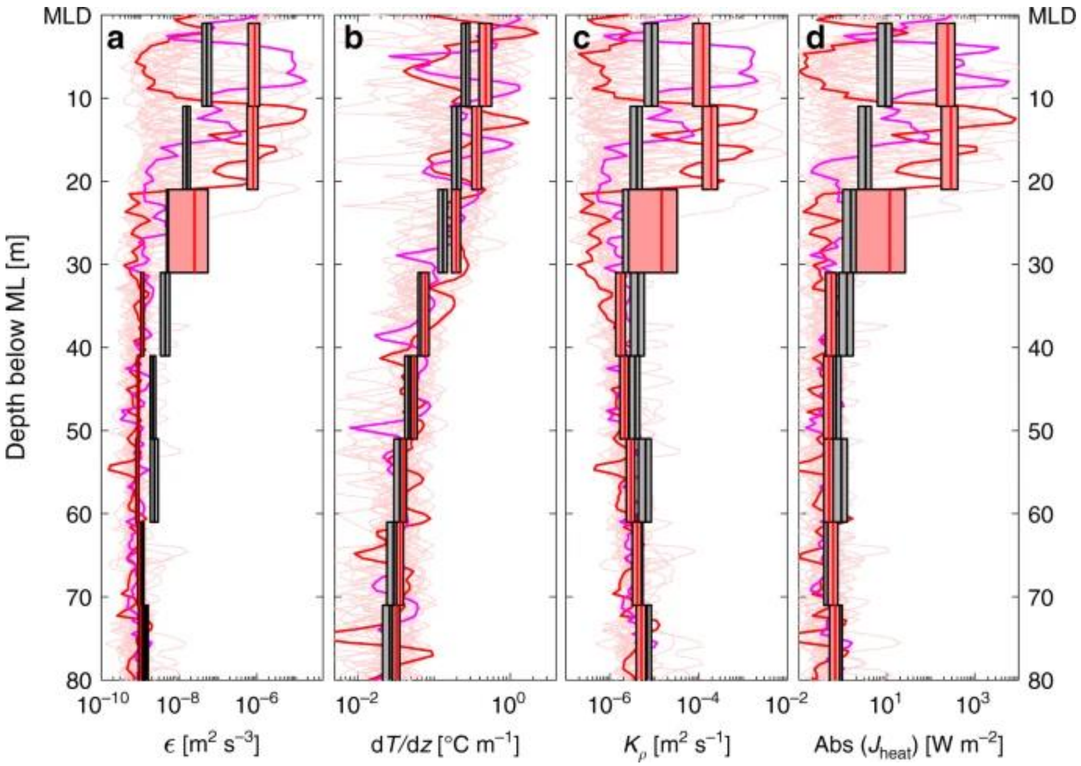
HAMLET-Physics, KU


Super yacht Bayesian sinks after encounter with extremely rare water spout (20/8/24, FT)

Tropical SST anomalies can lead to restructuring of the global atmosphere



Jochum et al. (2013)



A rare direct observation of a strong mixing event (Hummels et al. 2020). The turbulent diffusivity (3 orders larger than molecular) is shown in panel c.

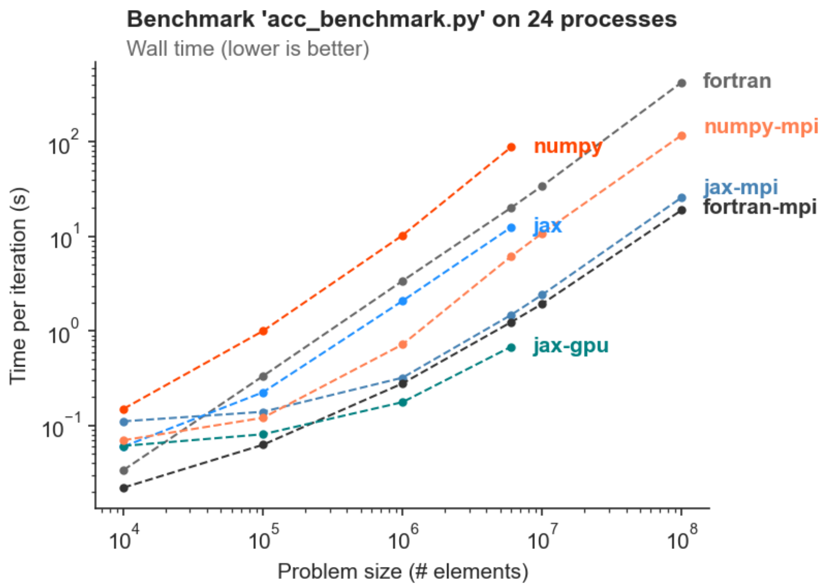
One of the largest sources of uncertainty is vertical mixing

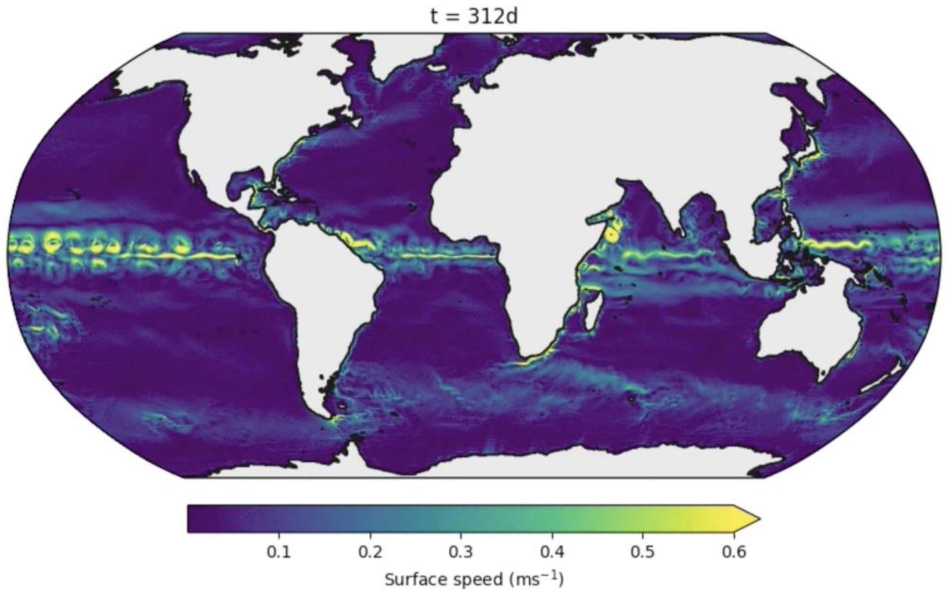
- Vertical turbulent mixing creates a homogeneous surface layer that, like a skin, exchanges heat and momentum with the atmosphere
- Mixing is difficult to observe, but the mixed layer depth (MLD) is well observed and a key metric for model performance

Foltz et al. (2003)

Veros: Versatile Ocean Simulator

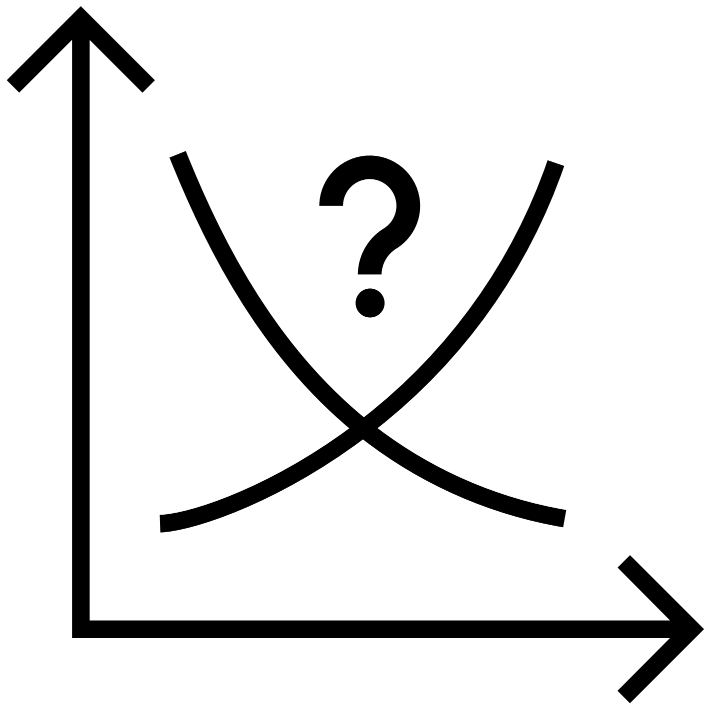
Python/JAX
MPI for GPUs





Fortran on 2000 CPUs or Python/JAX on 16 A100 GPUs
...at a fifth of the energy!

Häfner et al. (2021)

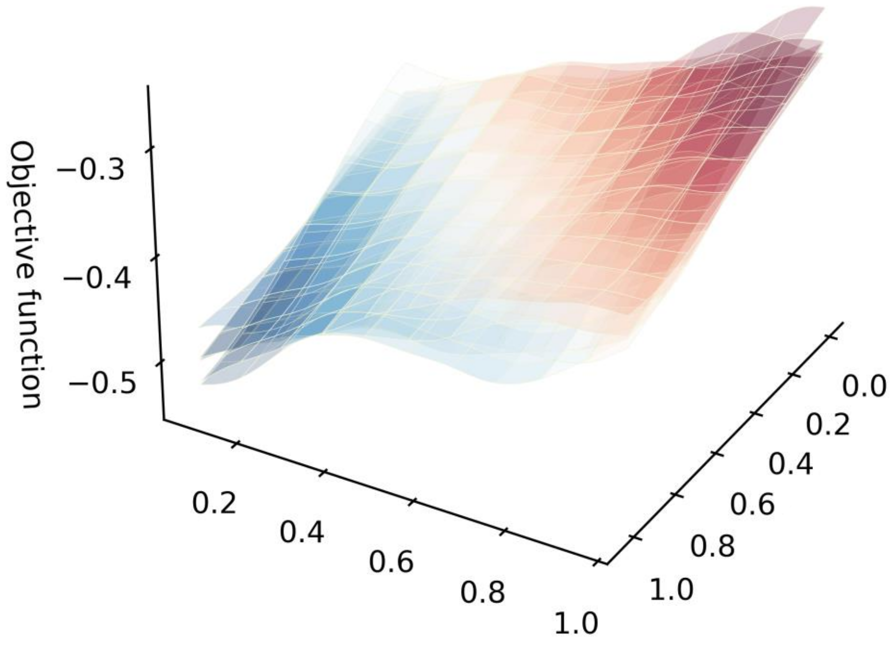


The Problem

- We want to optimize the **objective function**

$$f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$$

- We don't know anything about the function shape (so-called *black box objective function*)
- The objective function is **expensive to evaluate**



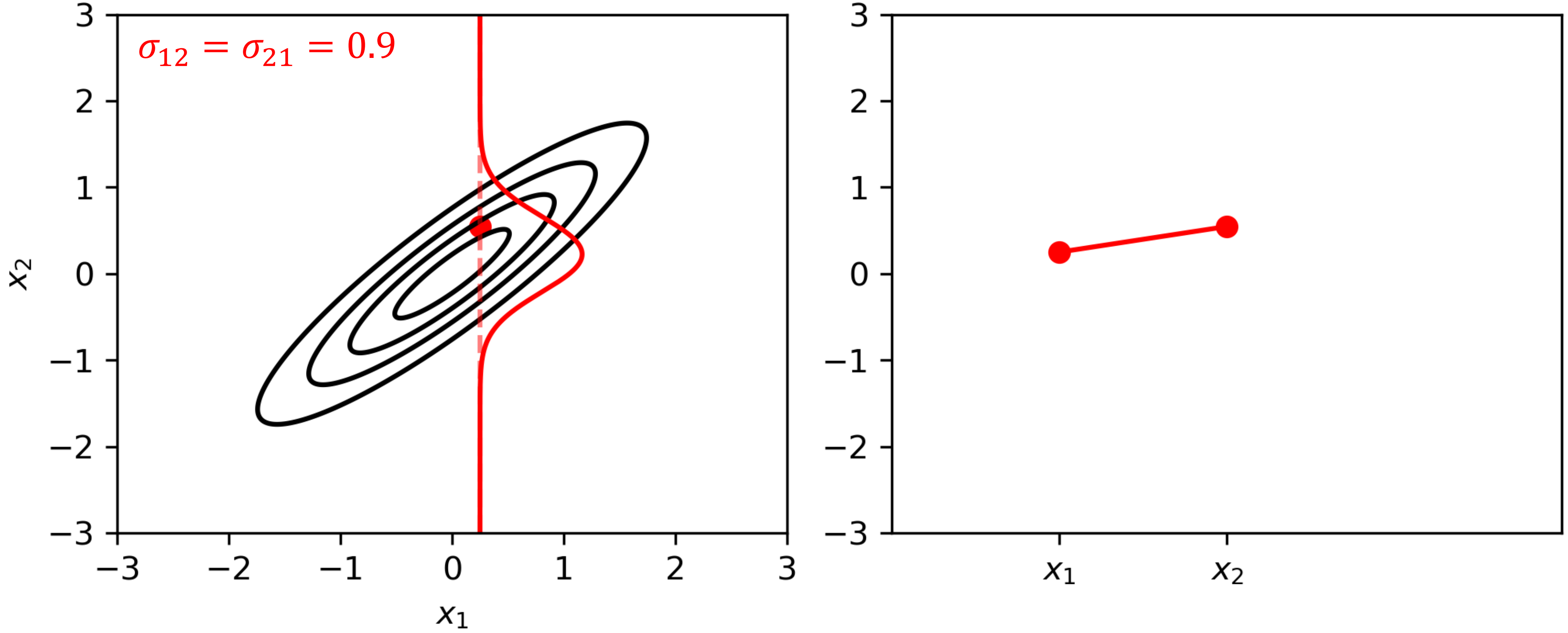
The Solution: Bayesian Optimization

- Based on a few objective function evaluations, **construct a surrogate model** of the objective function over the full parameter space
- Using the model of the objective function, **decide the next optimal parameter set to evaluate**

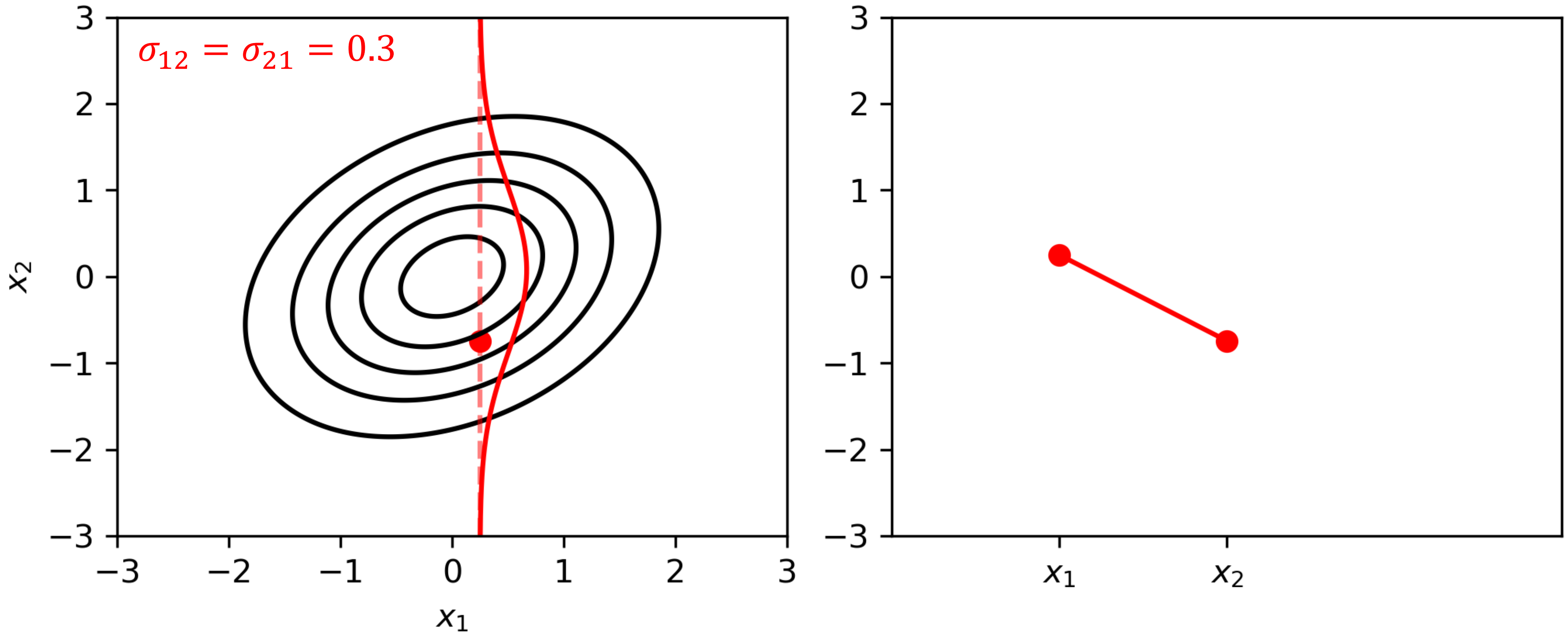
Agenda

1. Gaussian process regression models
2. Bayesian optimization with VerOpt
3. Optimizing the turbulent kinetic energy closure scheme in Veros

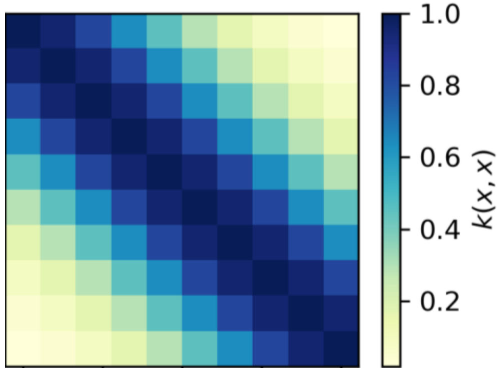
Gaussian process (GP) regression models

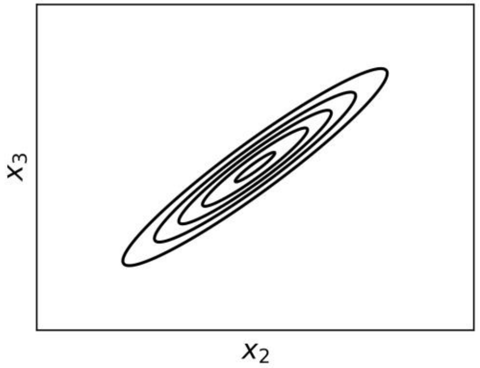


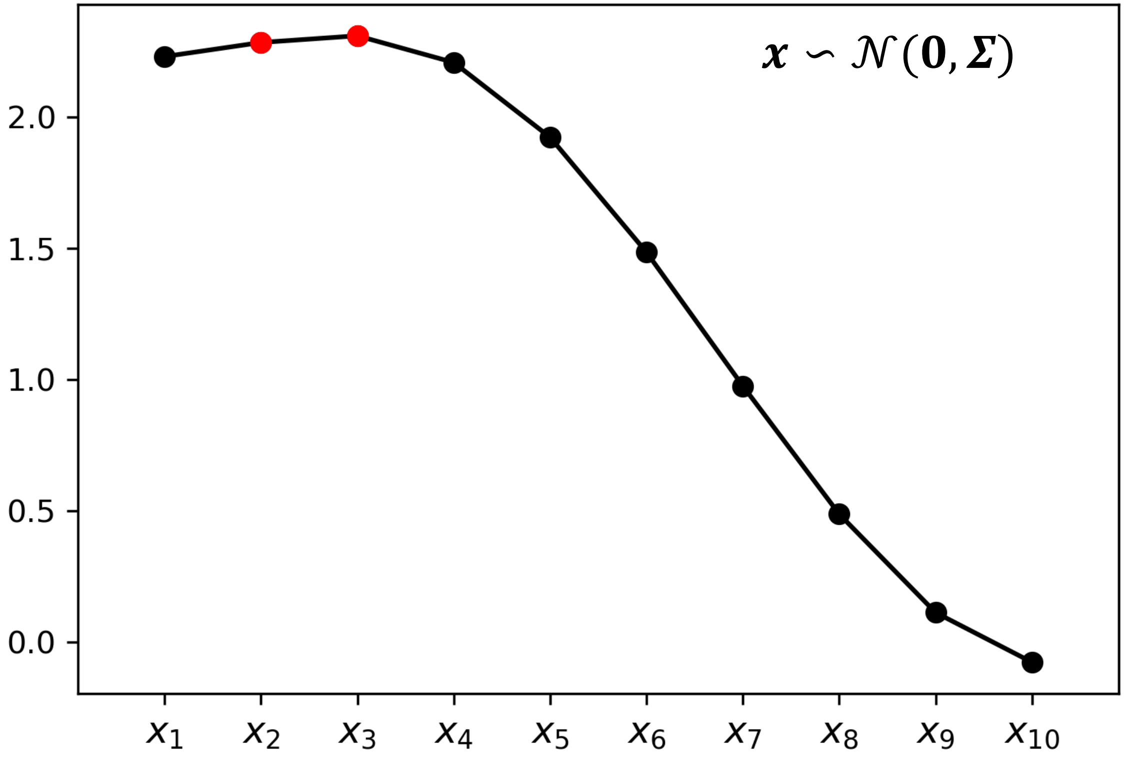
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \backsim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}\right)$$



$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \backsim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}\right)$$

$\boldsymbol{\Sigma} =$ 

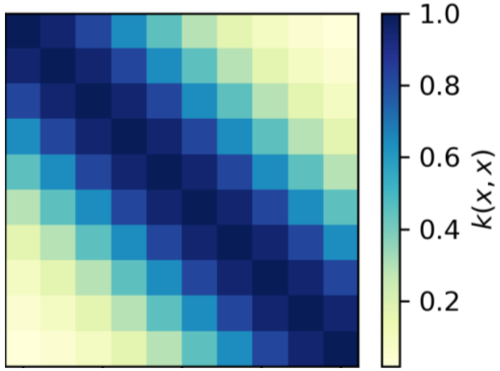


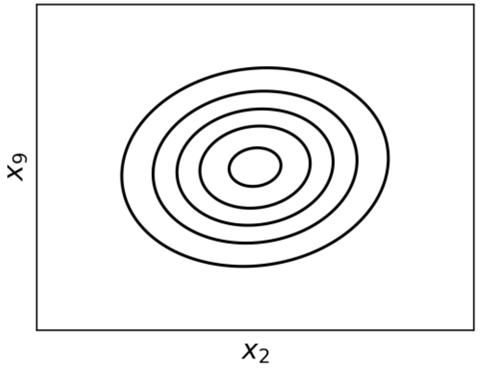


$$\sigma_{23} = \sigma_{32} \approx 0.95$$

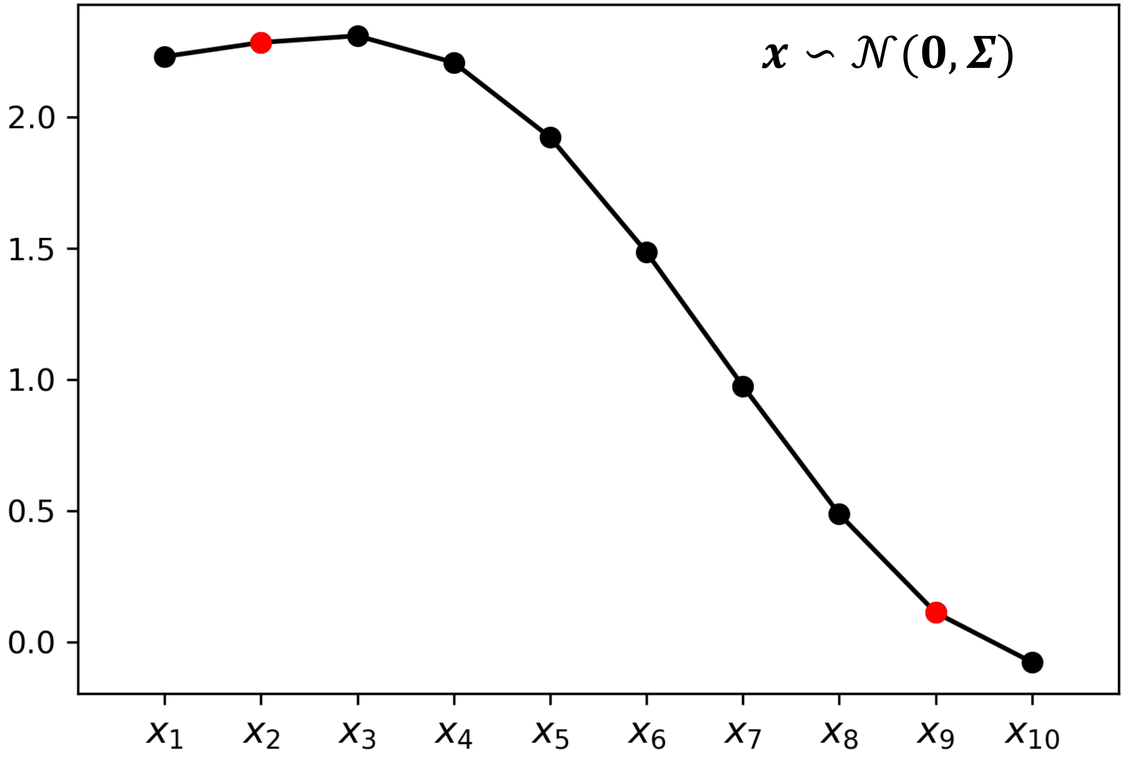
$$\begin{bmatrix} x_2 \\ x_3 \end{bmatrix} \backsim \mathcal{N}\left(\begin{bmatrix} \mu_2 \\ \mu_3 \end{bmatrix}, \begin{bmatrix} \sigma_{22} & \sigma_{23} \\ \sigma_{32} & \sigma_{33} \end{bmatrix}\right)$$

$\boldsymbol{\Sigma} =$

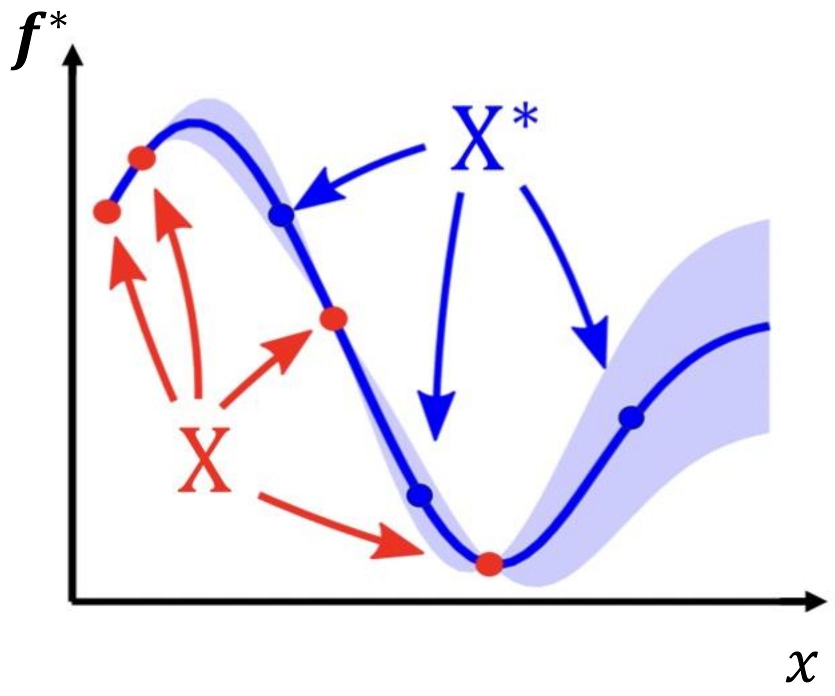




$\sigma_{23} = \sigma_{32} \approx 0.09$



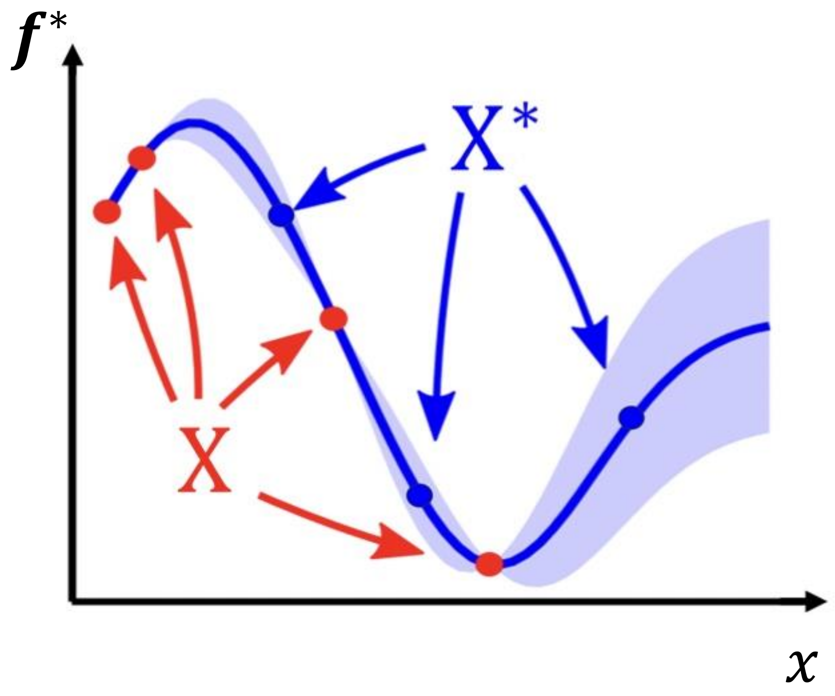
$$\begin{bmatrix} x_2 \\ x_9 \end{bmatrix} \backsim \mathcal{N}\left(\begin{bmatrix} \mu_2 \\ \mu_9 \end{bmatrix}, \begin{bmatrix} \sigma_{22} & \sigma_{29} \\ \sigma_{92} & \sigma_{99} \end{bmatrix}\right)$$



GP regression generalized

- **X**: a set of input points
- **X***: a set of test points
- Joint distribution:

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\boldsymbol{X}) \\ m(\boldsymbol{X}^*) \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K}^* \\ \boldsymbol{K}^* & \boldsymbol{K}^{**} \end{bmatrix} \right)$$

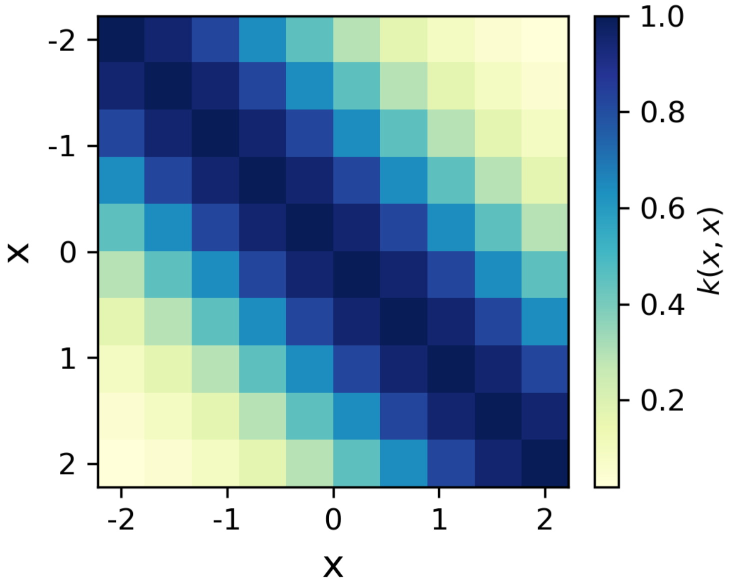


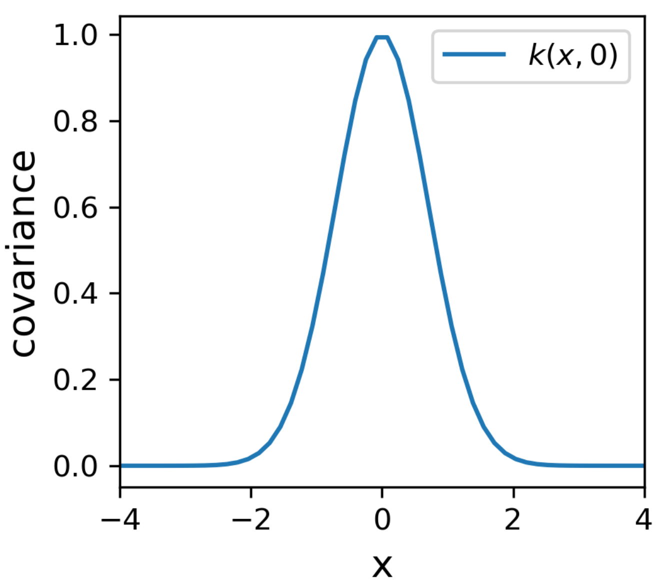
GP regression generalized

- **X**: a set of input points
- **X***: a set of test points
- Conditional distribution:

$$\boldsymbol{f}^*|\boldsymbol{f},\boldsymbol{X},\boldsymbol{X}^* \sim \mathcal{N}\left(\boldsymbol{K}^{*\boldsymbol{T}}\boldsymbol{K}^{-1}\boldsymbol{f},\boldsymbol{K}^{**}-\boldsymbol{K}^{*\boldsymbol{T}}\boldsymbol{K}^{-1}\boldsymbol{K}\right)$$

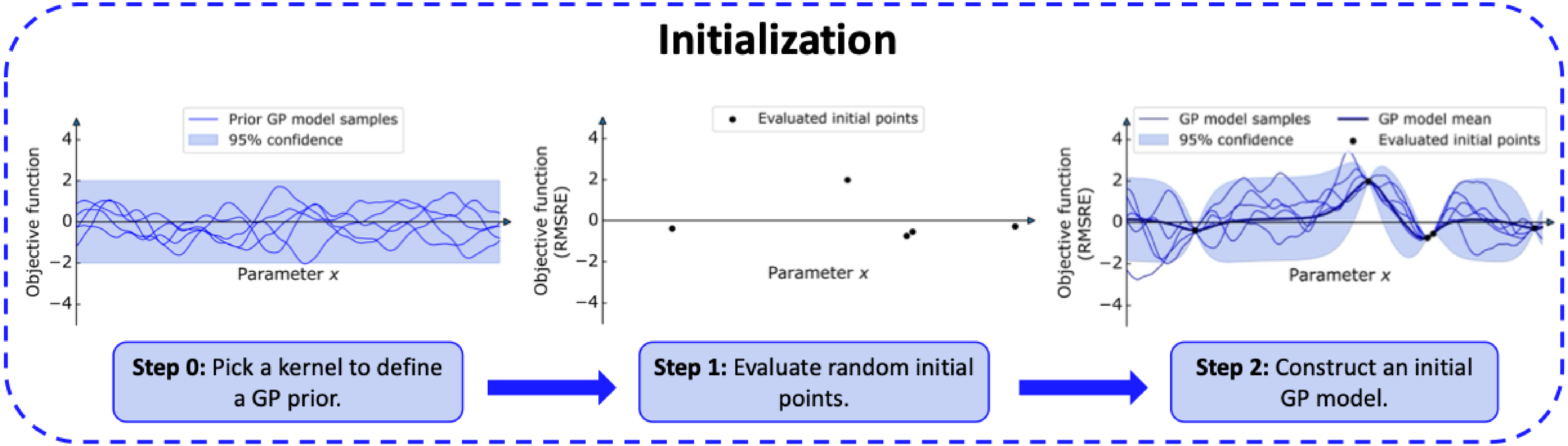
The kernel



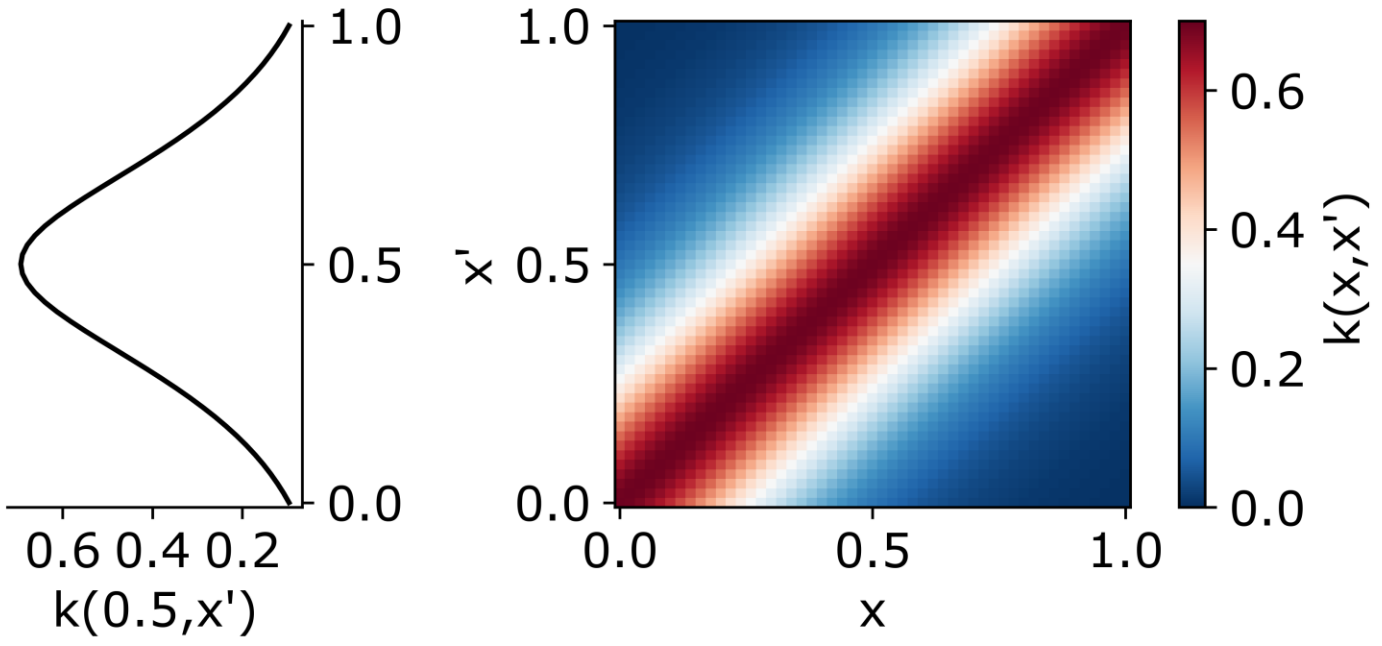


$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right) \qquad \boldsymbol{K} = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}$$

Bayesian Optimization with VerOpt

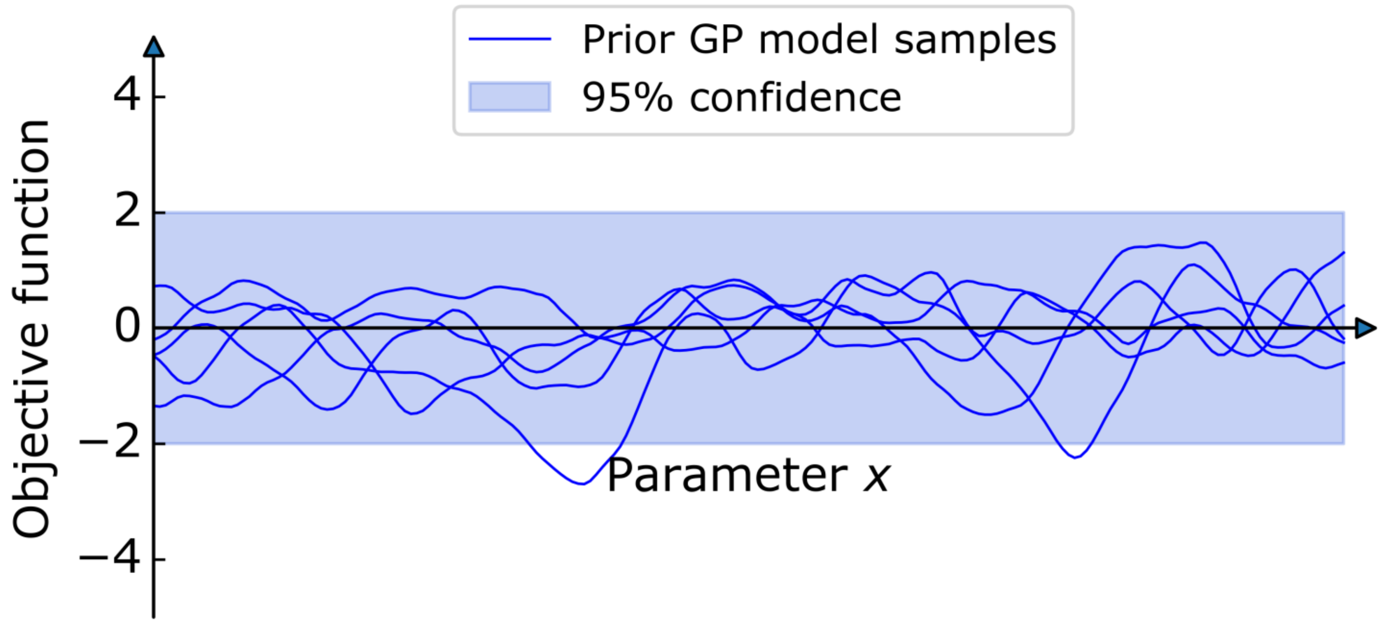

Initialization
Prior GP model samples
95% confidence
Objective function
Parameter x
Evaluated initial points
Objective function (RMSRE)
GP model samples
GP model mean
Step 0: Pick a kernel to define a GP prior.
Step 1: Evaluate random initial points.
Step 2: Construct an initial GP model.

Step 0: Pick a kernel to define GP prior

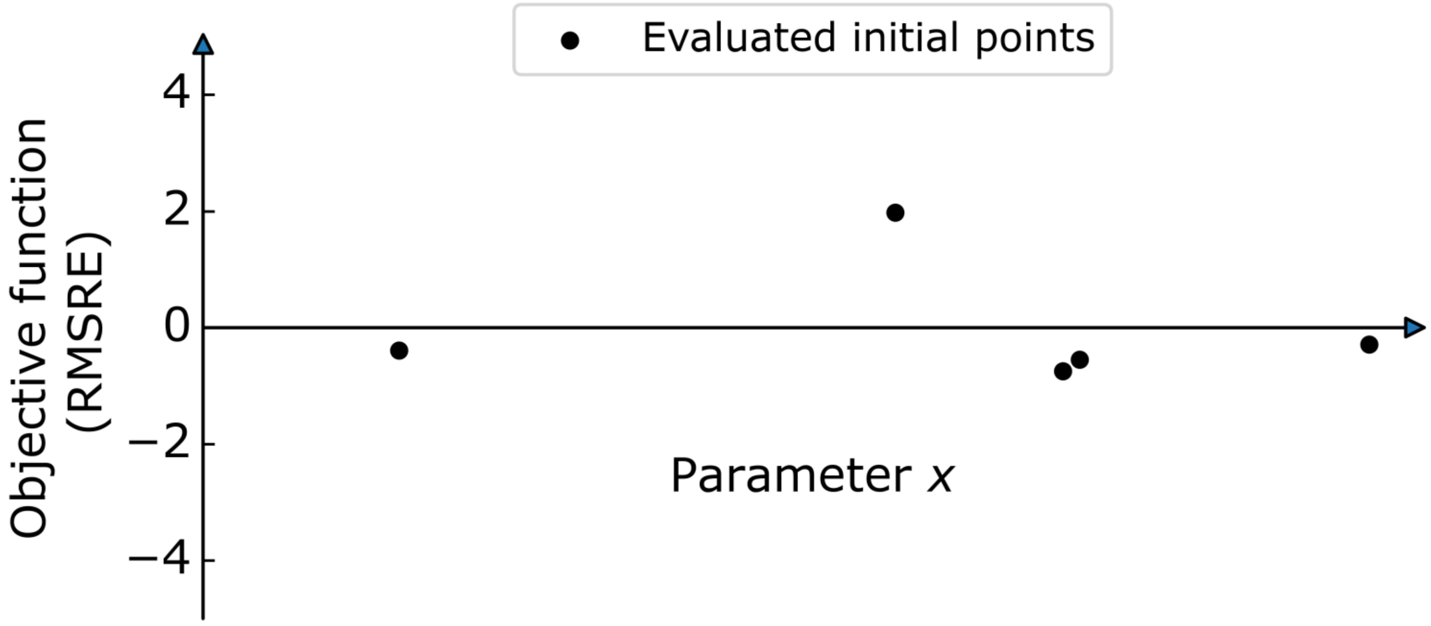


$$C_\nu(d) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{d}{\rho}\right)^\nu K_\nu\left(\sqrt{2\nu}\frac{d}{\rho}\right)$$

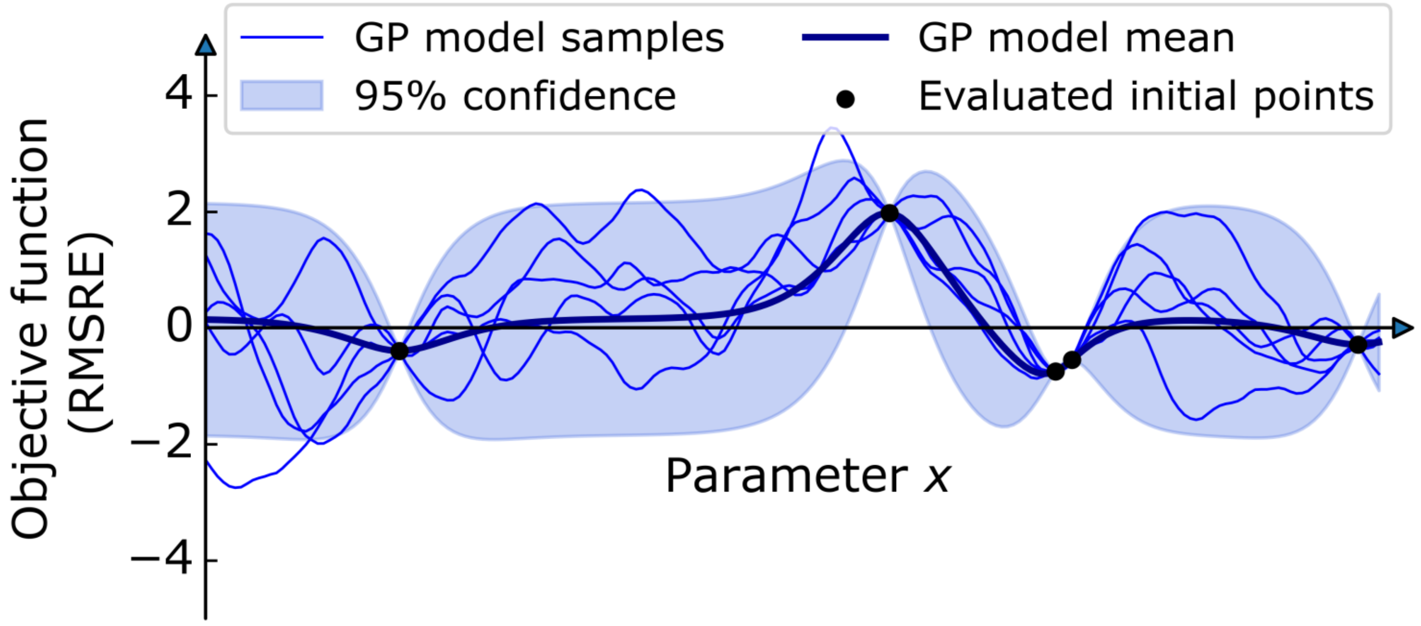
Step 0: Pick a kernel to define GP prior



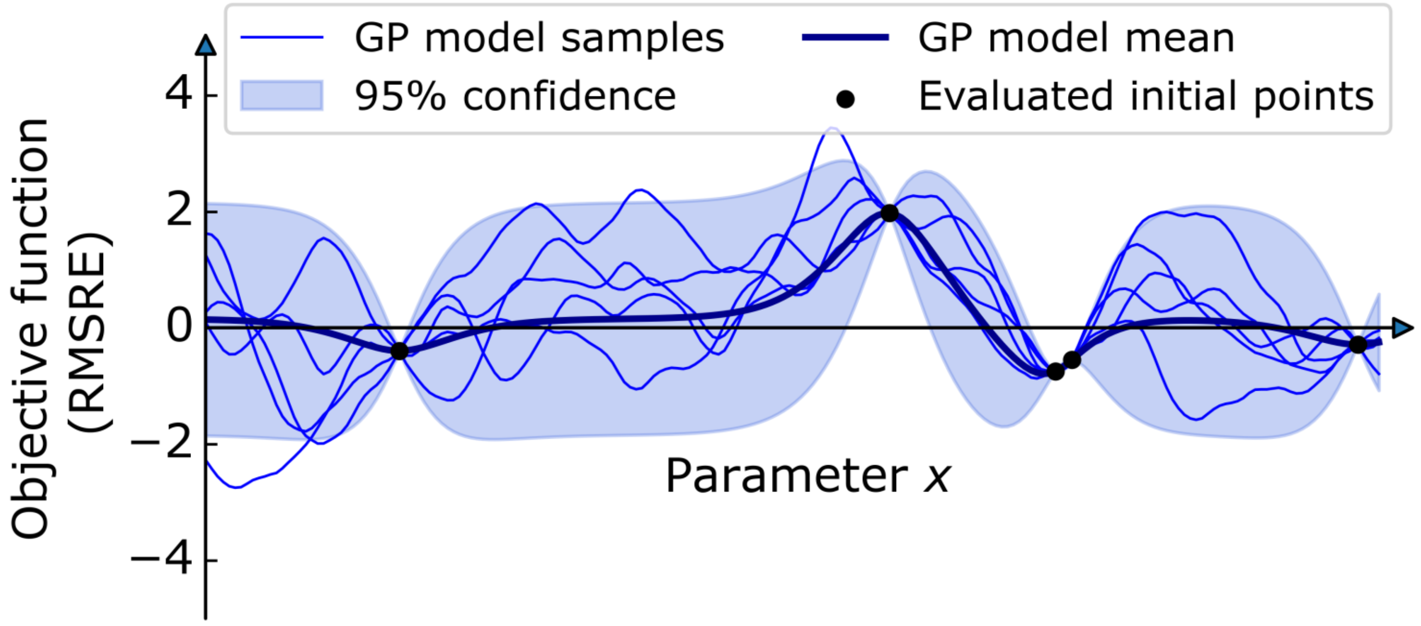
Step 1: Evaluate random initial points



Step 2: Construct an initial GP model

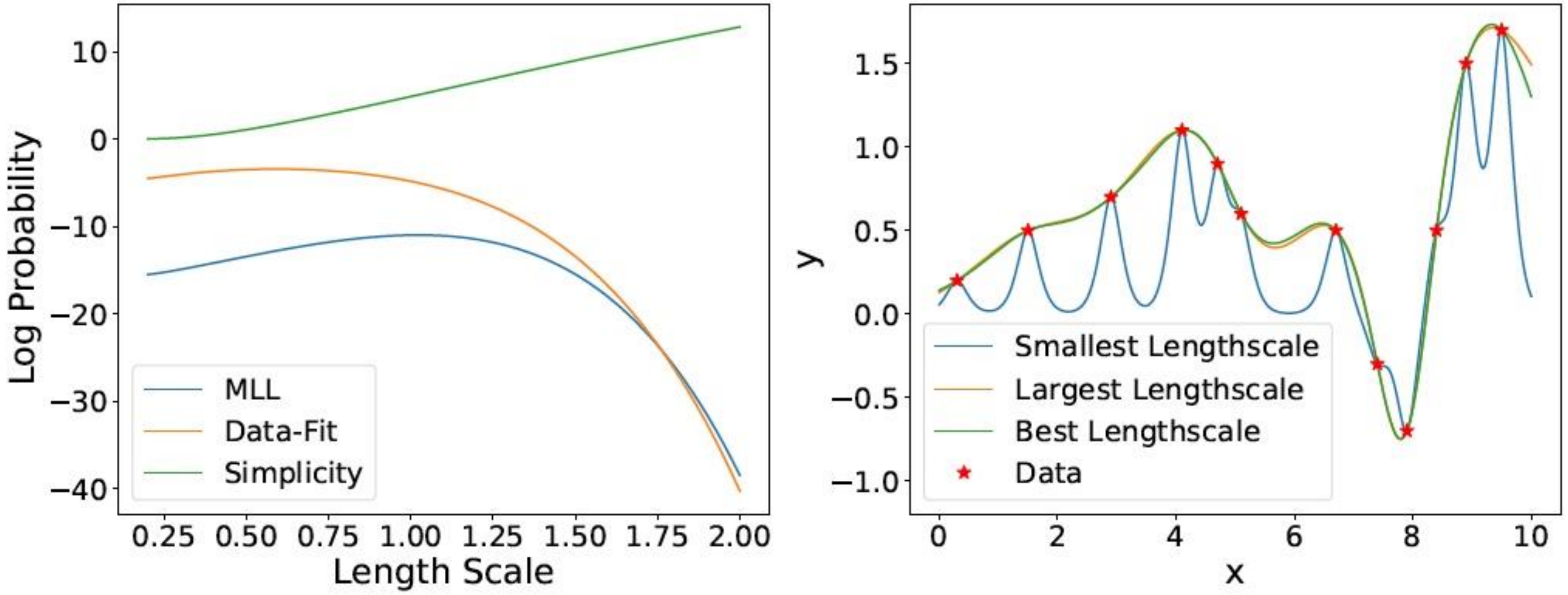


Step 2: Construct an initial GP model



…by minimizing **the log marginal likelihood** with respect to the kernel hyperparameters.

Interlude: Learning the GP model hyperparameter(s)

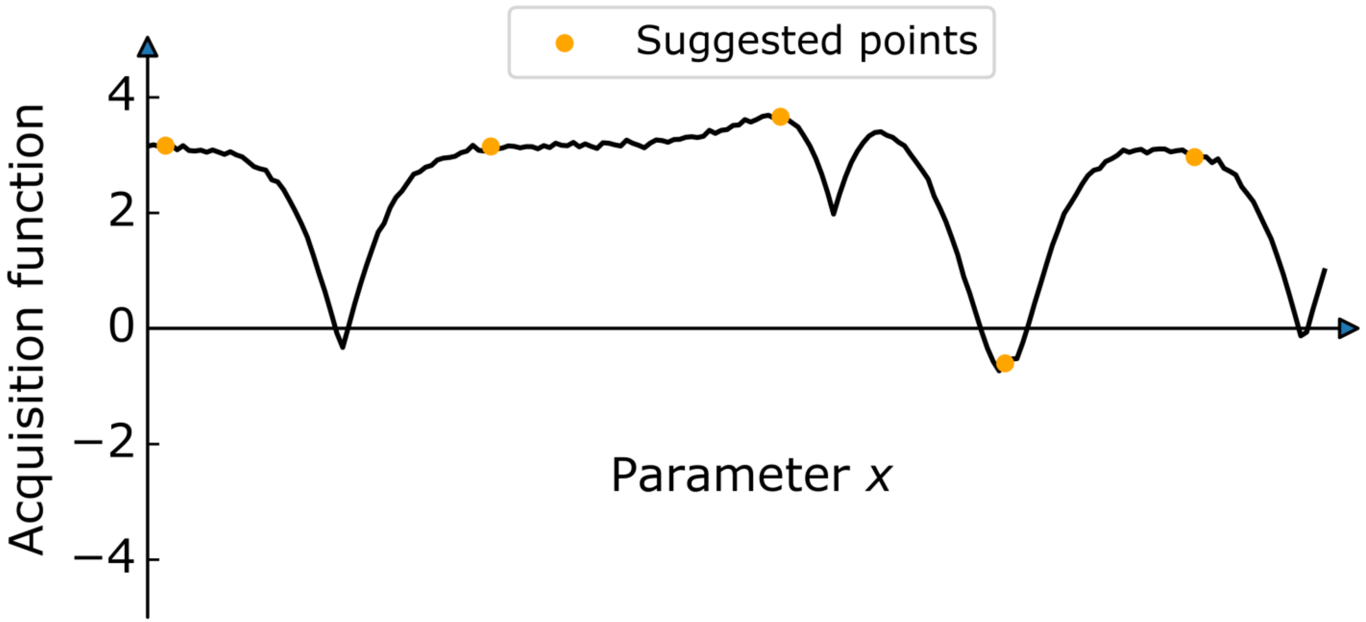


log(MLL) = data fit + simplicity + normalization factor

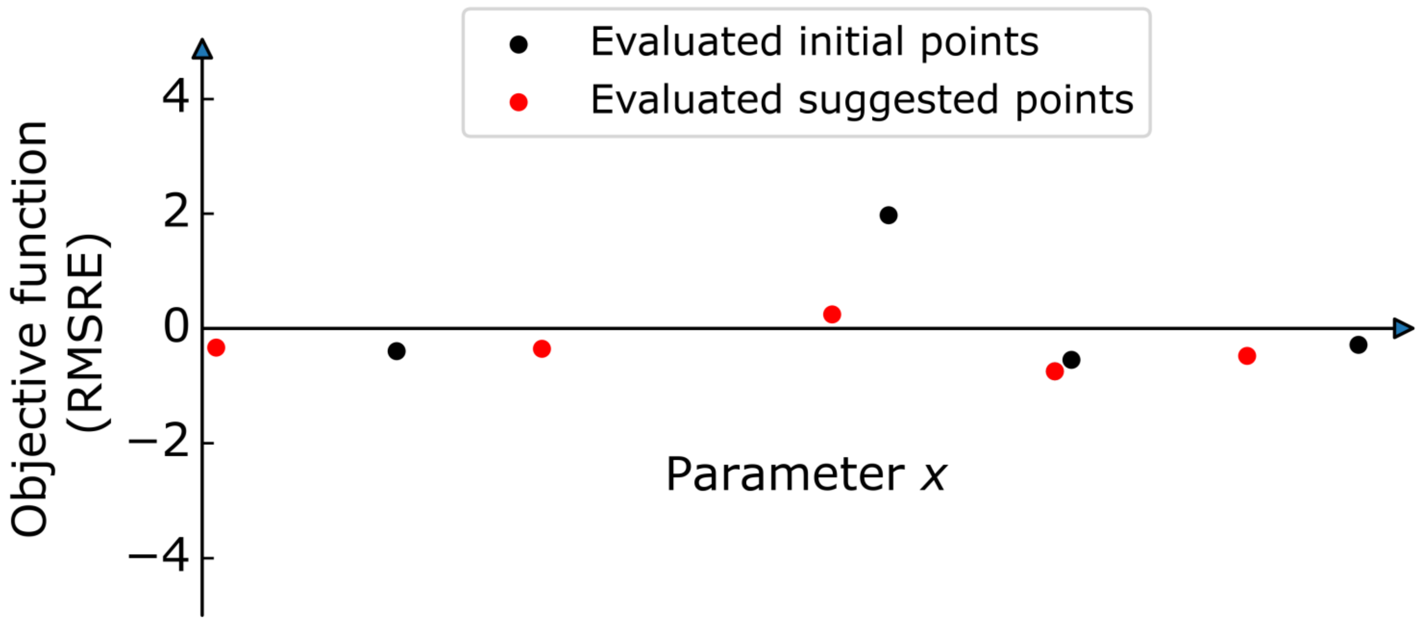
$$\Theta^* = \arg\max_{\Theta} \log p(\mathbf{y} \mid \mathbf{X}, \Theta)$$

Stoustrup (2021)

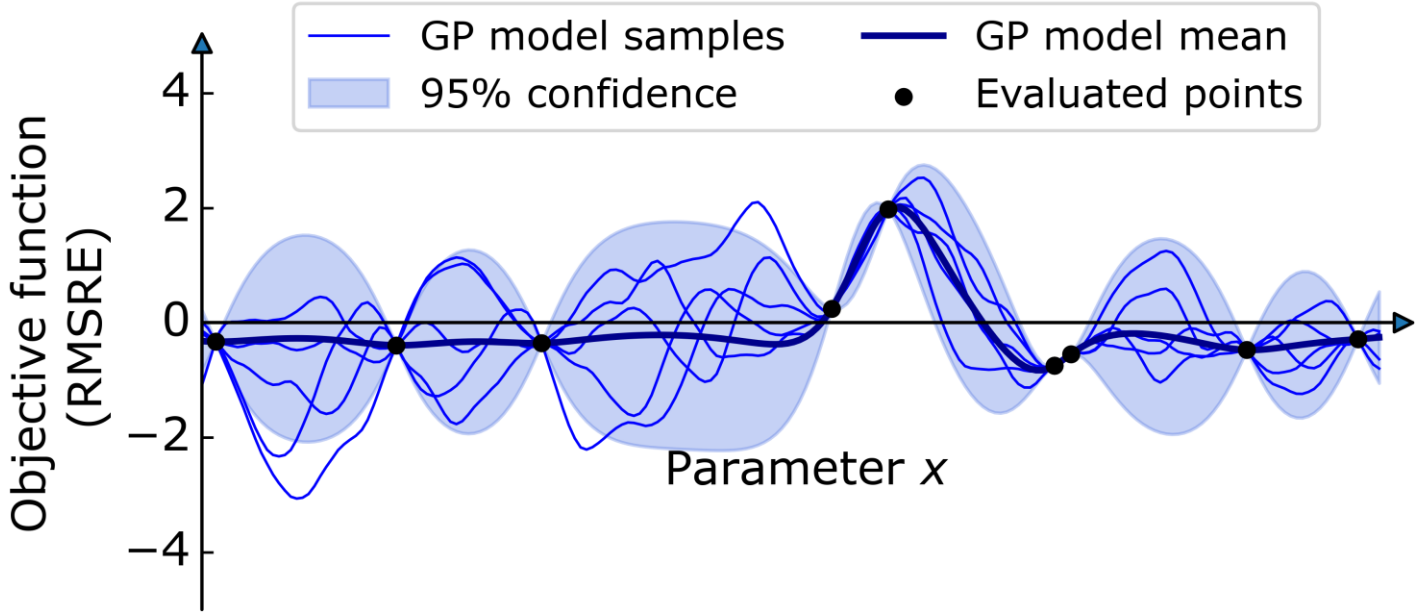
Step 3: Suggest new points to evaluate by optimizing the UCB acquisition function



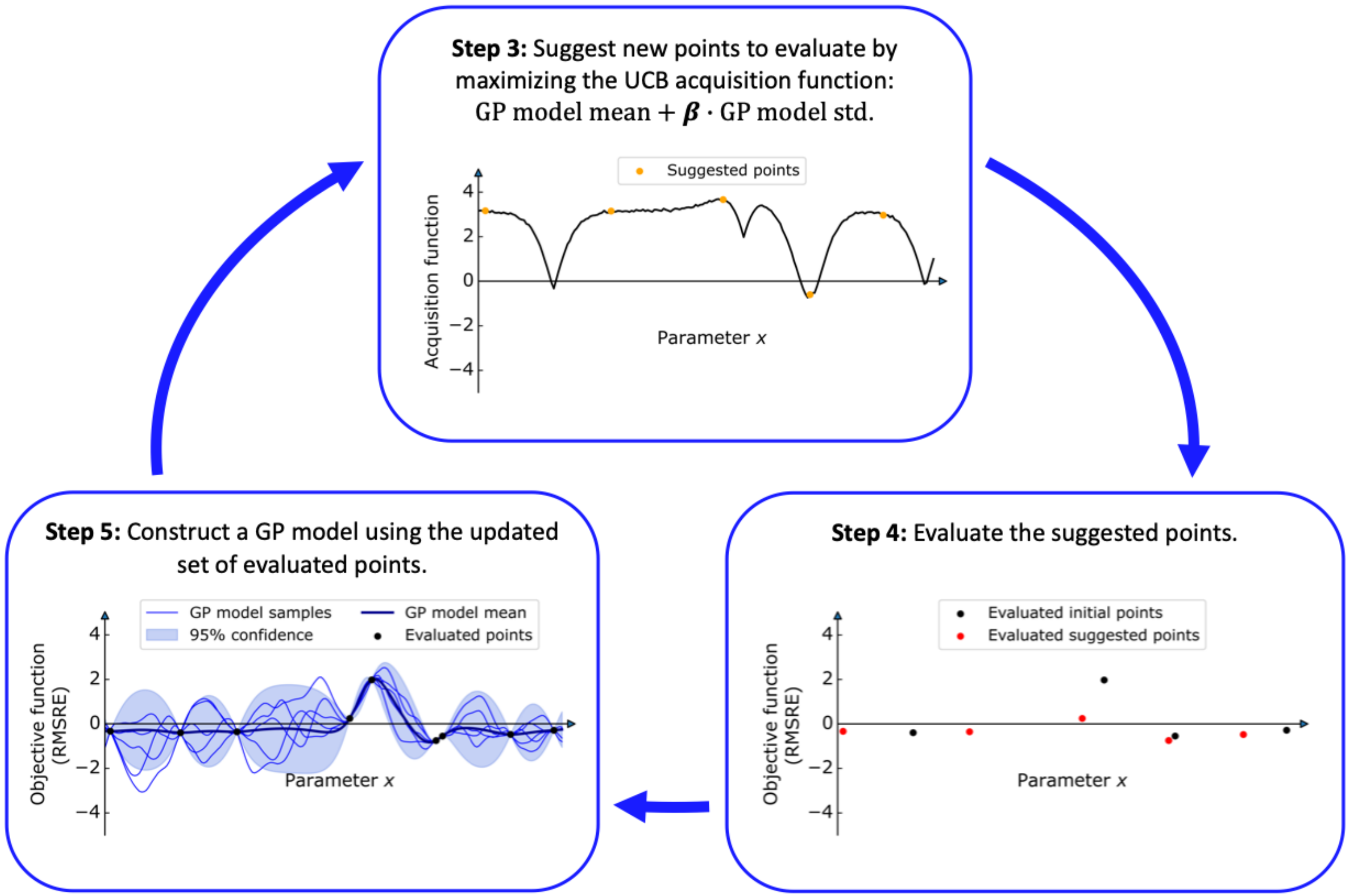
Step 4: Evaluate suggested points



Step 4: Construct a GP model using the updated set of evaluated points



Optimization loop



Optimizing the TKE closure scheme in Veros

Turbulent Kinetic Energy (TKE)

Turbulent fluxes:

$$\overline{-T'w'} = \kappa_h \frac{\partial \overline{T}}{\partial z} \qquad \overline{-\vec{u}_h{}'w'} = \kappa_m \frac{\partial \overline{\vec{u}_h}}{\partial z}$$

Prognostic TKE equation:

$$\frac{\partial E}{\partial t} = \frac{\partial}{\partial z}\left(\kappa_e \frac{\partial E}{\partial z}\right) - c_\epsilon \frac{E^{\frac{3}{2}}}{L} - N^2 \kappa_h + \kappa_m \left(\frac{\partial \overrightarrow{u_h}}{\partial z}\right)^2$$

TKE diffusion TKE dissipation Buoyancy flux Shear production

Parameterization of eddy diffusivities:

$$\kappa_h = c_k L \sqrt{E} \qquad \kappa_e = \alpha_{tke} \kappa_m \qquad \kappa_h = \frac{\kappa_m}{Pr}$$

Free parameters: $\boxed{c_k \in [0,1] \quad c_\epsilon \in [0,1] \quad \alpha_{tke}}$ Default values: $c_k = 0.1,\ c_\epsilon = 0.7,\ \alpha_{tke} = 30$

Gaspar et al. (1990)

MLD optimization

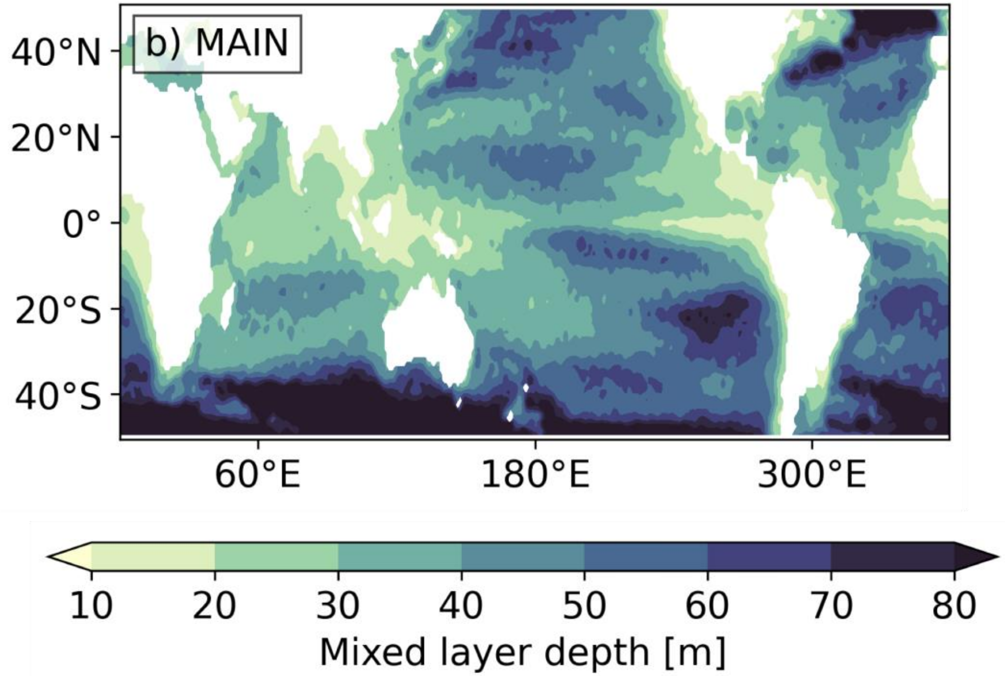
Model:

Veros 1ºx1º
60 vertical layers
2.5m surface resolution
Forced by ECMWF reanalysis
winds (ie observations
assimilated into numerical model

Setup:

Duration of simulation: 30 years
Initial points: 10
Bayes points: 30
Evaluations per step: 2

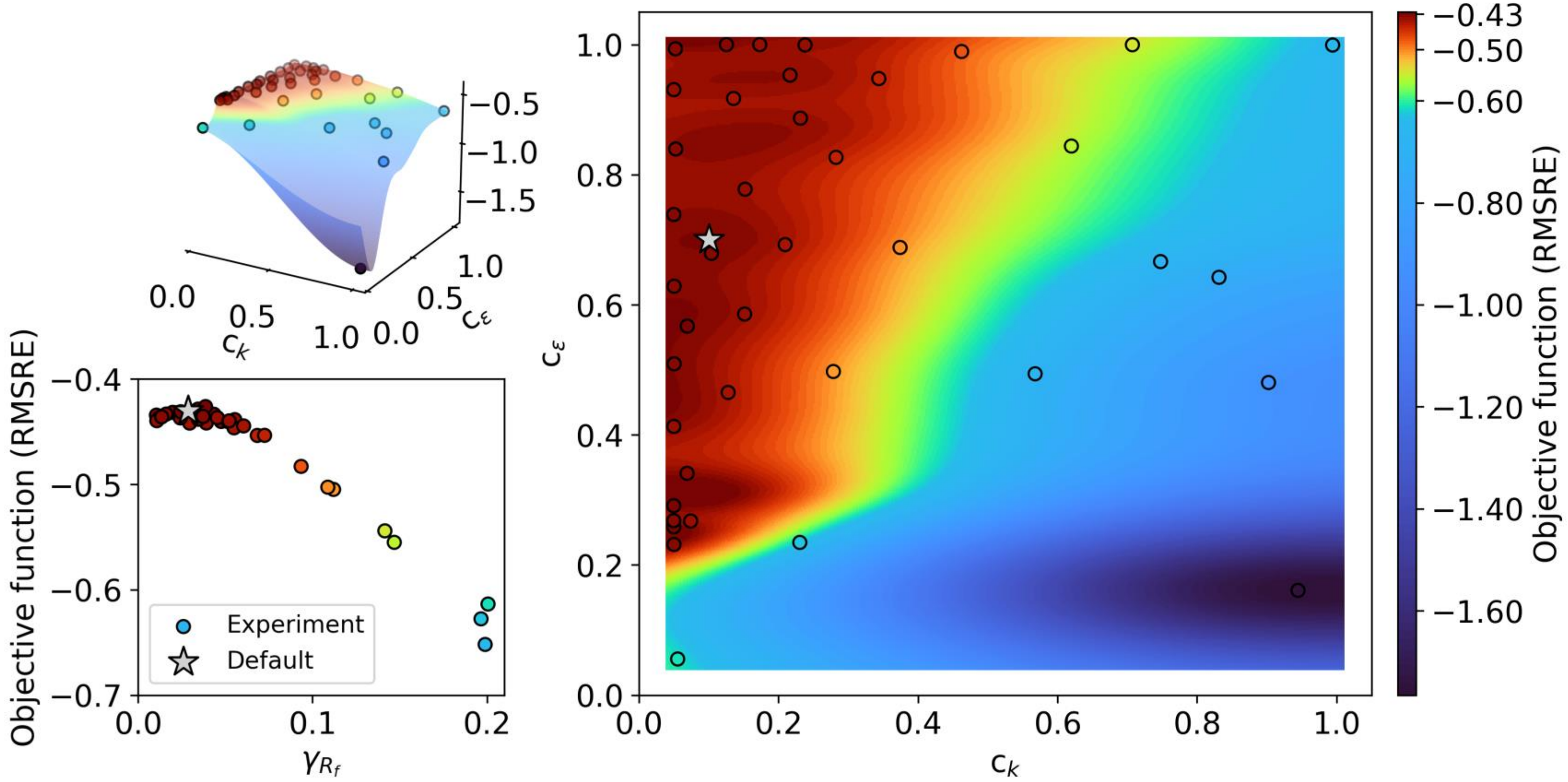
Objective function: $$-\sqrt{\frac{1}{NM}\sum_{i,j=0,0}^{N,M}\left(\frac{MLD_{i,j}^{sim}-MLD_{i,j}^{obs}}{MLD_{i,j}^{obs}}\right)^2}$$



ERA-Interim: Dee et al. (2011); Ifremer MLD: De Boyer Montégut et al. (2022)

Optimization results

The default TKE parameterization lays within the parameter space region where the MLD bias is minimized.

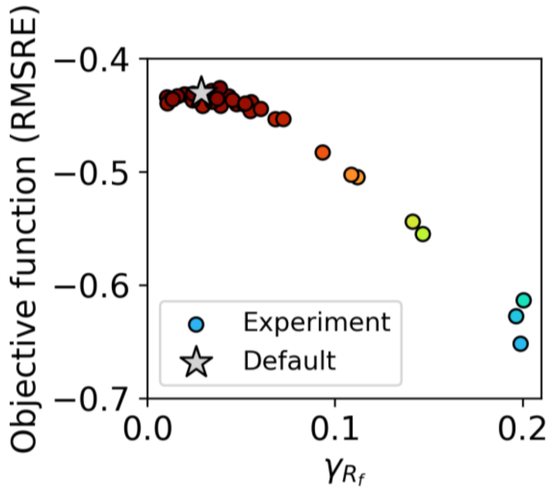


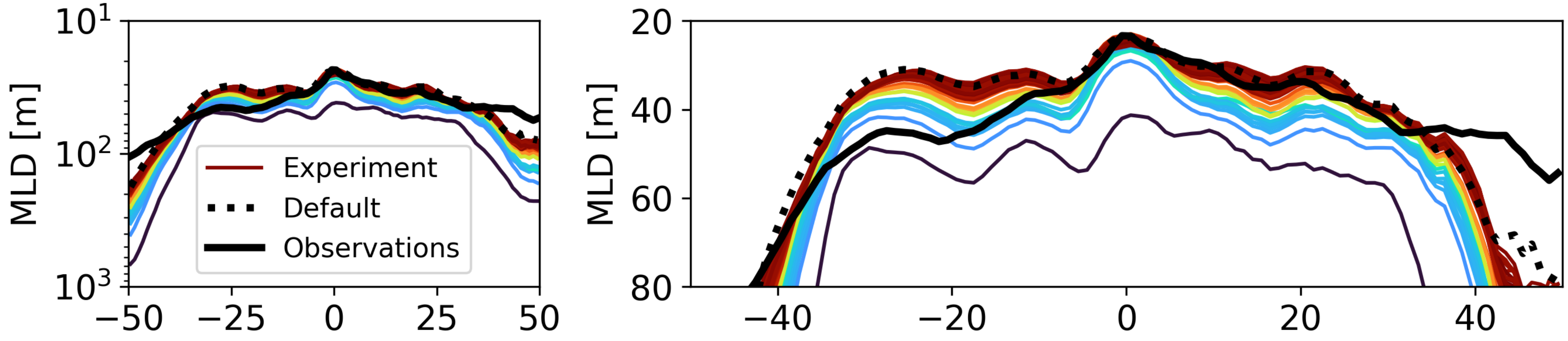
Optimization results

The default TKE parameterization lays within the parameter space region where the MLD bias is minimized.

$$\gamma_{R_f} = \frac{R_f}{1 - R_f}$$

(lab experiments suggest 0.05-0.2),
Ie the amount of energy converted
To potential energy and not heat)





Why use Bayesian optimization?

- The method is transparent (not a black box)
- Does not rely on gradients
- Relatively few objective function evaluations are needed
- Easy to build with Python packages such as PyTorch
- Has just last week been ported to LUMI to optimize a 9-d parameter space …
- … on 1000 GPUs!

Thank you for your attention!

Supported by EU project NextGEMS, the LUMI consortium and the Danish Center of Climate Computing at KU