Probabilistic data fusion and physics-informed machine learning

Paris Perdikaris
Department of Mechanical Engineering and Applied Mechanics
University of Pennsylvania
email: pgp@seas.upenn.edu

IPAM workshop
Big Data Meets Large-Scale Computing
September 28th, 2018

U.S. DEPARTMENT OF ENERGY
INCITE
LEADERSHIP COMPUTING

BROWN

Argonne
NATIONAL LABORATORY

JÜLICH
FORSCHUNGSZENTRUM

*Simulation credit: Leopold Grinberg (IBM)

$\mathcal{O}(10^6)$ lines of code, ~20 PhD theses (1990-2010)

32k supercomputer nodes (IBM BG/P, 128k processors)

Finalist for the Gordon-Bell prize in supercomputing (2011, only US entry)

Simulation of one cardiac cycle took ~24hr

4 MW at $0.10/kWh is $400 an hour or about $3.5 million per year.

U.S. DEPARTMENT OF ENERGY
INCITE
LEADERSHIP COMPUTING

BROWN

Argonne
NATIONAL LABORATORY

JÜLICH
FORSCHUNGSZENTRUM

*Simulation credit: Leopold Grinberg (IBM)

Impressive computation, but still an approximation to the true underlying physics

Validity of results relies on accurate model calibration

Many sources of uncertainty: geometry, IC/BCs, rheology, material properties, etc.

$\mathcal{O}(10 - 10^2)$ parameters to be calibrated using clinical data

U.S. DEPARTMENT OF ENERGY
INCITE
LEADERSHIP COMPUTING

BROWN

Argonne
NATIONAL LABORATORY

JÜLICH
FORSCHUNGSZENTRUM

*Simulation credit: Leopold Grinberg (IBM)

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta), \quad \mathcal{L}(\theta) := ||m_\theta(\boldsymbol{x}) - \boldsymbol{t}||$$

θ : Model parameters
x : Model inputs
t : Target quantities of interest

- High dimensional optimization
- Expensive, black-box loss function
- No gradients available

U.S. DEPARTMENT OF ENERGY
INCITE
LEADERSHIP COMPUTING

BROWN

Argonne
NATIONAL LABORATORY

JÜLICH
FORSCHUNGSZENTRUM

*Simulation credit: Leopold Grinberg (IBM)

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta), \quad \mathcal{L}(\theta) : ||m_\theta(\boldsymbol{x}) - \boldsymbol{t}||$$

Big data vs small data → Dimensionality vs # of observations

U.S. DEPARTMENT OF ENERGY
INCITE
LEADERSHIP COMPUTING

BROWN

Argonne
NATIONAL LABORATORY

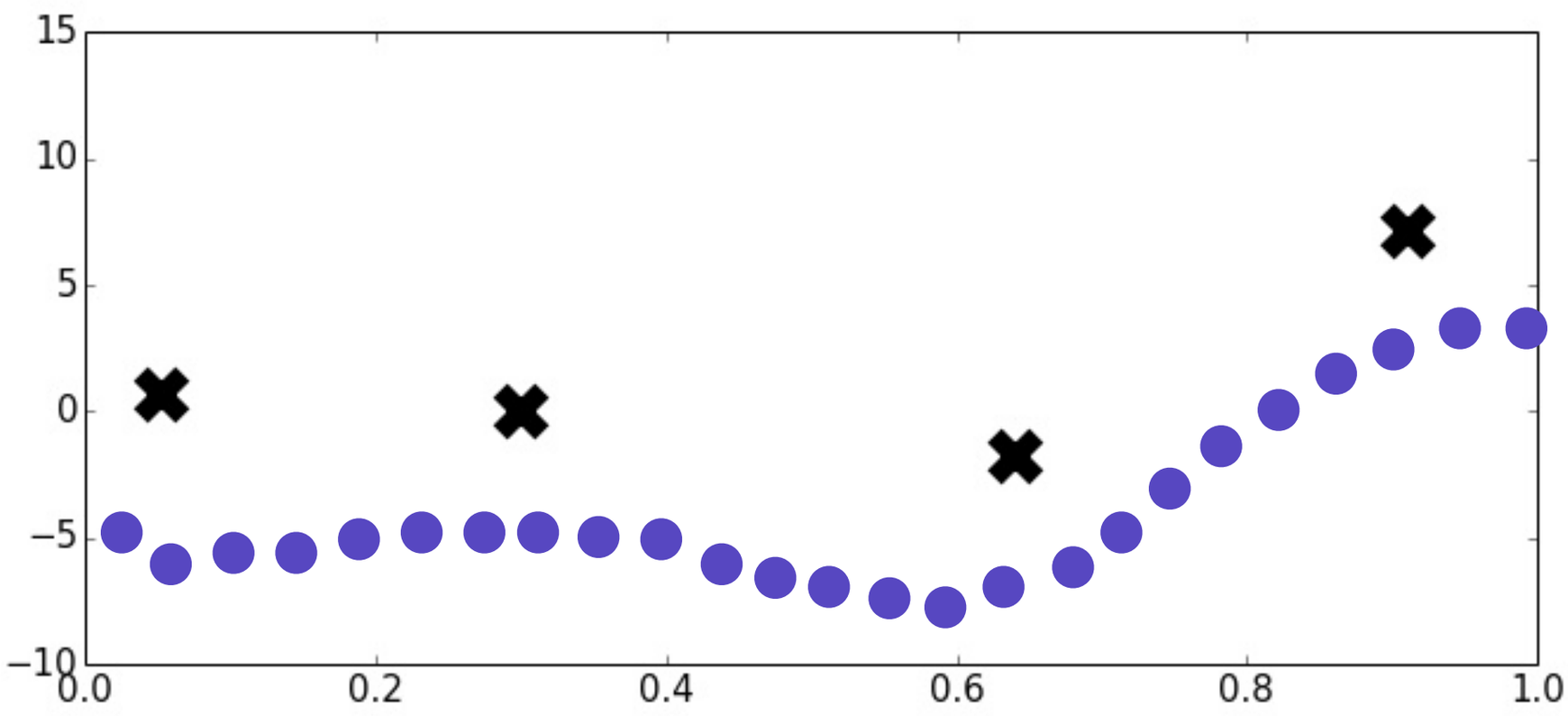
JÜLICH
FORSCHUNGSZENTRUM

*Simulation credit: Leopold Grinberg (IBM)

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta), \quad \mathcal{L}(\theta) : ||m_\theta(\boldsymbol{x}) - \boldsymbol{t}||$$

Can we solve this problem using surrogate models that are cheaper to compute?

Optimization of expensive black-box functions



Where is the minimum of $\mathcal{L}(\theta)$?

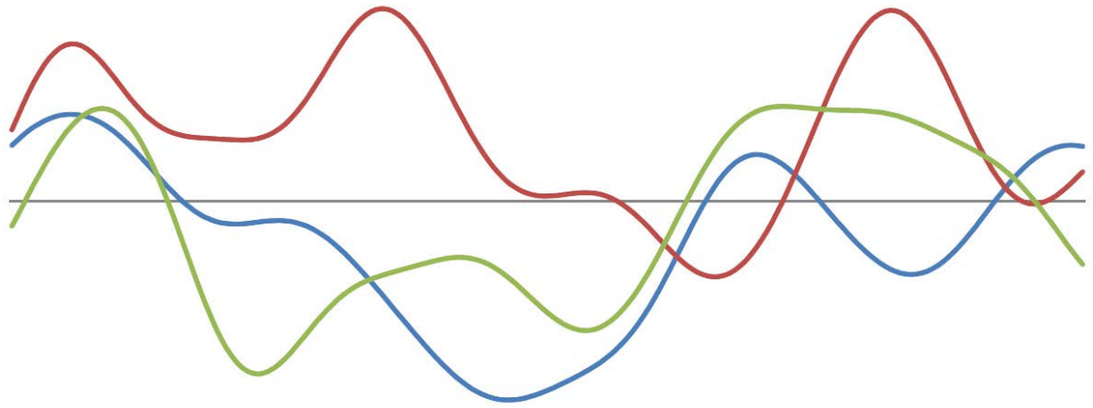
Where should we take our next evaluation?

How can we distill useful information from simplified models to accelerate this process?

Data-driven modeling with Gaussian processes

Priors over functions:

$$f \sim \mathcal{GP}(\mu(x), K(\mathbf{x}, \mathbf{x}'; \theta))$$

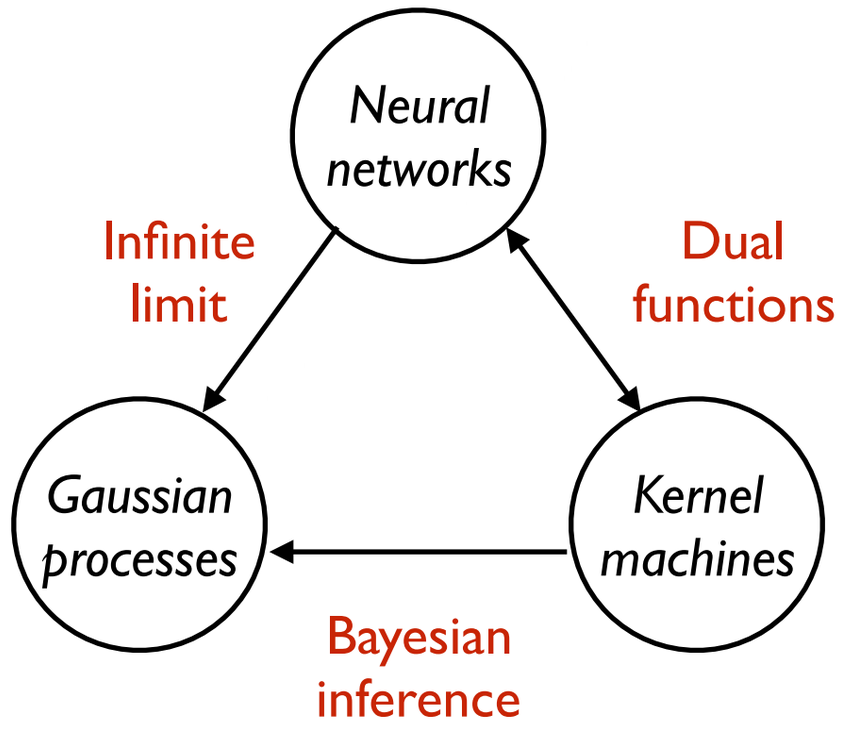


Samples from a GP prior

Marginalization:

$p(\mathbf{f}_A, \mathbf{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$. Then:

$$p(\mathbf{f}_A) = \int_{\mathbf{f}_B} p(\mathbf{f}_A, \mathbf{f}_B) \mathrm{d}\mathbf{f}_B = \mathcal{N}(\boldsymbol{\mu}_A, \mathbf{K}_{AA})$$



Posterior is also Gaussian:

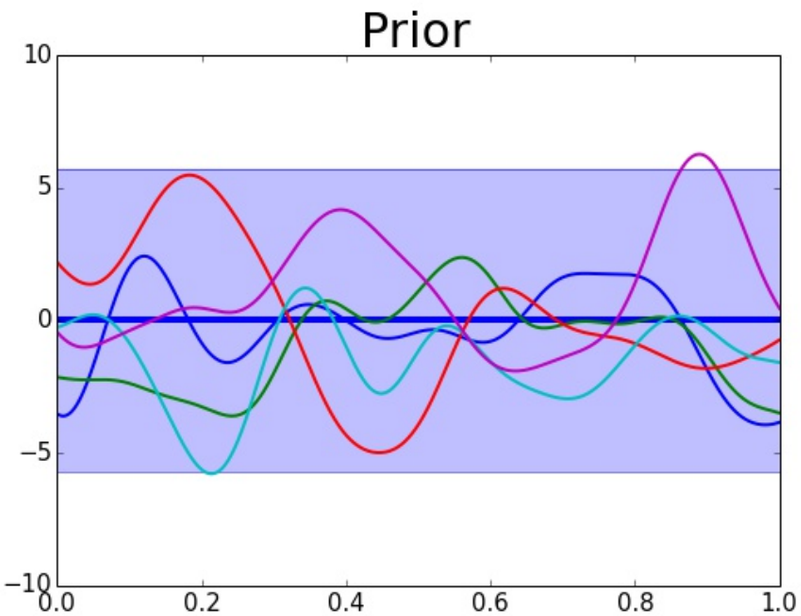
$p(\mathbf{f}_A, \mathbf{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$. Then:

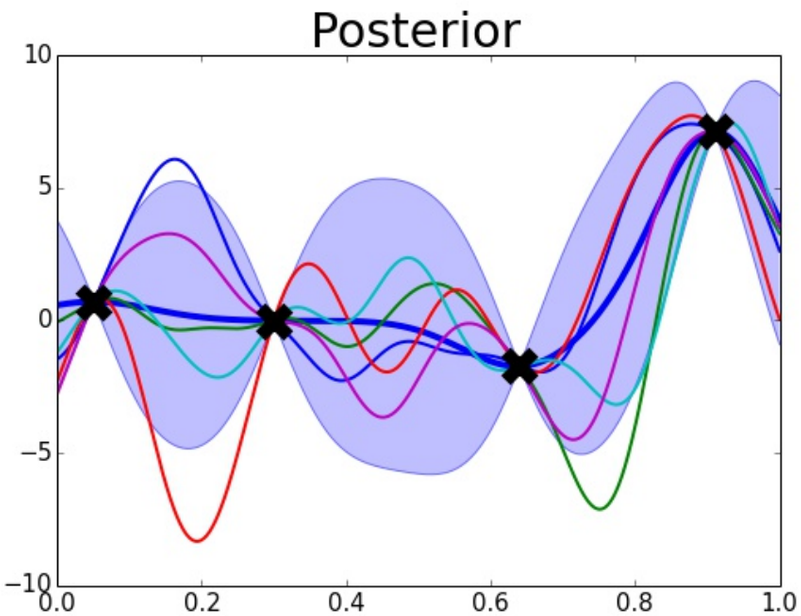
$$p(\mathbf{f}_A|\mathbf{f}_B) = \mathcal{N}(\boldsymbol{\mu}_A + \mathbf{K}_{AB}\mathbf{K}_{BB}^{-1}(\mathbf{f}_B - \boldsymbol{\mu}_B), \mathbf{K}_{AA} - \mathbf{K}_{AB}\mathbf{K}_{BB}^{-1}\mathbf{K}_{BA})$$

Rasmussen, C.E. and Williams C. Gaussian processes for machine learning (2006)

Data-driven modeling with Gaussian processes

$$y = f(\boldsymbol{x}) + \epsilon \qquad f \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}))$$





Training via maximizing the marginal likelihood

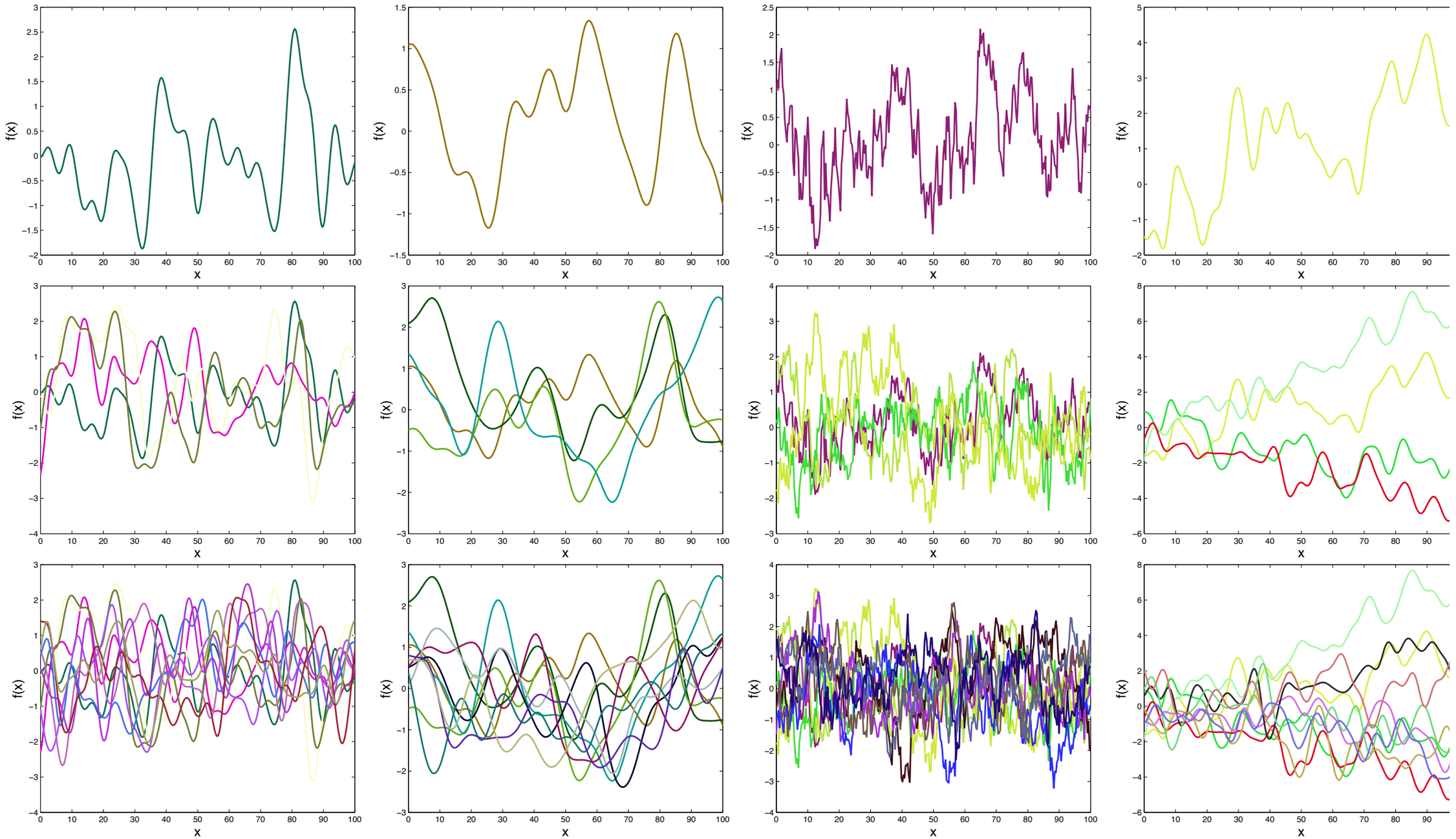
$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\log|\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y} - \frac{N}{2}\log 2\pi$$

Prediction via conditioning on available data

$$p(f_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*) = \mathcal{N}(f_*|\mu_*, \sigma_*^2),$$
$$\mu_*(\boldsymbol{x}_*) = \boldsymbol{k}_{*N}(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y},$$
$$\sigma_*^2(\boldsymbol{x}_*) = \boldsymbol{k}_{**} - \boldsymbol{k}_{*N}(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{k}_{N*},$$

Demo code: https://github.com/PredictiveIntelligenceLab/GPTutorial

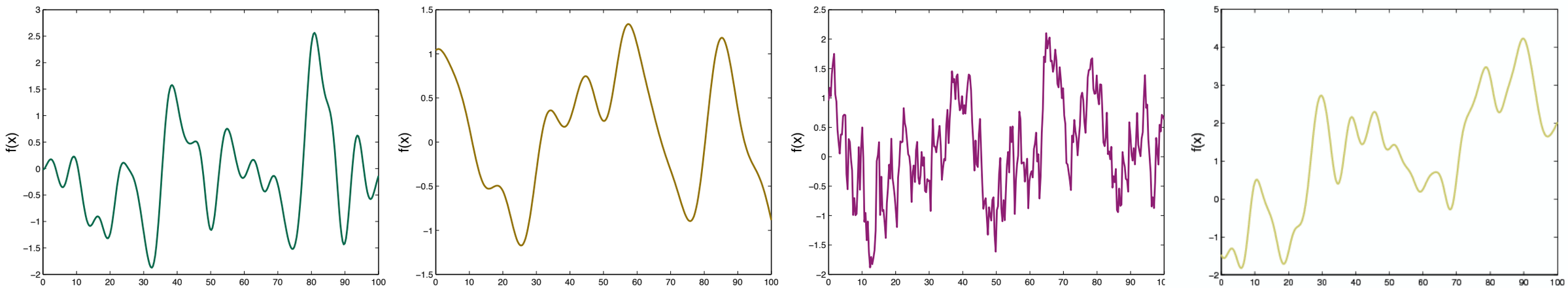
Samples from GPs with different $K(x, x')$



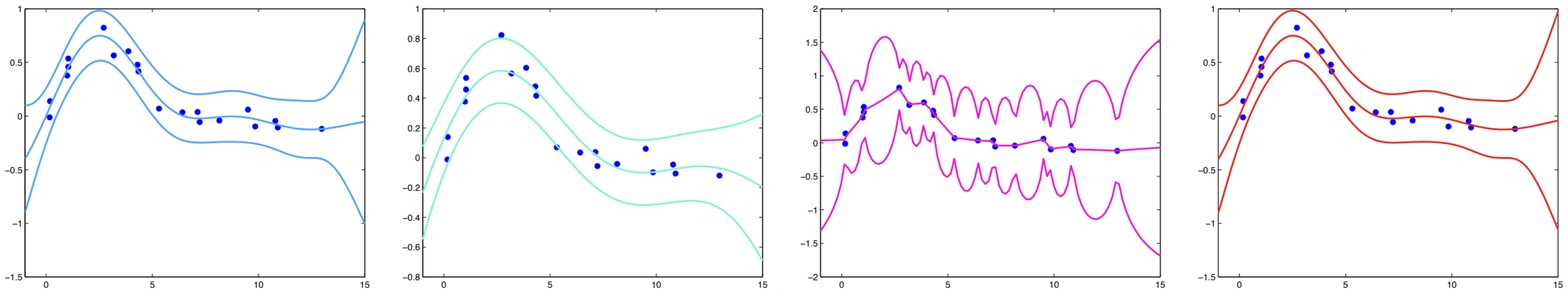
http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Prediction using GPs with different $K(x, x')$

A sample from the prior for each covariance function:

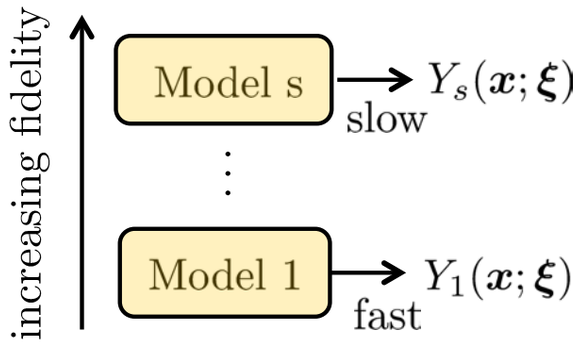


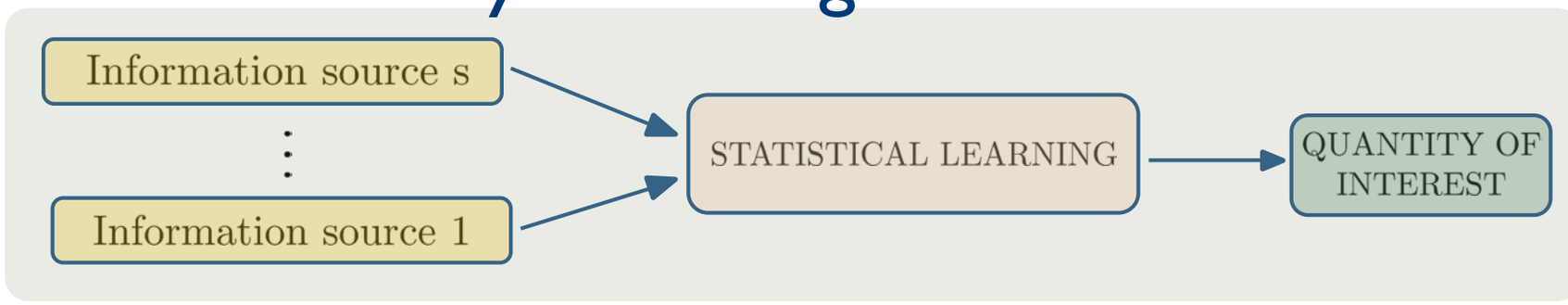
Corresponding predictions, mean with two standard deviations:



http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Multi-fidelity modeling





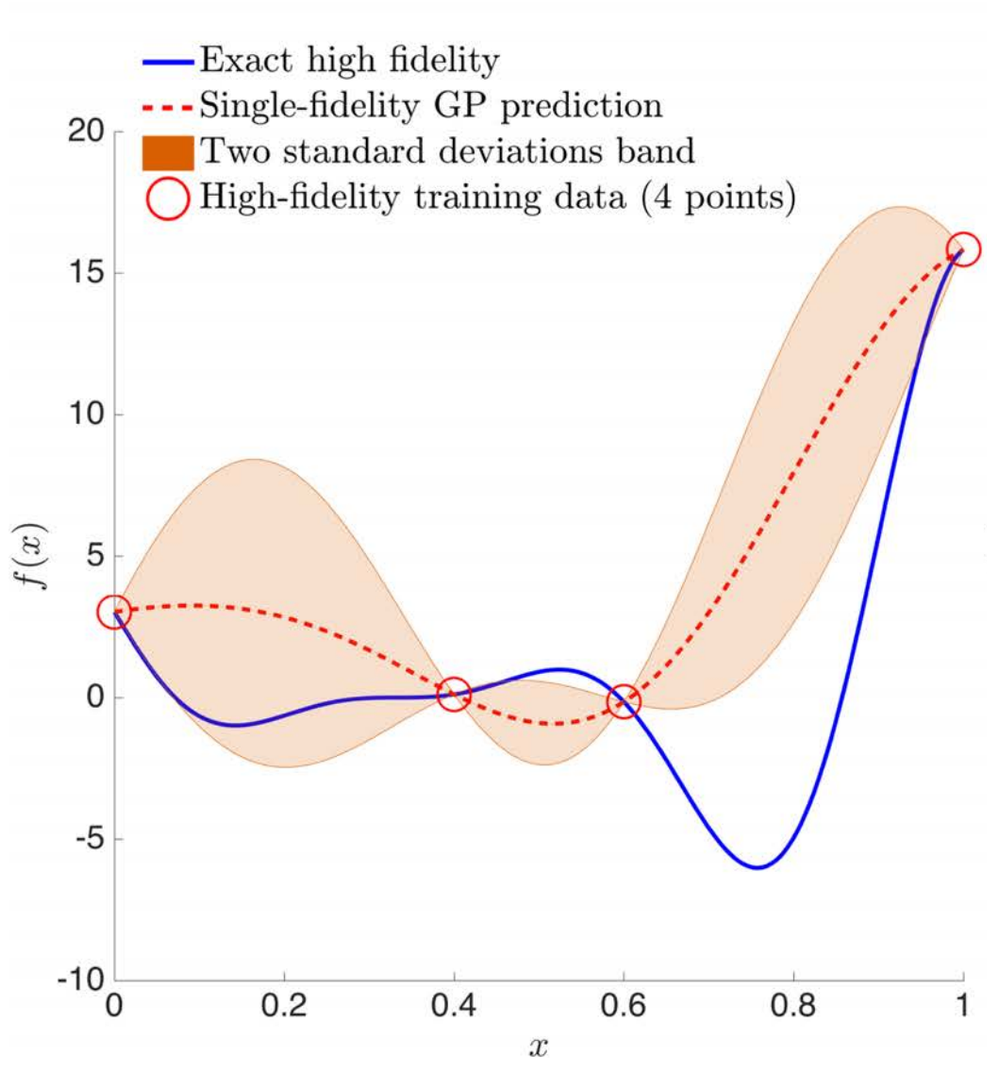
Number of runs is limited by time and computational resources

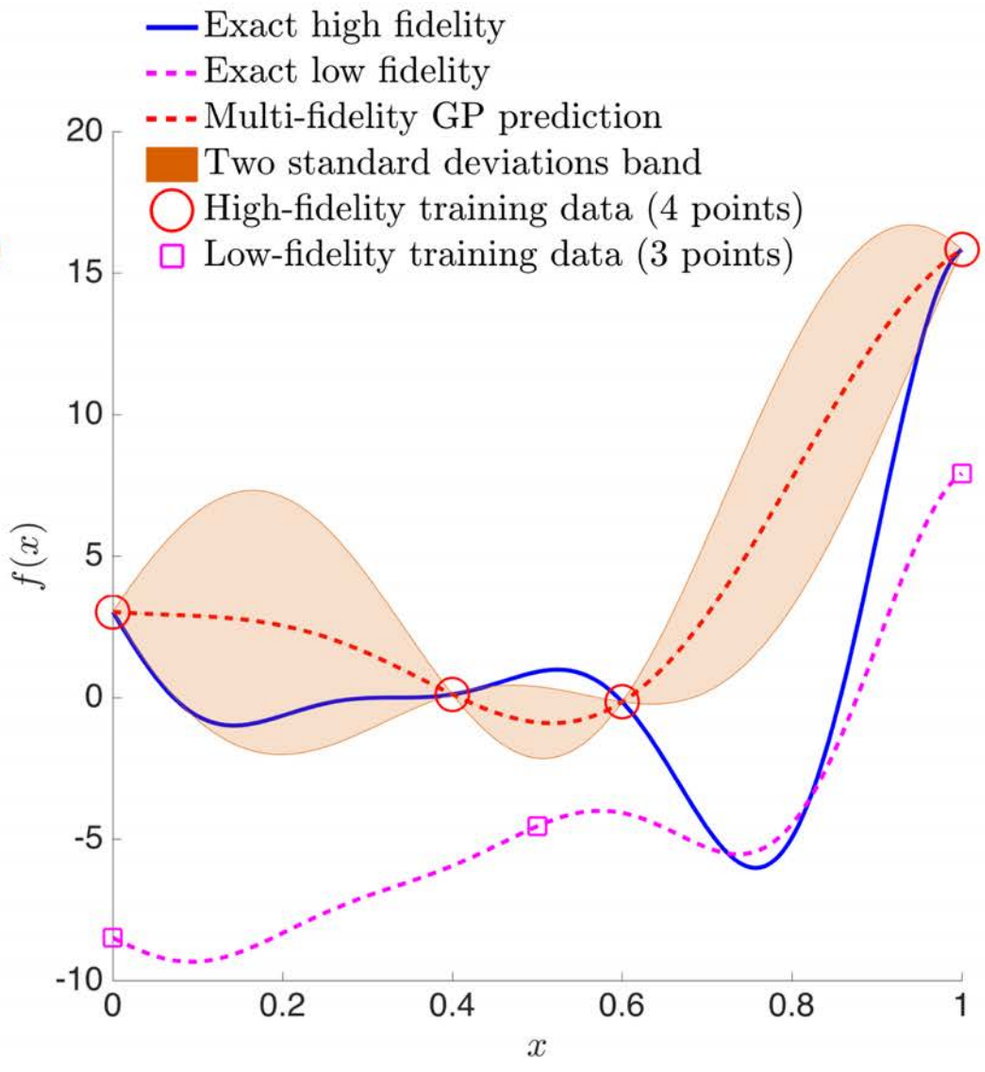
⇓

We cannot compute at all $(\boldsymbol{x};\boldsymbol{\xi})$

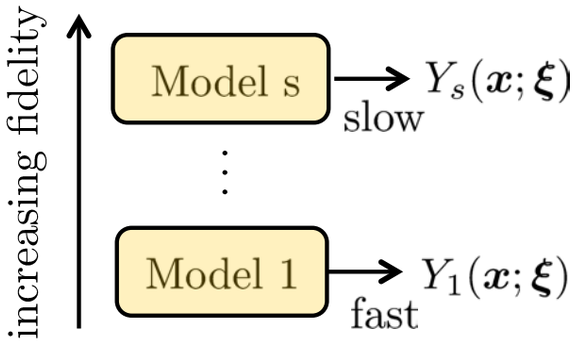
⇓

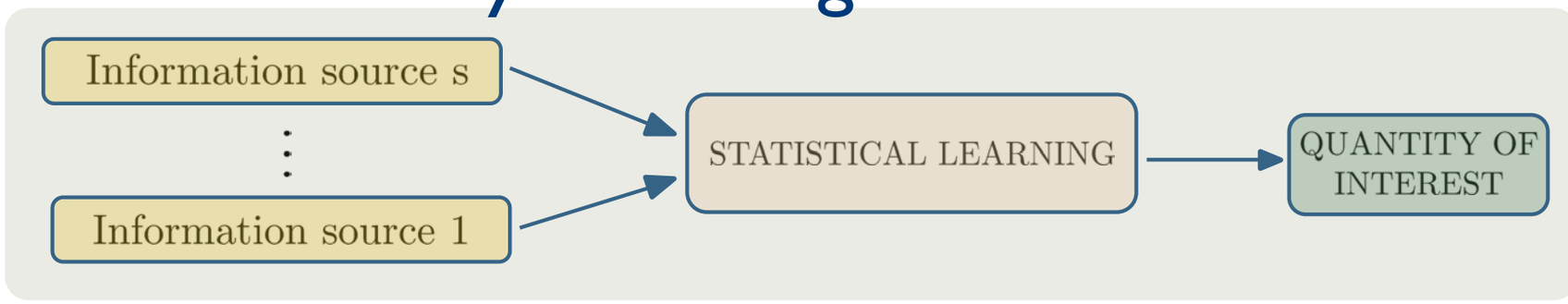
Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling





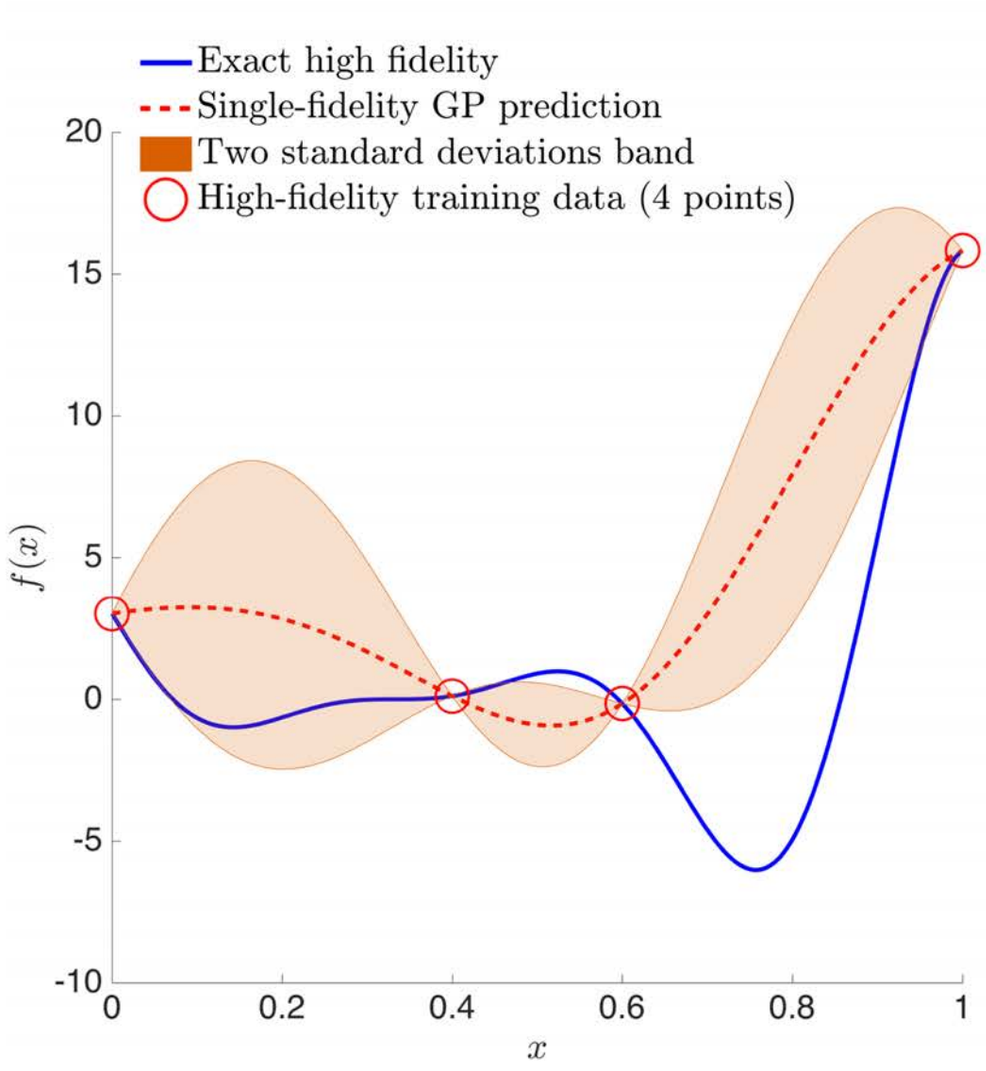
Number of runs is limited by time and computational resources

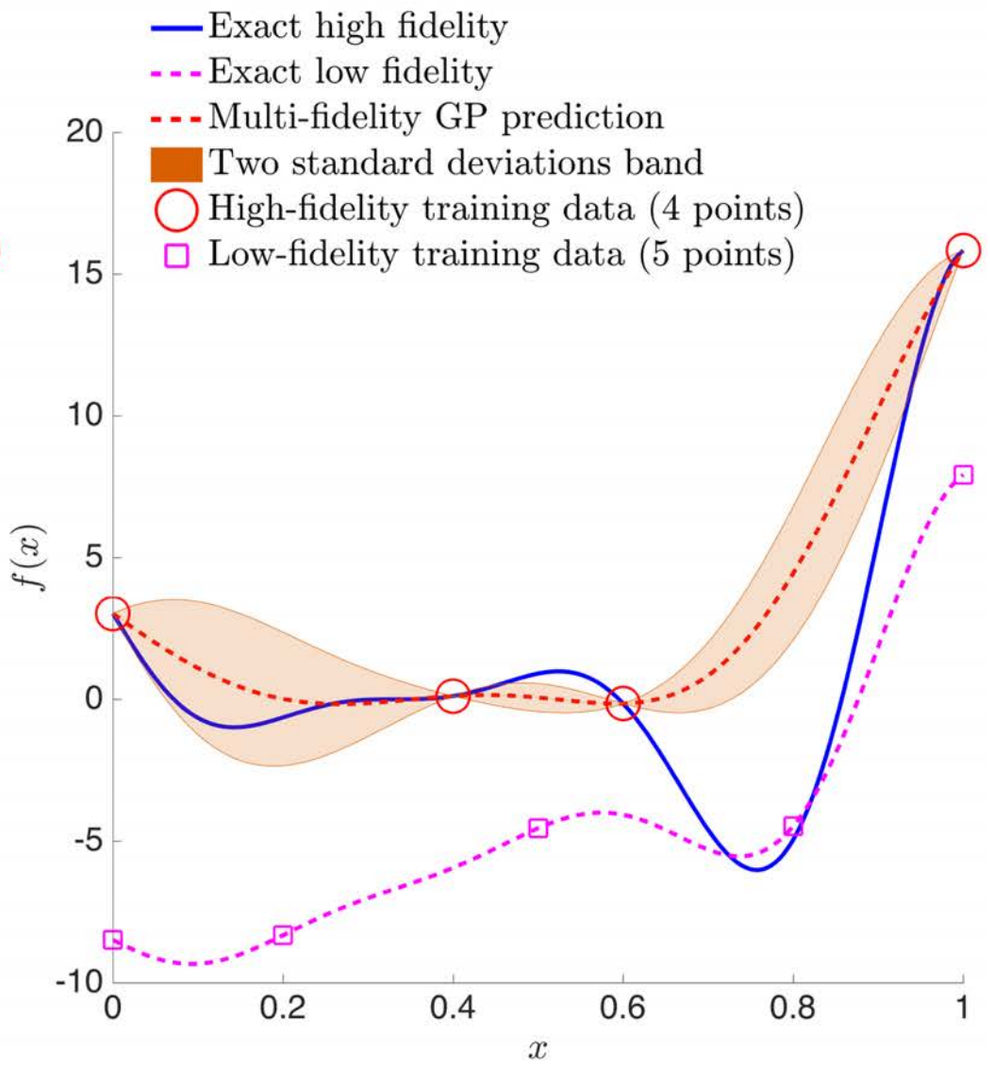
⇓

We cannot compute at all $(\boldsymbol{x};\boldsymbol{\xi})$

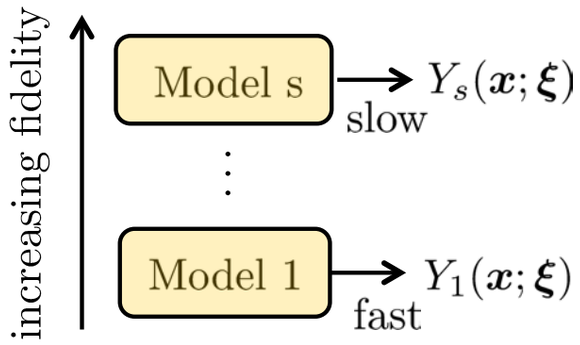
⇓

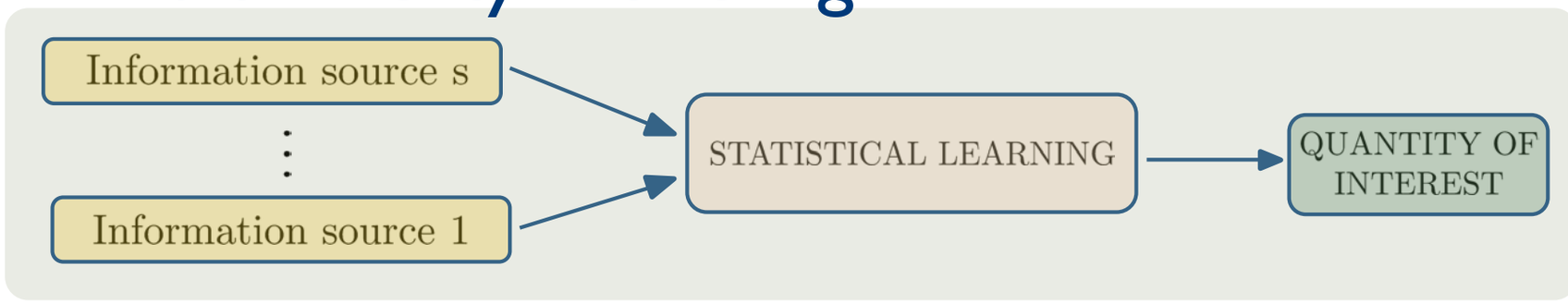
Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling





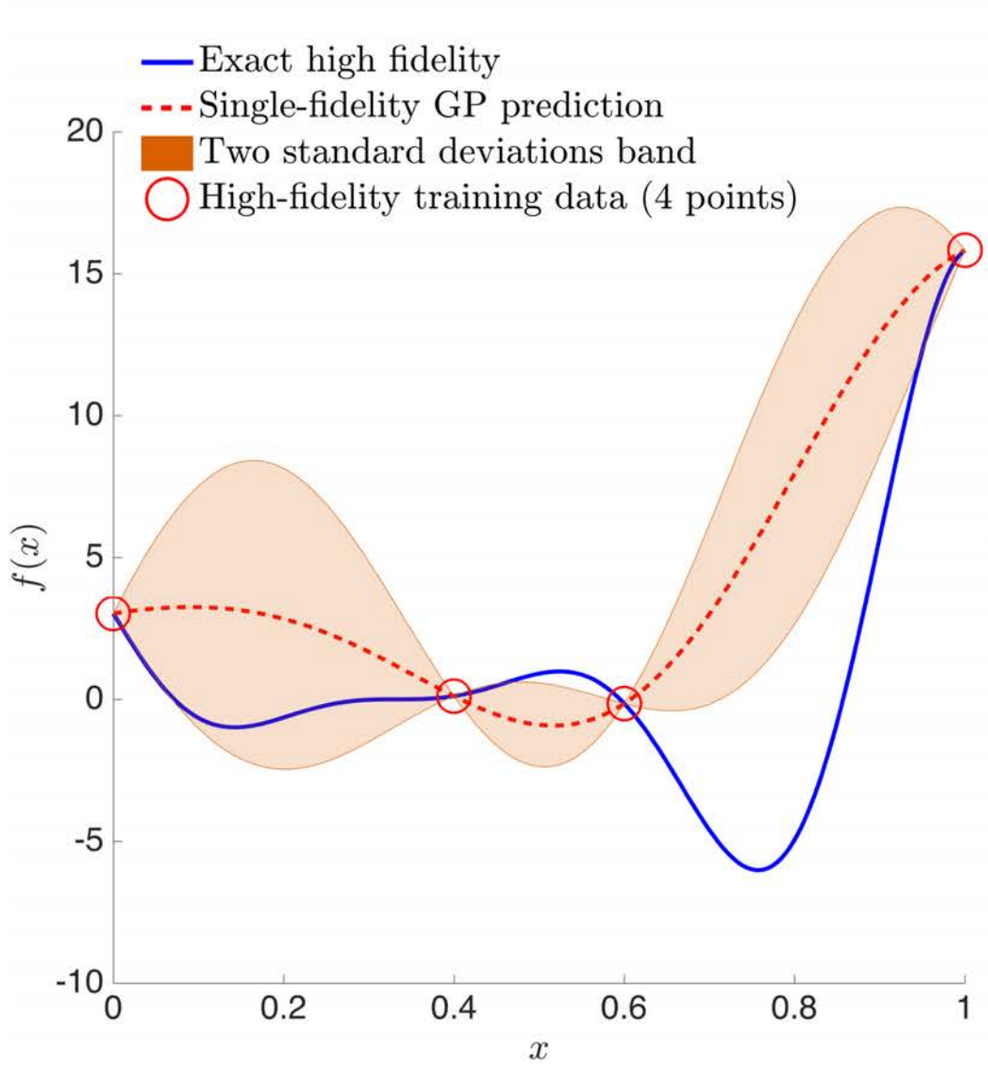
Number of runs is limited by time and computational resources

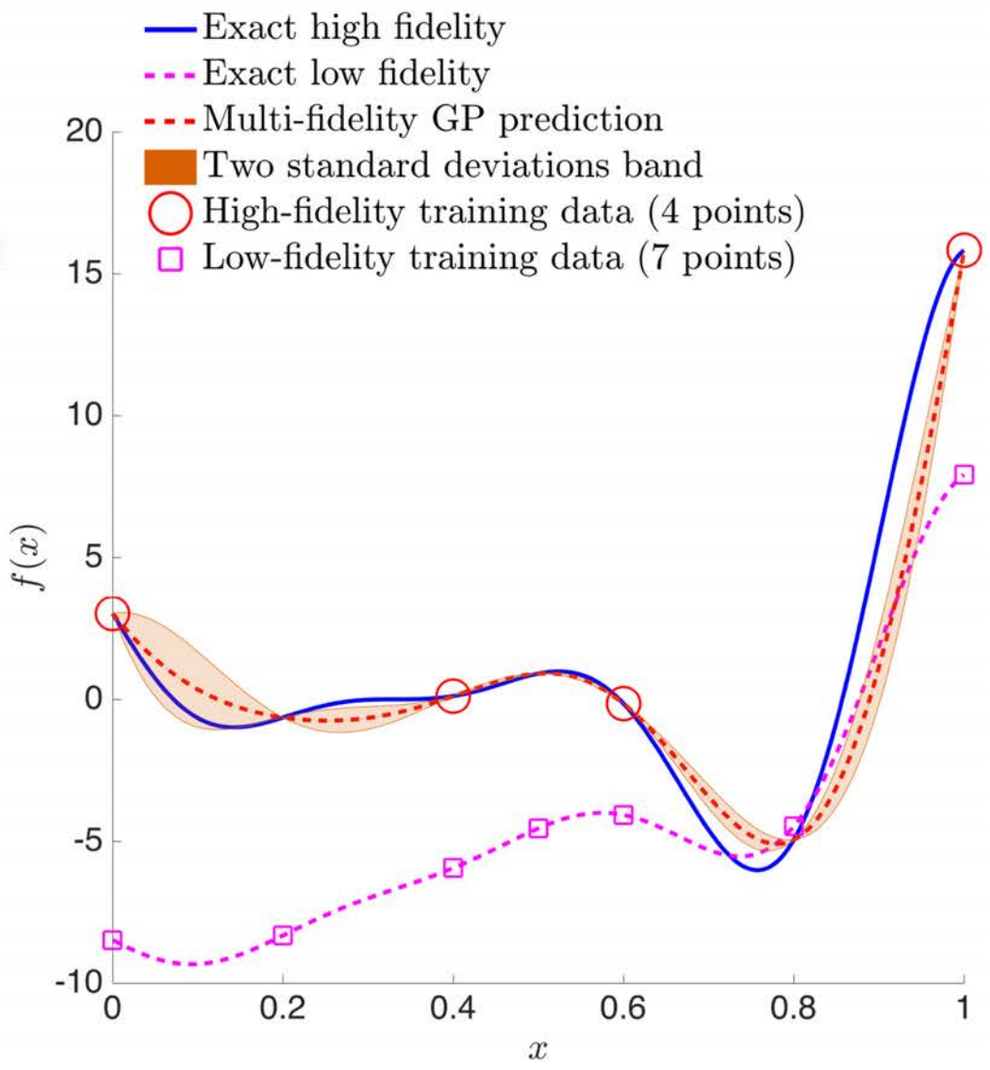
⇓

We cannot compute at all $(\boldsymbol{x};\boldsymbol{\xi})$

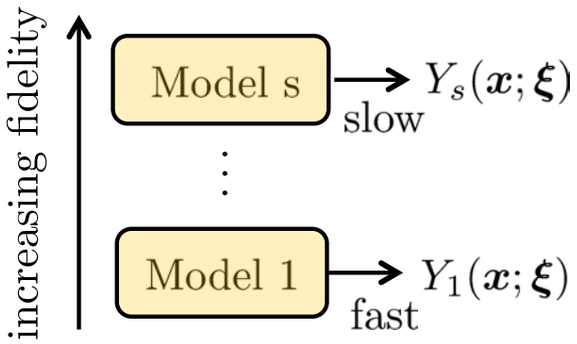
⇓

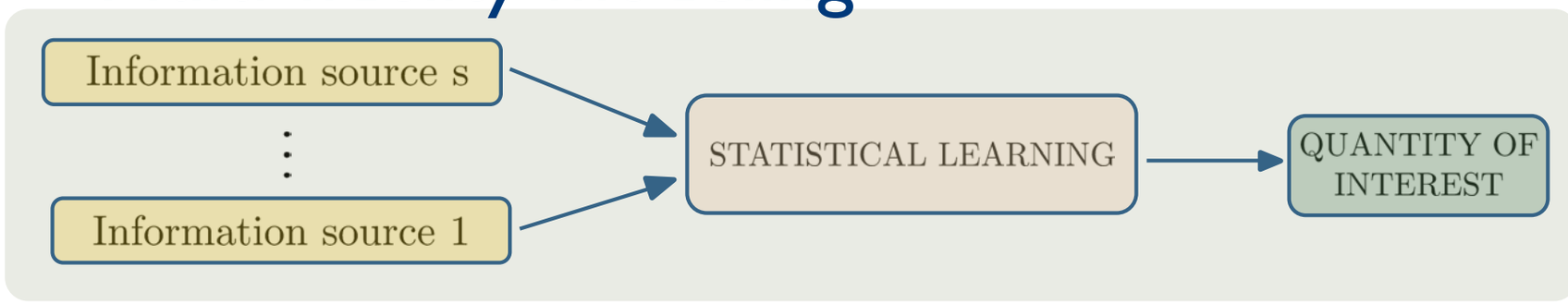
Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling





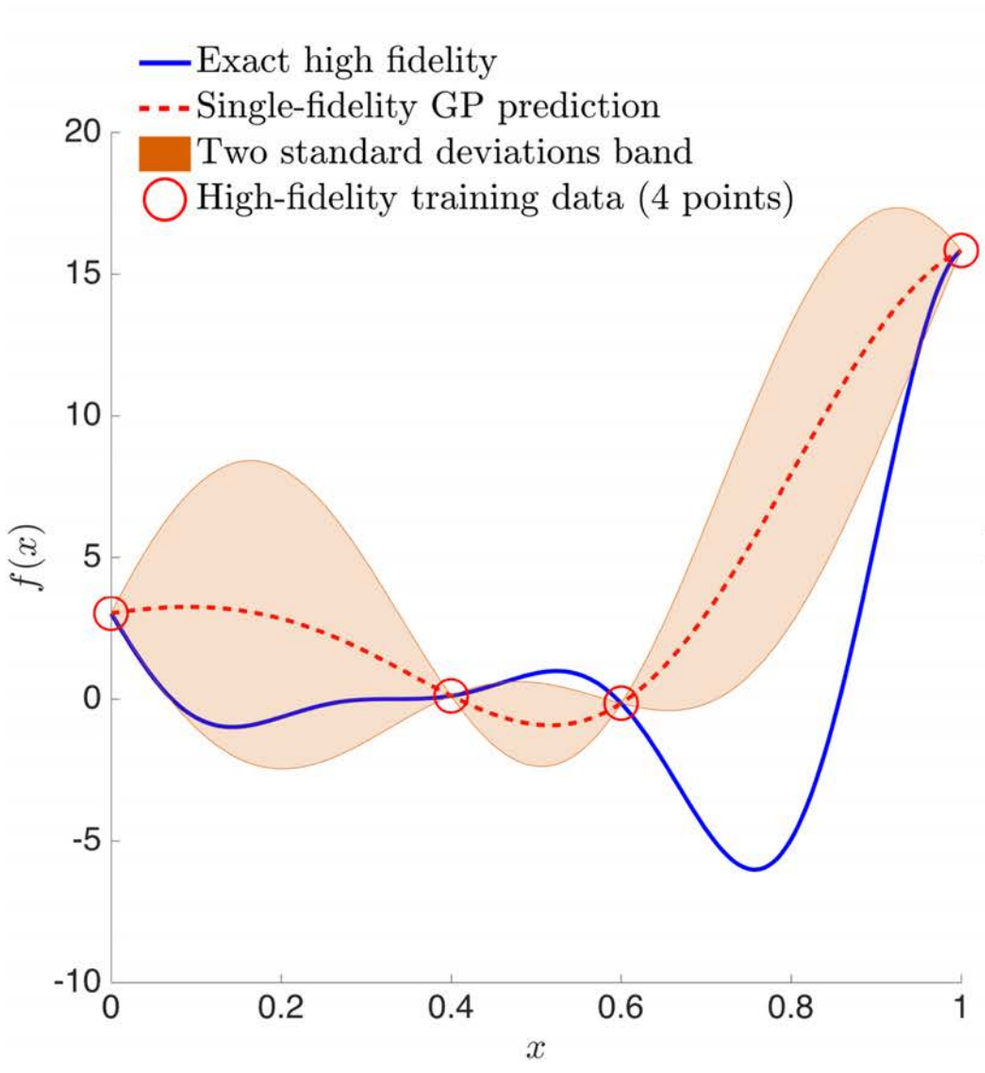
Number of runs is limited by time and computational resources

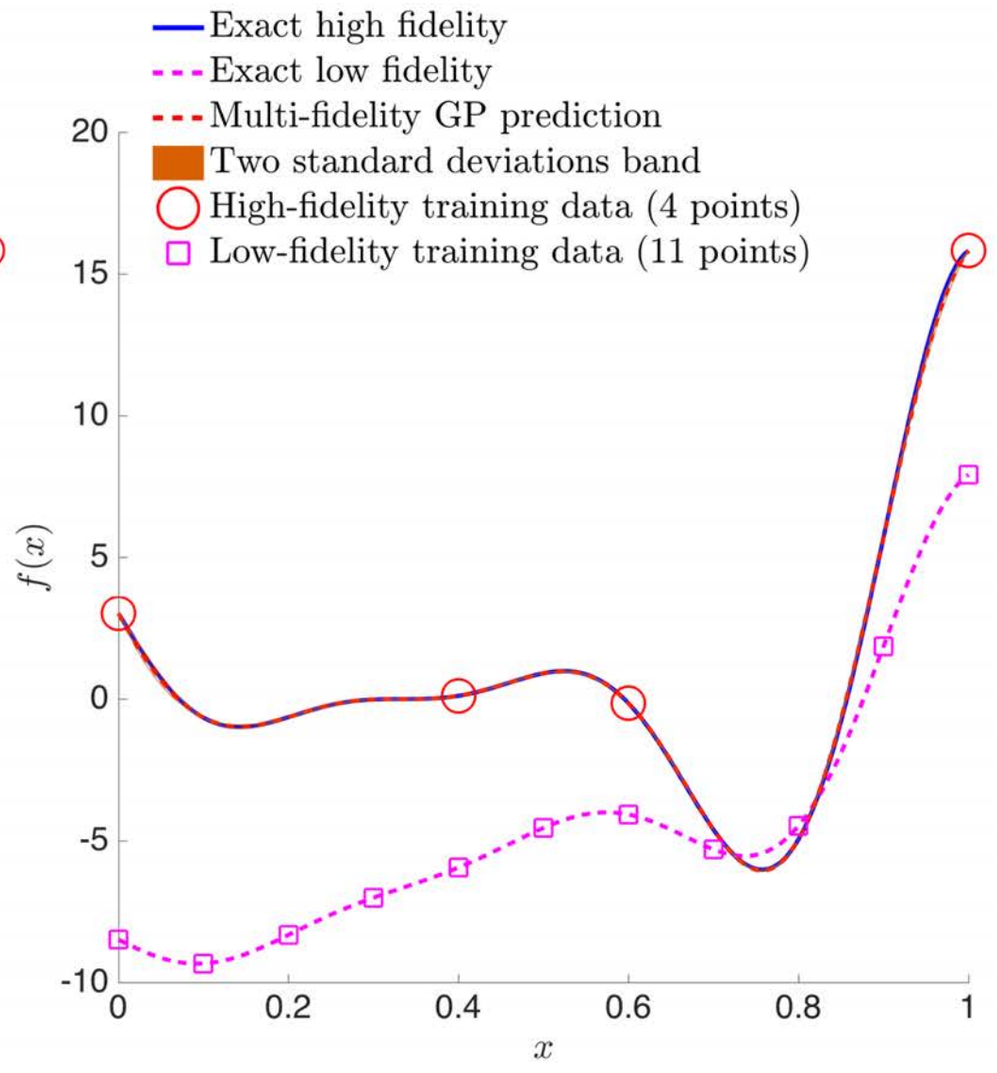
⇓

We cannot compute at all $(\boldsymbol{x};\boldsymbol{\xi})$

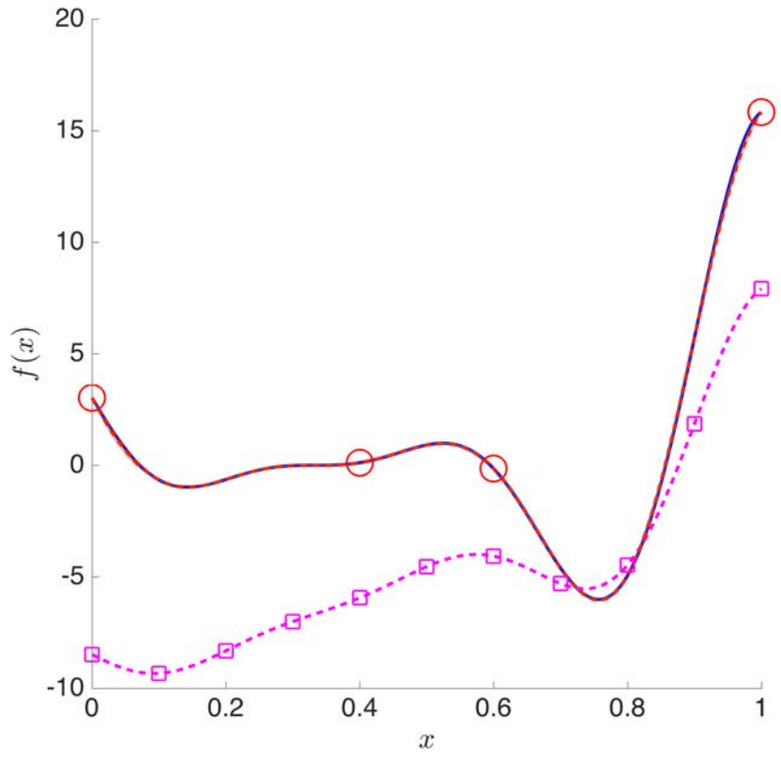
⇓

Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling



Multi-fidelity observations:

$$\boldsymbol{y}_L = f_L(\boldsymbol{x}_L) + \boldsymbol{\epsilon}_L$$
$$\boldsymbol{y}_H = f_H(\boldsymbol{x}_H) + \boldsymbol{\epsilon}_H$$

Probabilistic model:

$$f_H(\boldsymbol{x}) = \rho f_L(\boldsymbol{x}) + \delta(\boldsymbol{x})$$
$$f_L(\boldsymbol{x}) \sim \mathcal{GP}(0, k_L(\boldsymbol{x}, \boldsymbol{x}'; \theta_L))$$
$$\delta(\boldsymbol{x}) \sim \mathcal{GP}(0, k_H(\boldsymbol{x}, \boldsymbol{x}'; \theta_H))$$
$$\epsilon_L \sim \mathcal{N}(0, \sigma_{\epsilon_L}^2 \boldsymbol{I})$$
$$\epsilon_H \sim \mathcal{N}(0, \sigma_{\epsilon_H}^2 \boldsymbol{I})$$

Training:

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_L \\ \boldsymbol{y}_H \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} k_L(\boldsymbol{x}_L, \boldsymbol{x}'_L; \theta_L) + \sigma_{\epsilon_L}^2 \boldsymbol{I} & \rho k_L(\boldsymbol{x}_L, \boldsymbol{x}'_H; \theta_L) \\ \rho k_L(\boldsymbol{x}_H, \boldsymbol{x}'_L; \theta_L) & \rho^2 k_L(\boldsymbol{x}_H, \boldsymbol{x}'_H; \theta_L) + k_H(\boldsymbol{x}_H, \boldsymbol{x}'_H; \theta_H) + \sigma_{\epsilon_H}^2 \boldsymbol{I} \end{bmatrix} \right)$$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_L \\ \boldsymbol{x}_H \end{bmatrix} \qquad -\log p(\boldsymbol{y}|\boldsymbol{X}, \theta_L, \theta_H, \rho, \sigma_{\epsilon_L}^2, \sigma_{\epsilon_H}^2) = \frac{1}{2}\log|\boldsymbol{K}| + \frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}^{-1}\boldsymbol{y} - \frac{N_L + N_H}{2}\log 2\pi$$

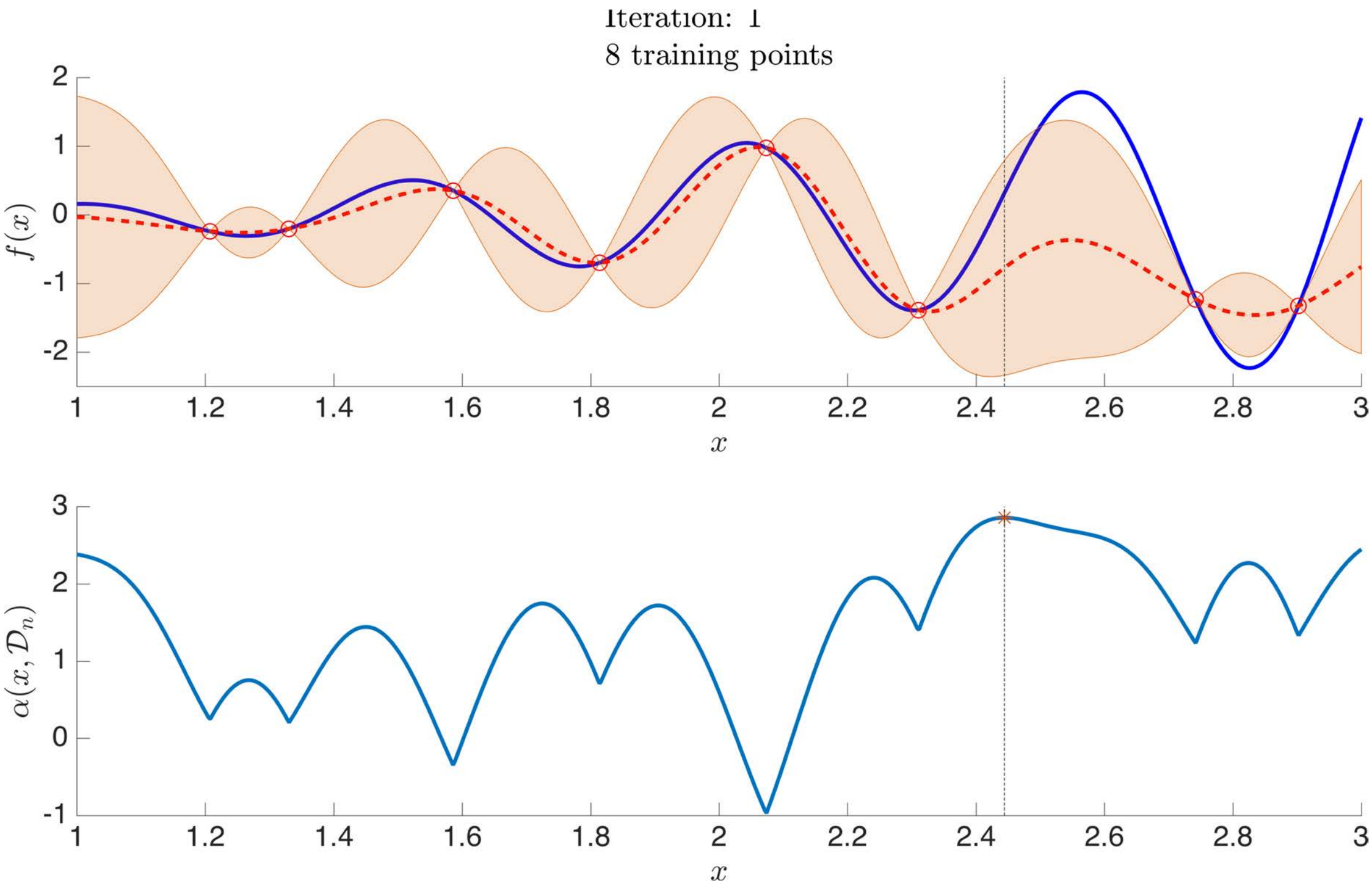
Prediction:

$$p(f(\boldsymbol{x}^*)|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}^*) \sim \mathcal{N}(f(\boldsymbol{x}^*)|\mu(\boldsymbol{x}^*), \sigma^2(\boldsymbol{x}^*))$$
$$\mu(\boldsymbol{x}^*) = \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X})\boldsymbol{K}^{-1}\boldsymbol{y}$$
$$\sigma(\boldsymbol{x}^*) = \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X})\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}^*)$$

M.C Kennedy, and A. O'Hagan. *Predicting the output from a complex computer code when fast approximations are available*, 2000.

Demo code: https://github.com/PredictiveIntelligenceLab/GPTutorial

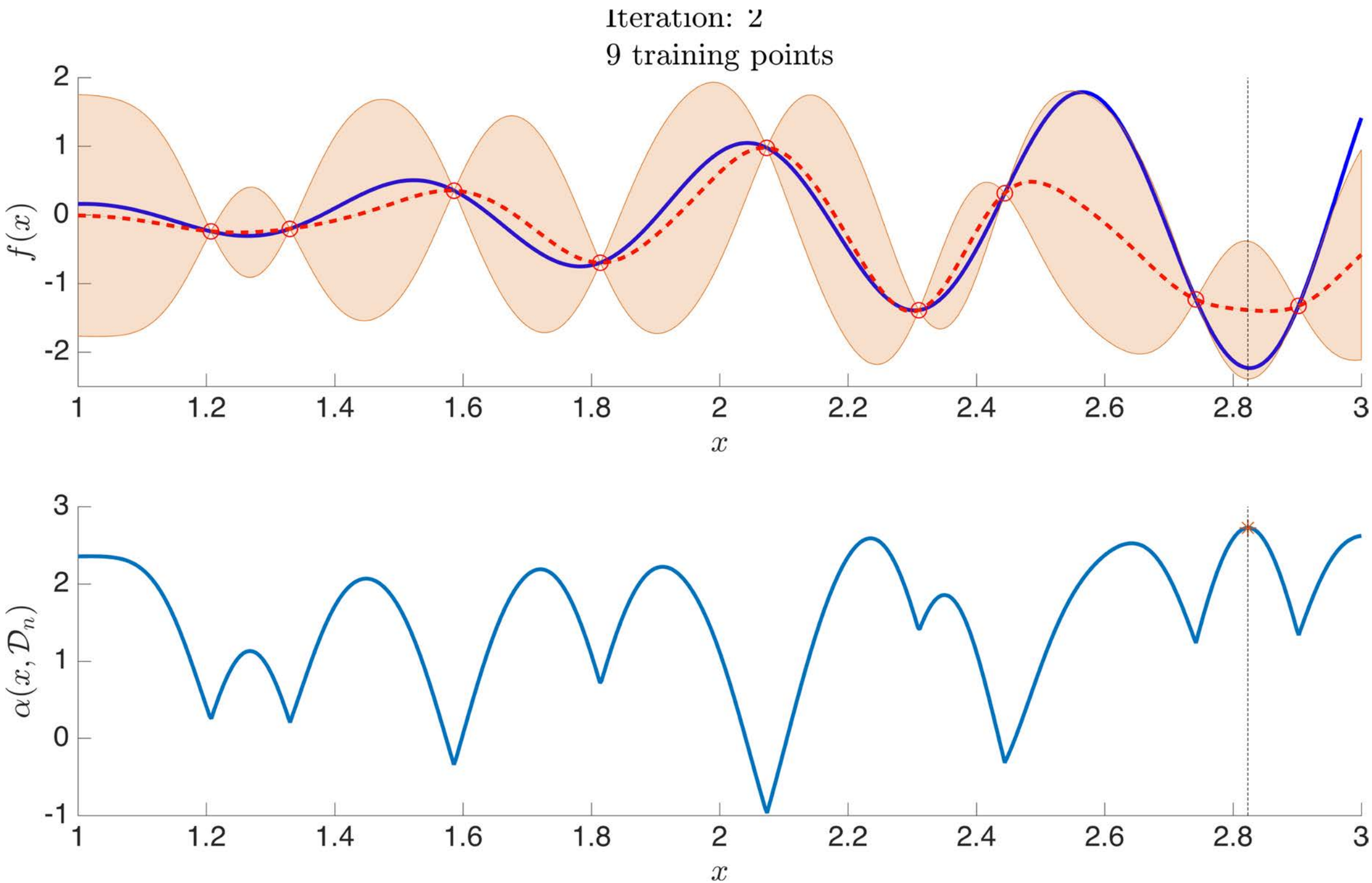
Multi-fidelity Bayesian optimization



Sample at the locations that minimize the lower super-quintile risk confidence bound of the posterior:

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \mu(\boldsymbol{x}) - \frac{\phi(\Phi^{-1}(\alpha))}{1-\alpha}\sigma(\boldsymbol{x})$$

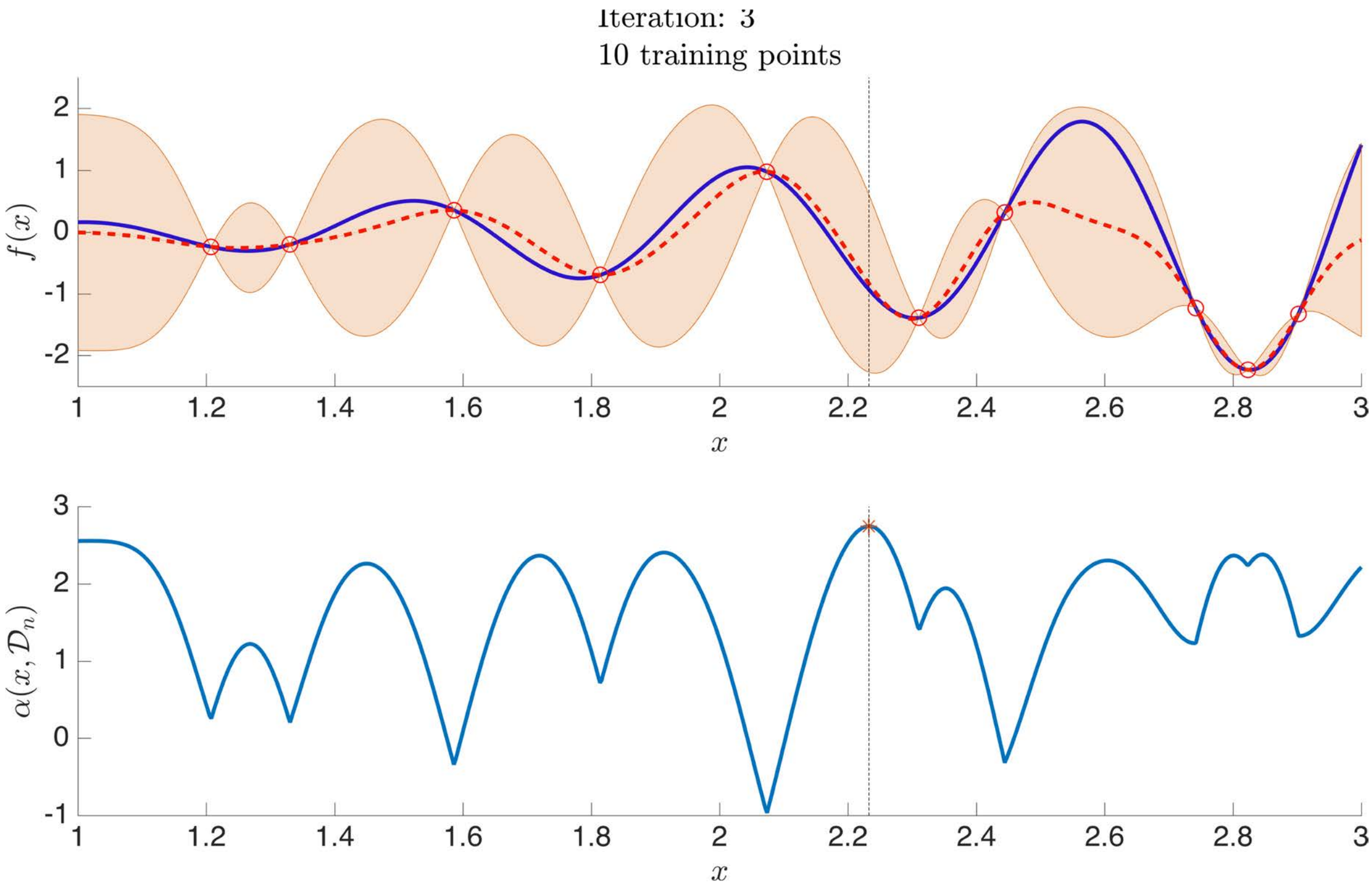
Multi-fidelity Bayesian optimization



Sample at the locations that minimize the lower super-quintile risk confidence bound of the posterior:

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \mu(\boldsymbol{x}) - \frac{\phi(\Phi^{-1}(\alpha))}{1-\alpha}\sigma(\boldsymbol{x})$$

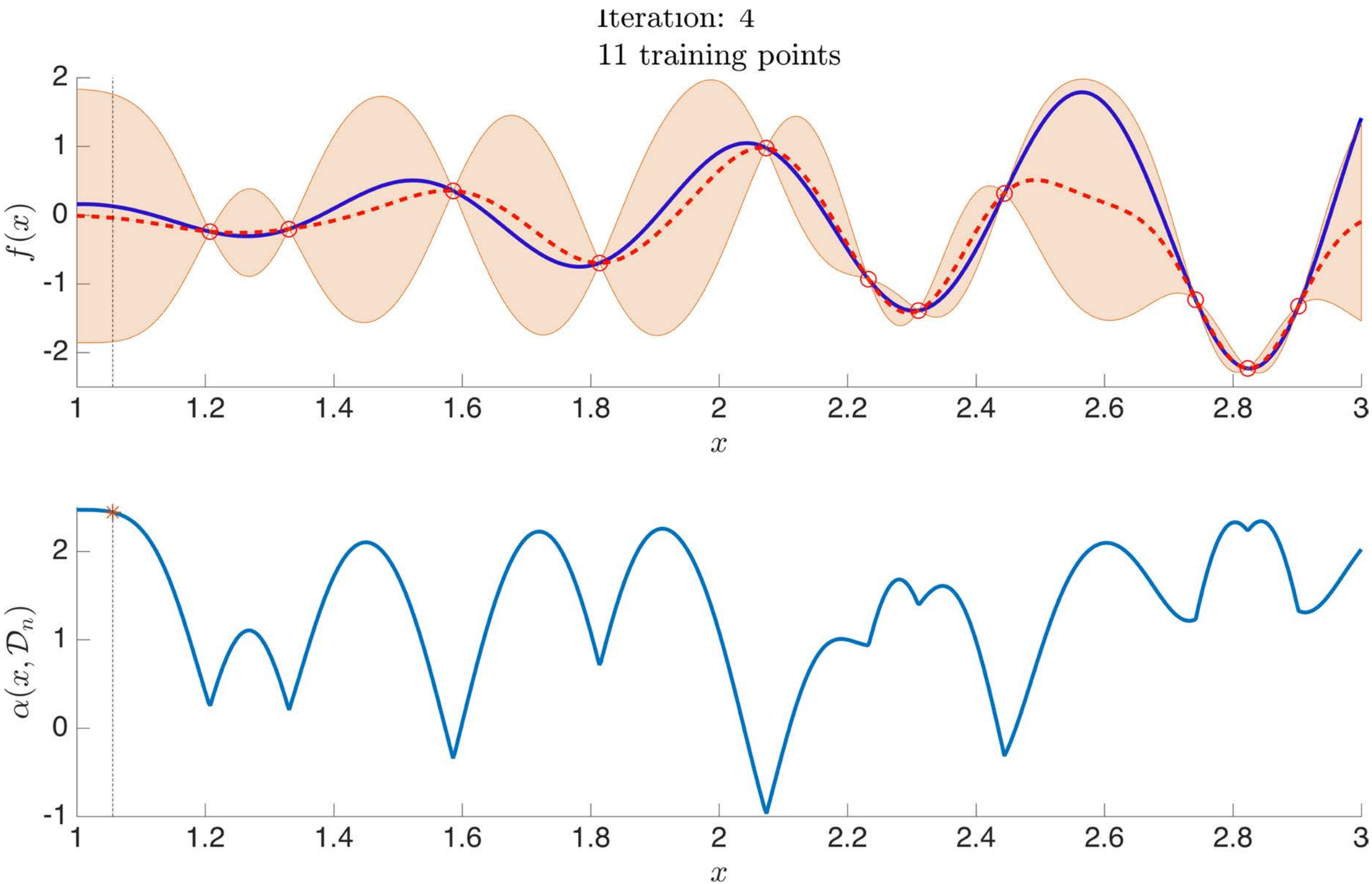
Multi-fidelity Bayesian optimization



Sample at the locations that minimize the lower super-quintile risk confidence bound of the posterior:

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \mu(\boldsymbol{x}) - \frac{\phi(\Phi^{-1}(\alpha))}{1-\alpha}\sigma(\boldsymbol{x})$$

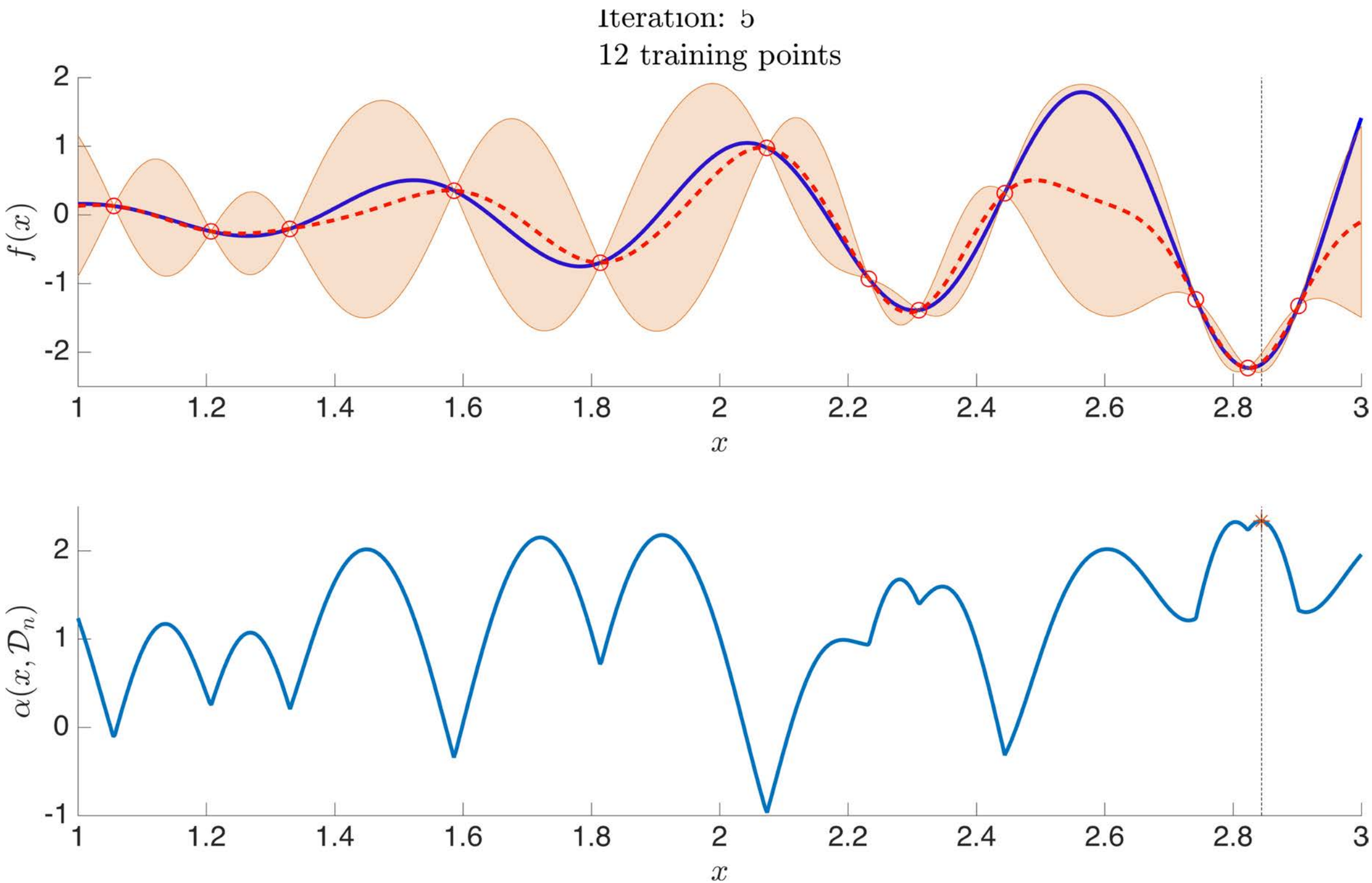
Multi-fidelity Bayesian optimization



Sample at the locations that minimize the lower super-quintile risk confidence bound of the posterior:

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \mu(\boldsymbol{x}) - \frac{\phi(\Phi^{-1}(\alpha))}{1-\alpha}\sigma(\boldsymbol{x})$$

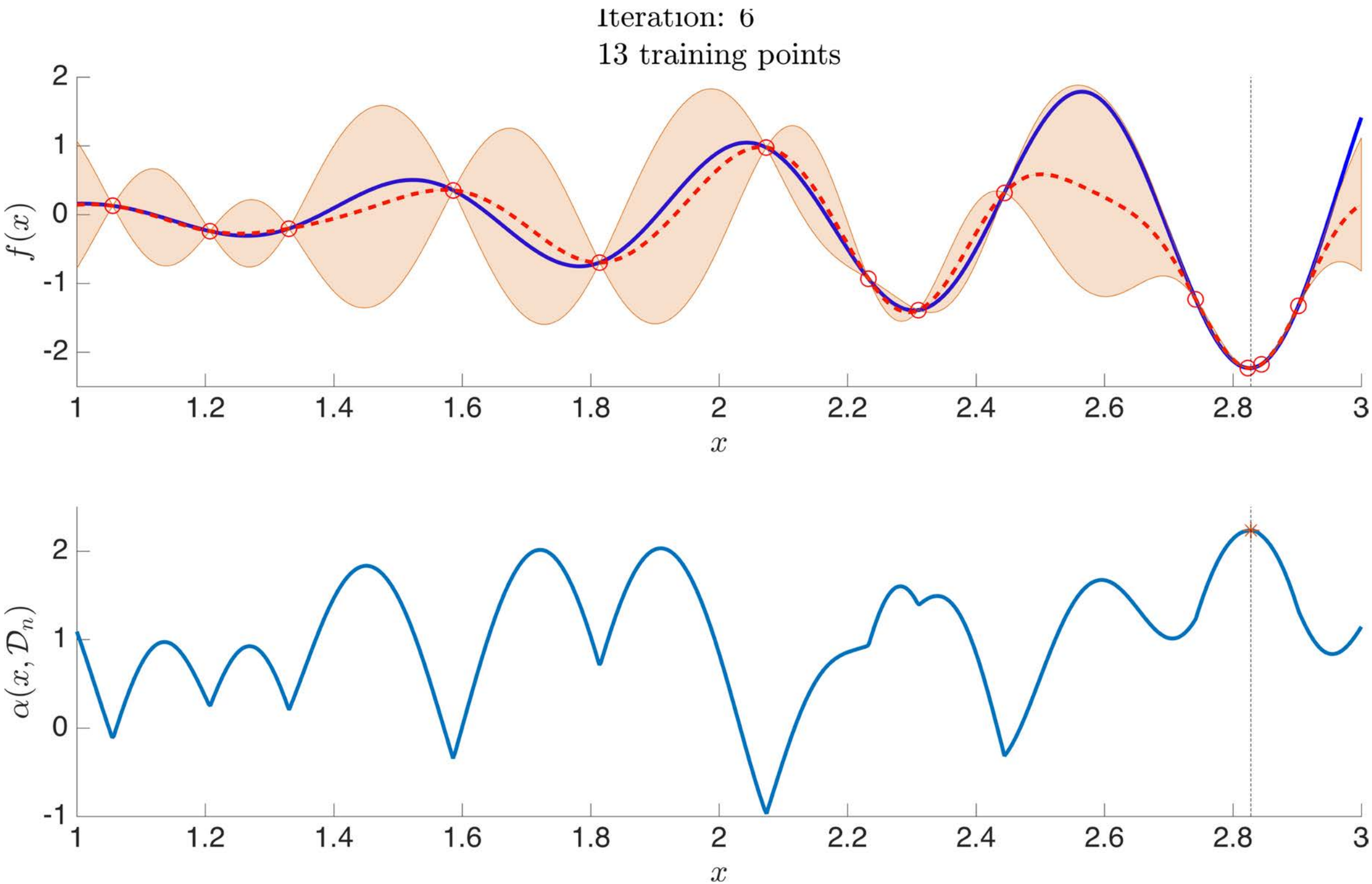
Multi-fidelity Bayesian optimization



Sample at the locations that minimize the lower super-quintile risk confidence bound of the posterior:

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \mu(\boldsymbol{x}) - \frac{\phi(\Phi^{-1}(\alpha))}{1-\alpha}\sigma(\boldsymbol{x})$$

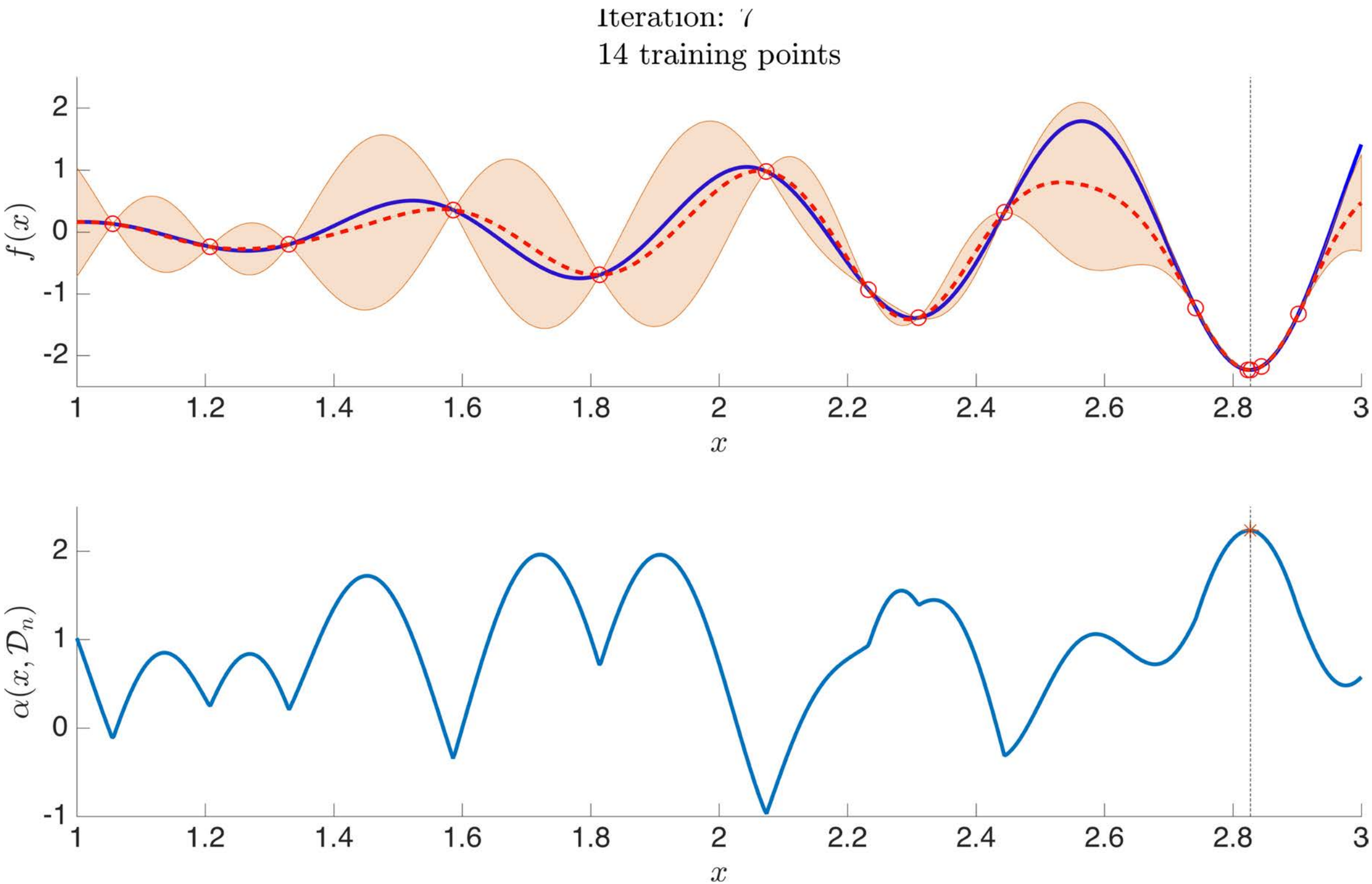
Multi-fidelity Bayesian optimization



Sample at the locations that minimize the lower super-quintile risk confidence bound of the posterior:

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \mu(\boldsymbol{x}) - \frac{\phi(\Phi^{-1}(\alpha))}{1-\alpha}\sigma(\boldsymbol{x})$$

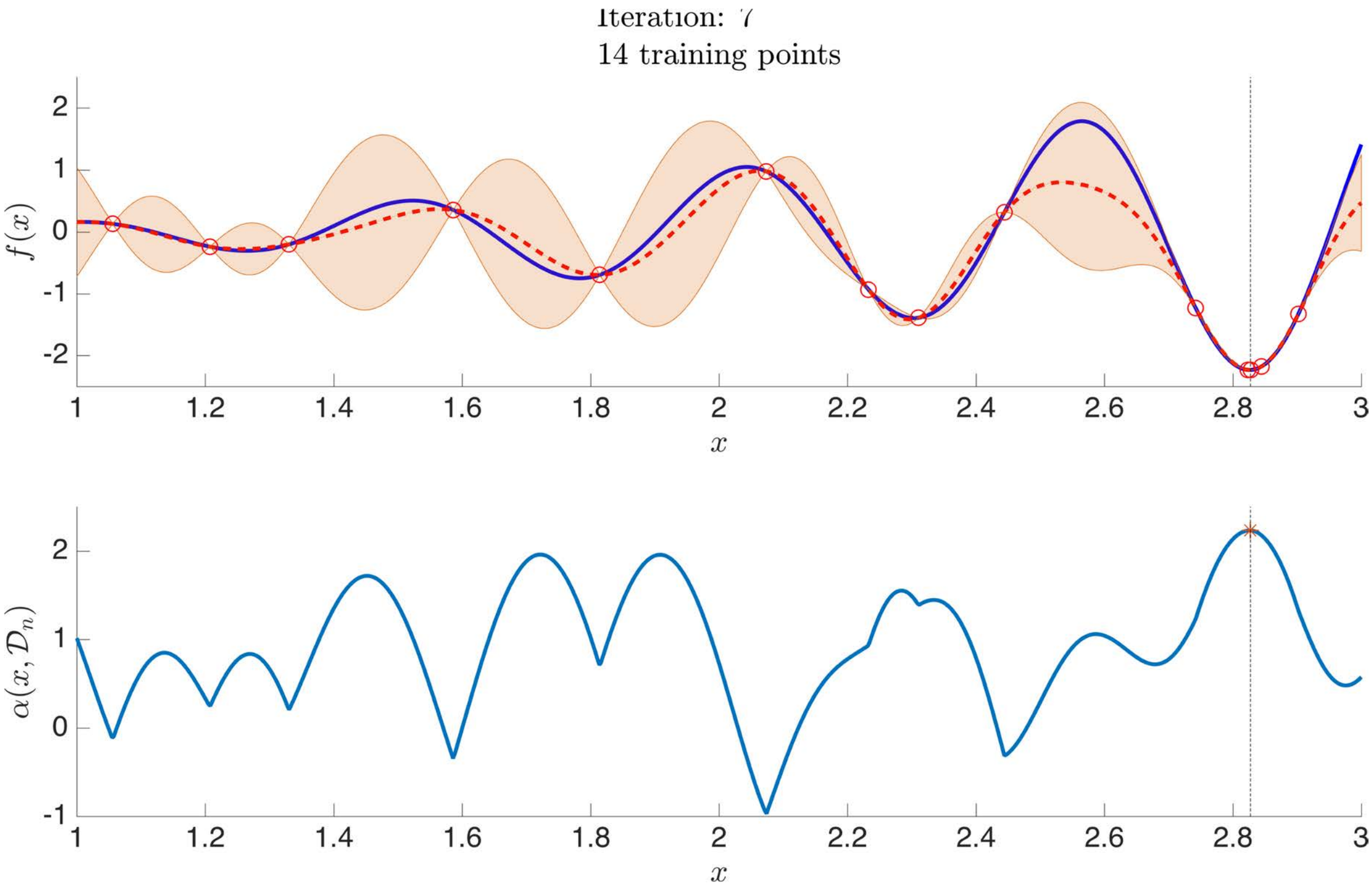
Multi-fidelity Bayesian optimization



Sample at the locations that minimize the lower super-quintile risk confidence bound of the posterior:

$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \mu(\boldsymbol{x}) - \frac{\phi(\Phi^{-1}(\alpha))}{1-\alpha}\sigma(\boldsymbol{x})$$

Multi-fidelity Bayesian optimization

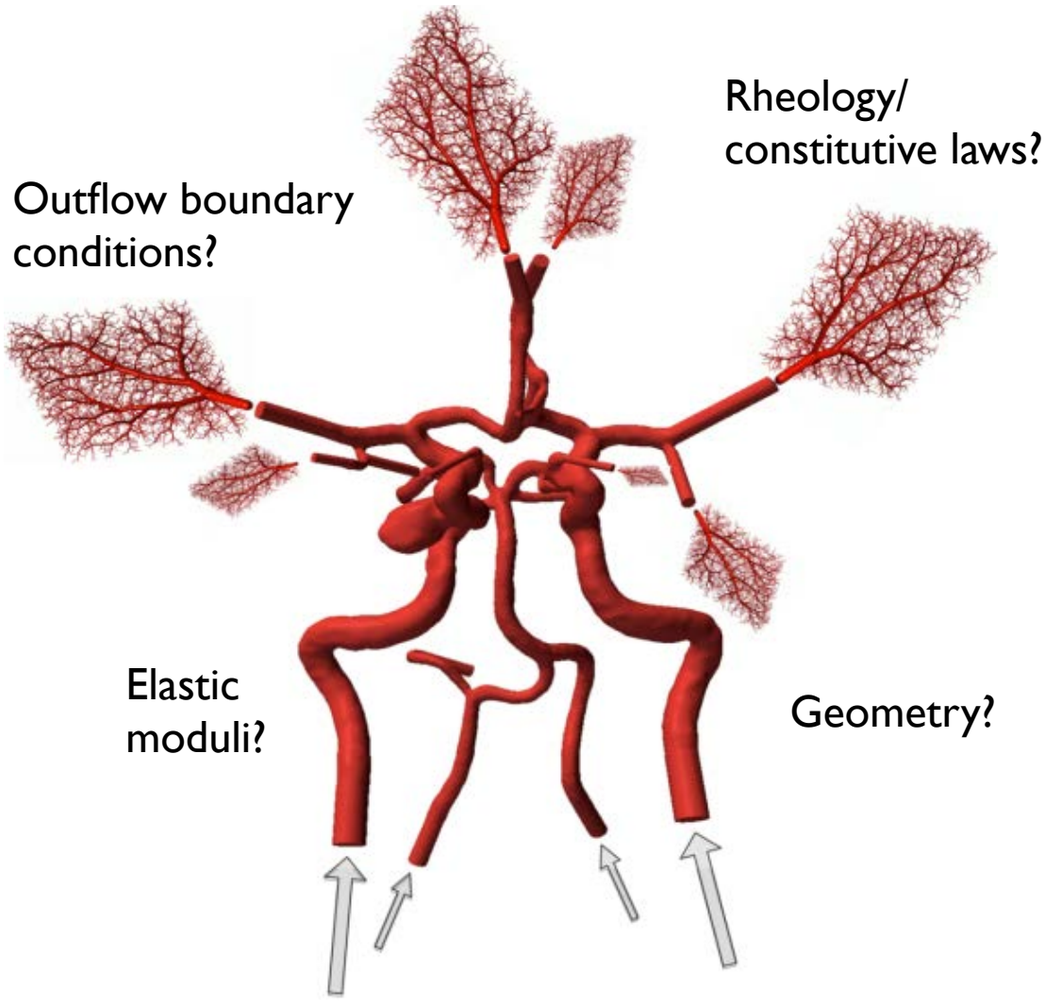


Sample at the locations that minimize the lower super-quintile risk confidence bound of the posterior:

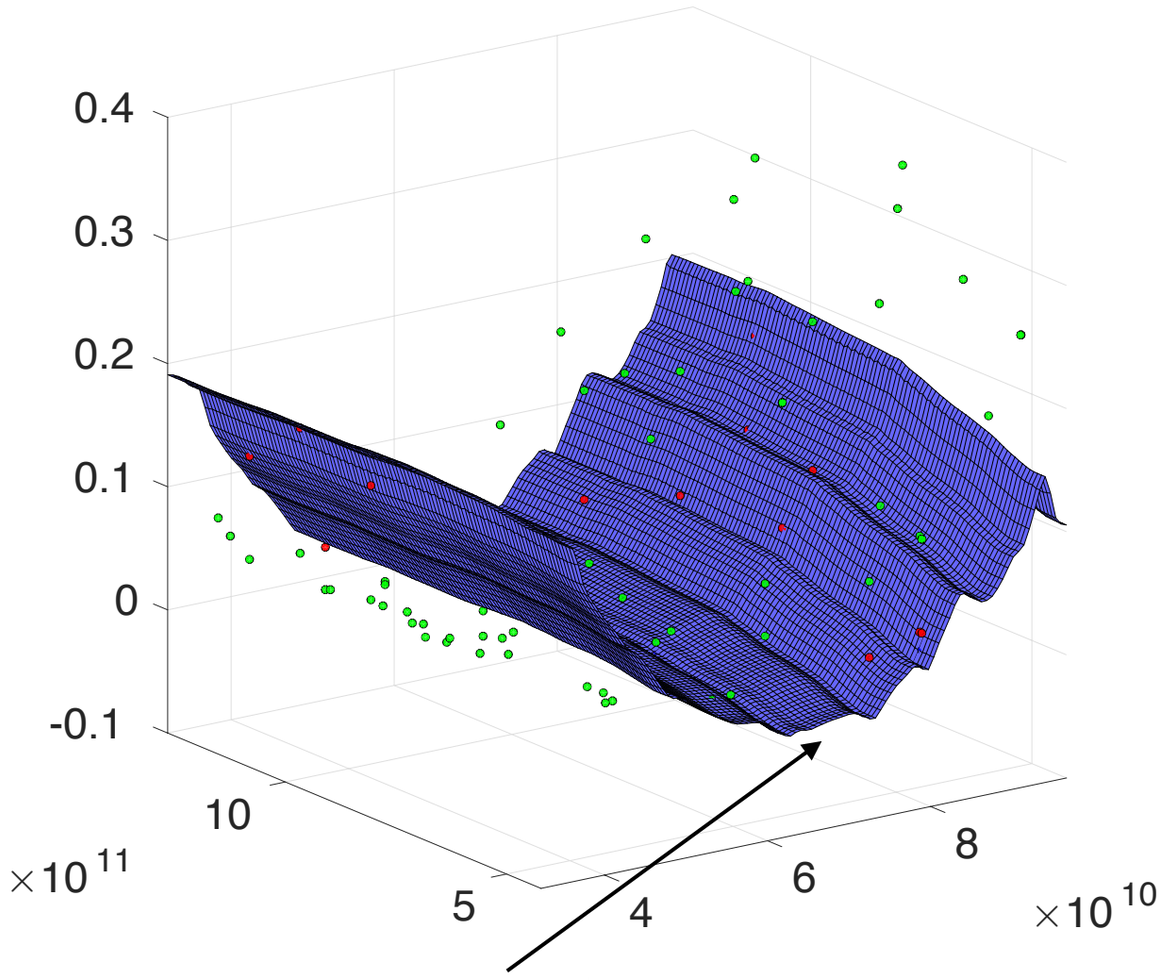
$$\boldsymbol{x}_{n+1} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \mu(\boldsymbol{x}) - \frac{\phi(\Phi^{-1}(\alpha))}{1-\alpha}\sigma(\boldsymbol{x})$$

Example application: *Calibration of blood flow simulations*

Goal: Calibrate the outflow boundary condition parameters to match a target inlet systolic pressure.



Models (e.g. 3D vs 1D,
continuum vs atomistic, etc.)



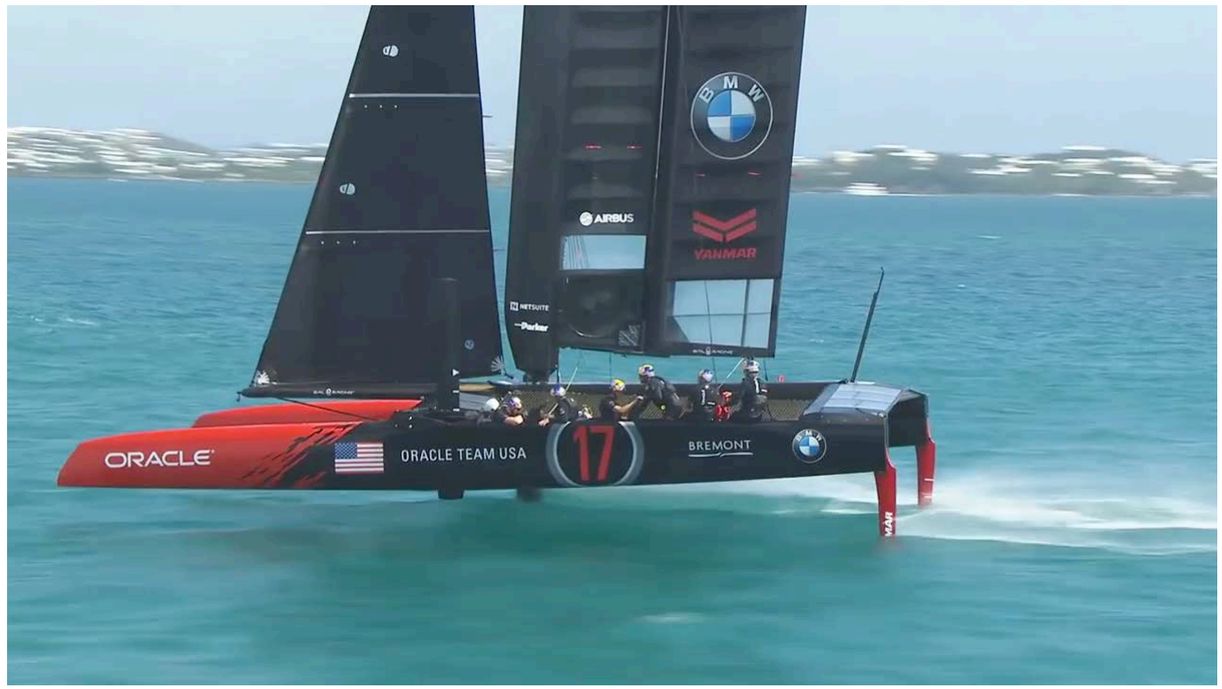
Decreased the relative error to 1e-3 after 3 iterations of BO, mainly sampling the lowest fidelity (cheapest) solver.

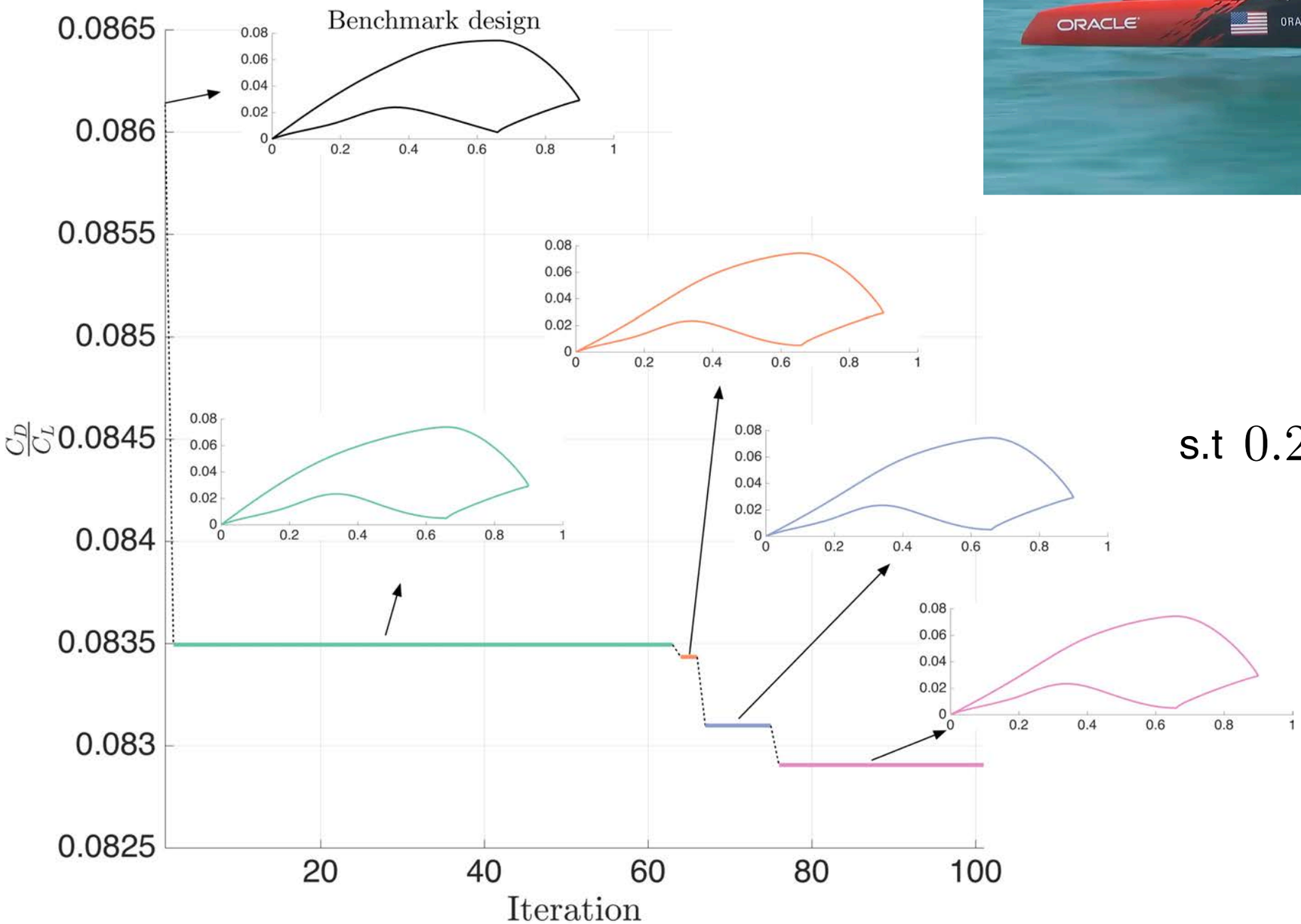
Multi-fidelity approach:

1.) 3D Navier-Stokes (spectral/hp elements, rigid artery) - high fidelity O(hrs)
2.) Non-linear 1D-FSI (DG, compliant artery) - intermediate fidelity O(mins)
3.) Linearized 1D-FSI solver around an inaccurate reference state - low fidelity O(s)

P. Perdikaris, and G.E Karniadakis. "Model inversion via multi-fidelity Bayesian optimization." J. R. Soc. Interface (2016)

Taking the human out of the loop: Multi-fidelity Bayesian optimization of super-cavitating hydrofoils





$$\min_{\mathbf{x}\in\mathcal{X}} \mathcal{R}_{0.9}\left(\frac{C_D}{C_L}(\boldsymbol{x};\boldsymbol{\xi})\right)$$

$$\text{s.t } 0.23 \leq \mathcal{R}_{0.5}(C_L(\boldsymbol{x};\boldsymbol{\xi})) \leq 0.26$$

$$\mathcal{R}_{0.5}(I_{\mathrm{x}}(\boldsymbol{x};\boldsymbol{\xi})) \geq 8.1 \cdot 10^{-6}$$

$$\mathcal{R}_{0.5}(T_{\mathrm{p}}(\boldsymbol{x};\boldsymbol{\xi})) \geq 0.00132$$

35 design variables
B-MF: 735 LF + 120 HF runs
GA-LF: ~50,000 LF runs

Bonfiglio, L., Perdikaris, P., Brizzolara, S., & Karniadakis, G. E. (2018). Multi-fidelity optimization of super-cavitating hydrofoils. Computer Methods in Applied Mechanics and Engineering.

Physics-informed priors

$$\mathcal{L}_x u(x) = f(x)$$

$$u(x) \sim \mathcal{GP}(0, g(x, x'; \theta)) \quad \longrightarrow \quad f(x) \sim \mathcal{GP}(0, k(x, x'; \theta))$$

$$k(x, x'; \theta) = \mathcal{L}_x \mathcal{L}_{x'} g(x, x'; \theta)$$

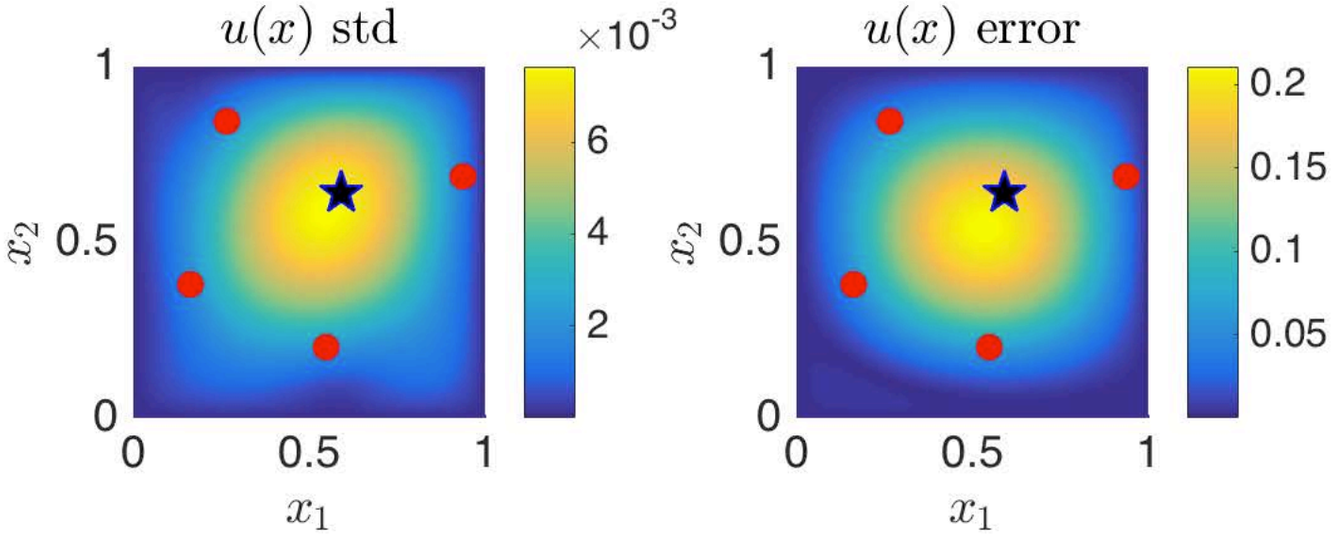
$$-\log p(\boldsymbol{y}|\phi, \theta, \sigma_{n_u}^2, \sigma_{n_f}^2) = \frac{1}{2}\log|\boldsymbol{K}| + \frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}^{-1}\boldsymbol{y} + \frac{N}{2}\log 2\pi\,,$$

where $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_u \\ \boldsymbol{y}_f \end{bmatrix}$, $p(\boldsymbol{y}|\phi, \theta, \sigma_{n_u}^2, \sigma_{n_f}^2) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$, and $\boldsymbol{K}$ is given by

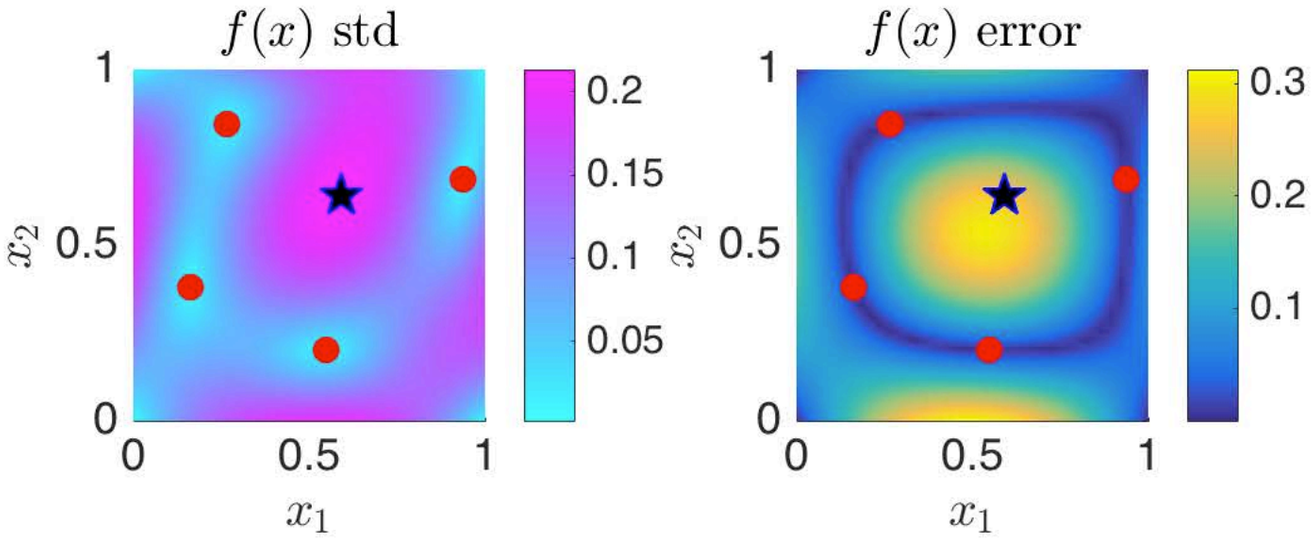
$$\boldsymbol{K} = \begin{bmatrix} k_{uu}(\boldsymbol{X}_u, \boldsymbol{X}_u; \theta) + \sigma_{n_u}^2 \boldsymbol{I}_{n_u} & k_{uf}(\boldsymbol{X}_u, \boldsymbol{X}_f; \theta, \phi) \\ k_{fu}(\boldsymbol{X}_f, \boldsymbol{X}_u; \theta, \phi) & k_{ff}(\boldsymbol{X}_f, \boldsymbol{X}_f; \theta, \phi) + \sigma_{n_f}^2 \boldsymbol{I}_{n_f} \end{bmatrix}.$$

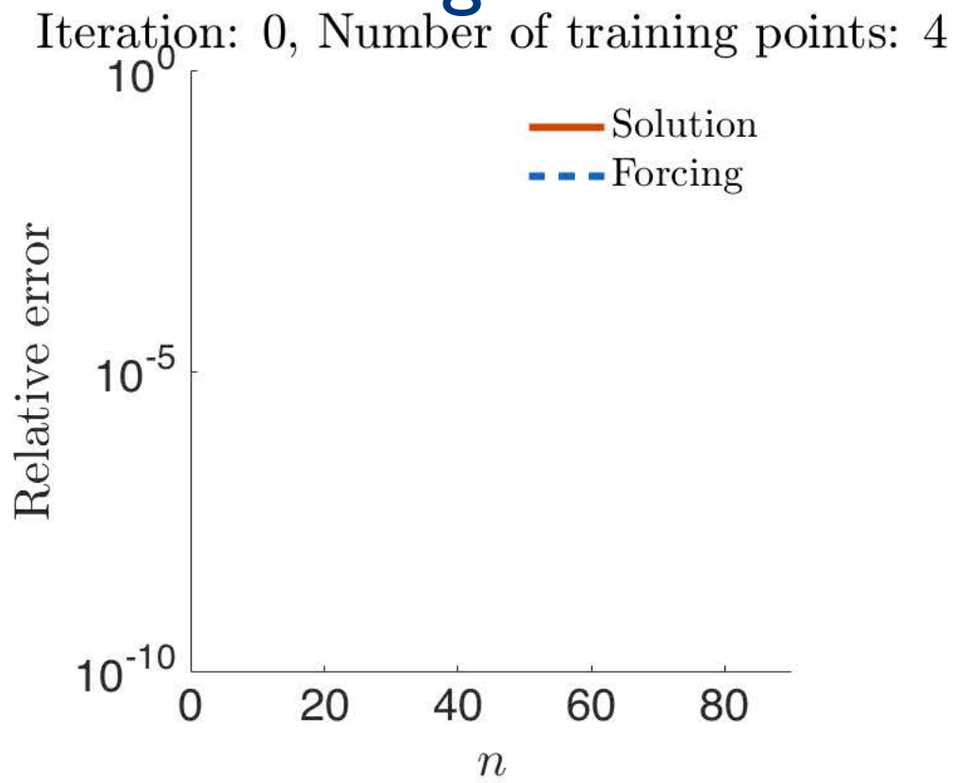
Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Inferring solutions of differential equations using noisy multi-fidelity data. Journal of Computational Physics, 335, 736-746.

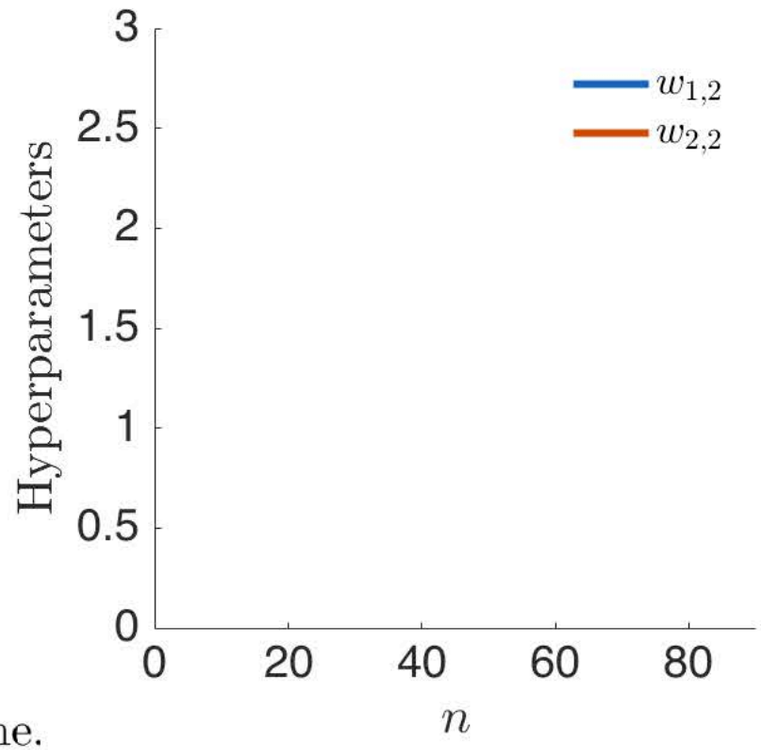
Adaptive refinement via active learning



$$\frac{\partial^2 u}{\partial {x_1}^2} + \frac{\partial^2 u}{\partial {x_2}^2} = f(x_1, x_2)$$







- denotes the training data actively collected by the scheme.
- ★ denotes the next sampling point suggested by the active learning scheme.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Inferring solutions of differential equations using noisy multi-fidelity data. Journal of Computational Physics, 335, 736-746.

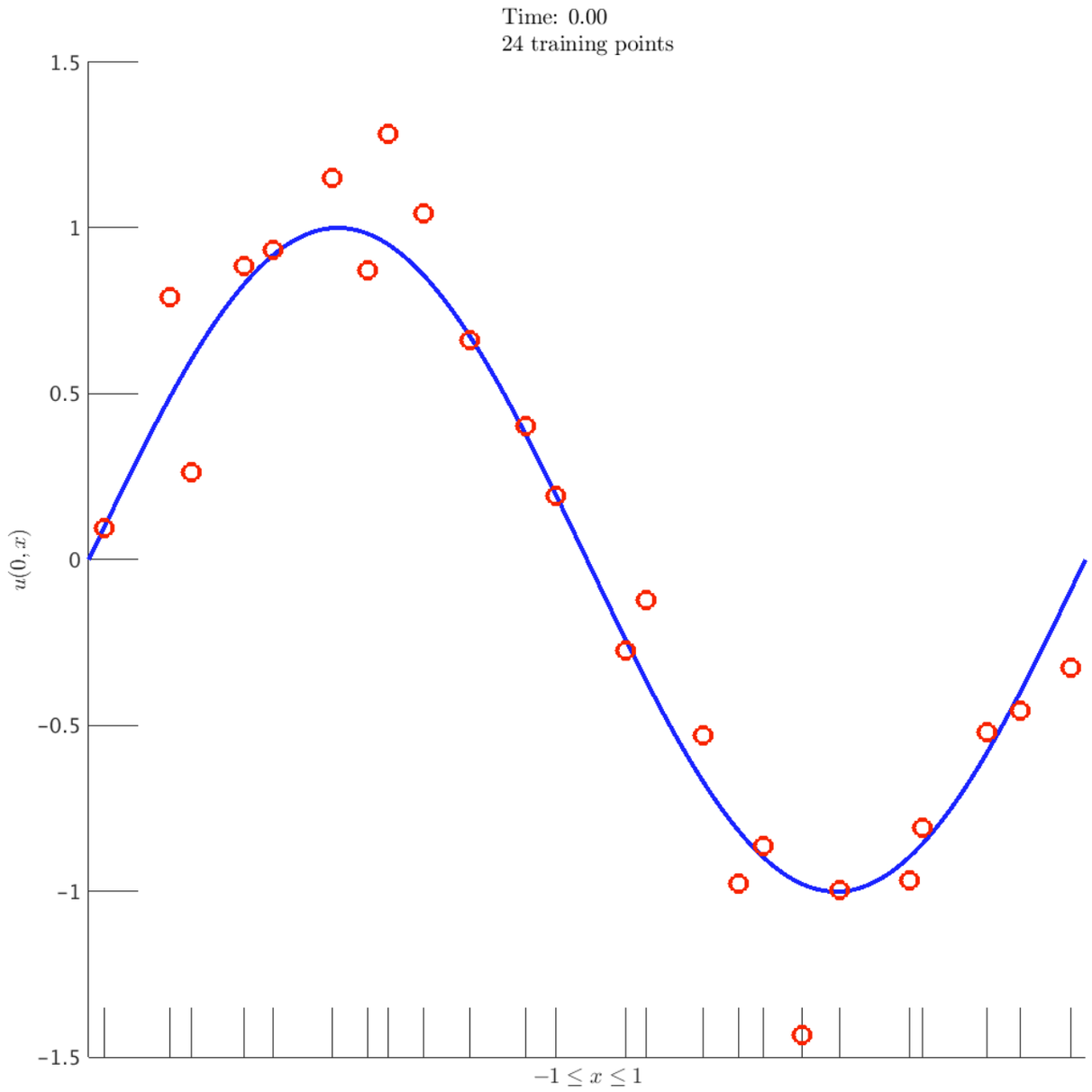
Extension to non-linear equations

Example: 1D viscous Burgers —> *The equation, along with the choice of a time-stepping scheme define a GP prior!*

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = u^0(x)$$
$$u(t, 0) = u(t, 1) = 0. \quad \nu = \pi/100$$

$$+ \quad \mathcal{L}_x u^{n+1}(x) := u^{n+1}(x) + \Delta t \; u^n(x) \; \frac{du^{n+1}(x)}{dx} - \Delta t \; \nu \frac{d^2 u^{n+1}(x)}{dx^2} = u^n(x)$$

e.g., Backward Euler time-stepping



Remarks:

- Despite starting from a stationary prior:

$$u^{n+1,n+1}(x) \sim \mathcal{GP}(0, k^{n+1,n+1}(x, x'; \theta))$$

the structure of the of the kernel:

$$k^{n,n}(x, x'; \theta) = \mathcal{L}_x \mathcal{L}_{x'} k^{n+1,n+1}(x, x'; \theta)$$

can generate discontinuous solutions!

- This is a general approach applicable to any nonlinear equation and any time-stepping scheme.
- We only need to derive the kernel $k^{n,n}(x, x'; \theta)$
- Under this setup, fully implicit time-stepping schemes have the same complexity as their explicit counterparts. Hence, one can obtain highly accurate and stable schemes at no extra cost.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2018). Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations. SIAM Journal on Scientific Computing, 40(1), A172-A198.

Discovery of "hidden physics"

$$h_t + \mathcal{N}_x^\lambda h = 0,\ x \in \Omega,\ t \in [0, T]$$

Temporal discretization: $h^n + \Delta t \mathcal{N}_x^\lambda h^n = h^{n-1}\ ,\quad h^n(x) \sim \mathcal{GP}(0, k(x, x', \theta))$

$$\downarrow$$

$$\begin{bmatrix} h^n \\ h^{n-1} \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} k^{n,n} & k^{n,n-1} \\ k^{n-1,n} & k^{n-1,n-1} \end{bmatrix}\right)$$

$k^{n,n}(x, x'; \theta)$, $k^{n,n-1}(x, x'; \theta, \lambda)$, $k^{n-1,n}(x, x'; \theta, \lambda)$, $k^{n-1,n-1}(x, x'; \theta, \lambda)$.

$\longrightarrow$

$k^{n,n} = k$, $k^{n,n-1} = \mathcal{L}_{x'}^\lambda k$, $k^{n-1,n} = \mathcal{L}_x^\lambda k$, $k^{n-1,n-1} = \mathcal{L}_x^\lambda \mathcal{L}_{x'}^\lambda k$

$$-\log p(\boldsymbol{h}|\theta, \lambda, \sigma^2) = \frac{1}{2}\boldsymbol{h}^T \boldsymbol{K}^{-1}\boldsymbol{h} + \frac{1}{2}\log|\boldsymbol{K}| + \frac{N}{2}\log(2\pi),$$

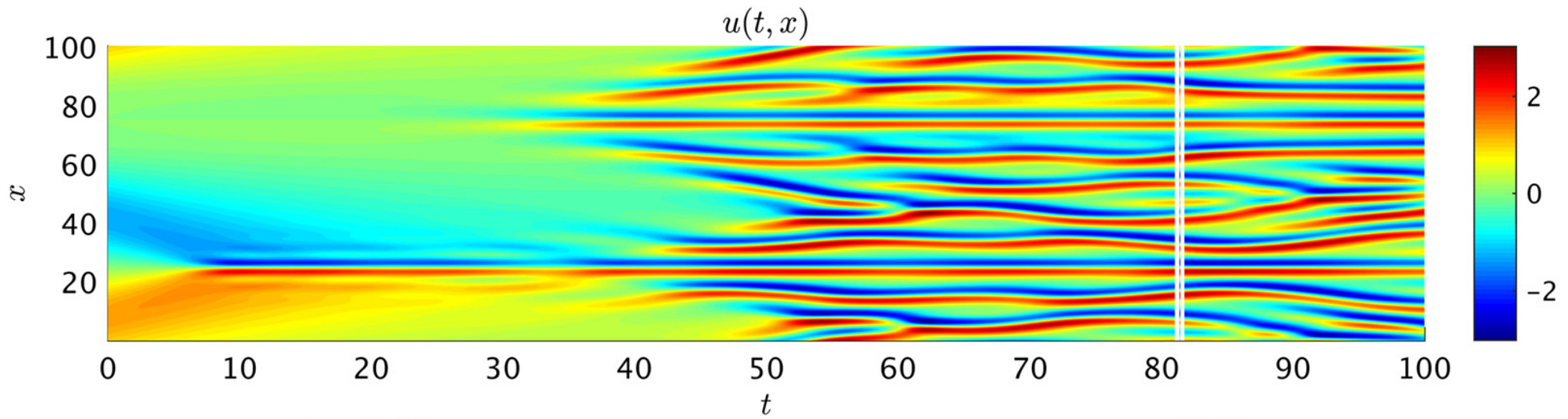
where $\boldsymbol{h} = \begin{bmatrix} \boldsymbol{h}^n \\ \boldsymbol{h}^{n-1} \end{bmatrix}$, $p(\boldsymbol{h}|\theta, \lambda, \sigma^2) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$, and $\boldsymbol{K}$ is given by

$$\boldsymbol{K} = \begin{bmatrix} k^{n,n}(\boldsymbol{x}^n, \boldsymbol{x}^n) & k^{n,n-1}(\boldsymbol{x}^n, \boldsymbol{x}^{n-1}) \\ k^{n-1,n}(\boldsymbol{x}^{n-1}, \boldsymbol{x}^n) & k^{n-1,n-1}(\boldsymbol{x}^{n-1}, \boldsymbol{x}^{n-1}) \end{bmatrix} + \sigma^2 \boldsymbol{I}.$$

Raissi, M., & Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. Journal of Computational Physics, 357, 125-141.

Discovery of "hidden physics"

Example: *Kuramoto-Sivashinsky equation:* $u_t + \lambda_1 u u_x + \lambda_2 u_{xx} + \lambda_3 u_{xxxx} = 0$



Correct PDE	$u_t + uu_x + u_{xx} + u_{xxxx} = 0$
Identified PDE (clean data)	$u_t + 0.952uu_x + 1.005u_{xx} + 0.980u_{xxxx} = 0$
Identified PDE (1% noise)	$u_t + 0.908uu_x + 0.951u_{xx} + 0.927u_{xxxx} = 0$

Kuramoto–Sivashinsky equation: Resulting statistics for the learned parameter values.

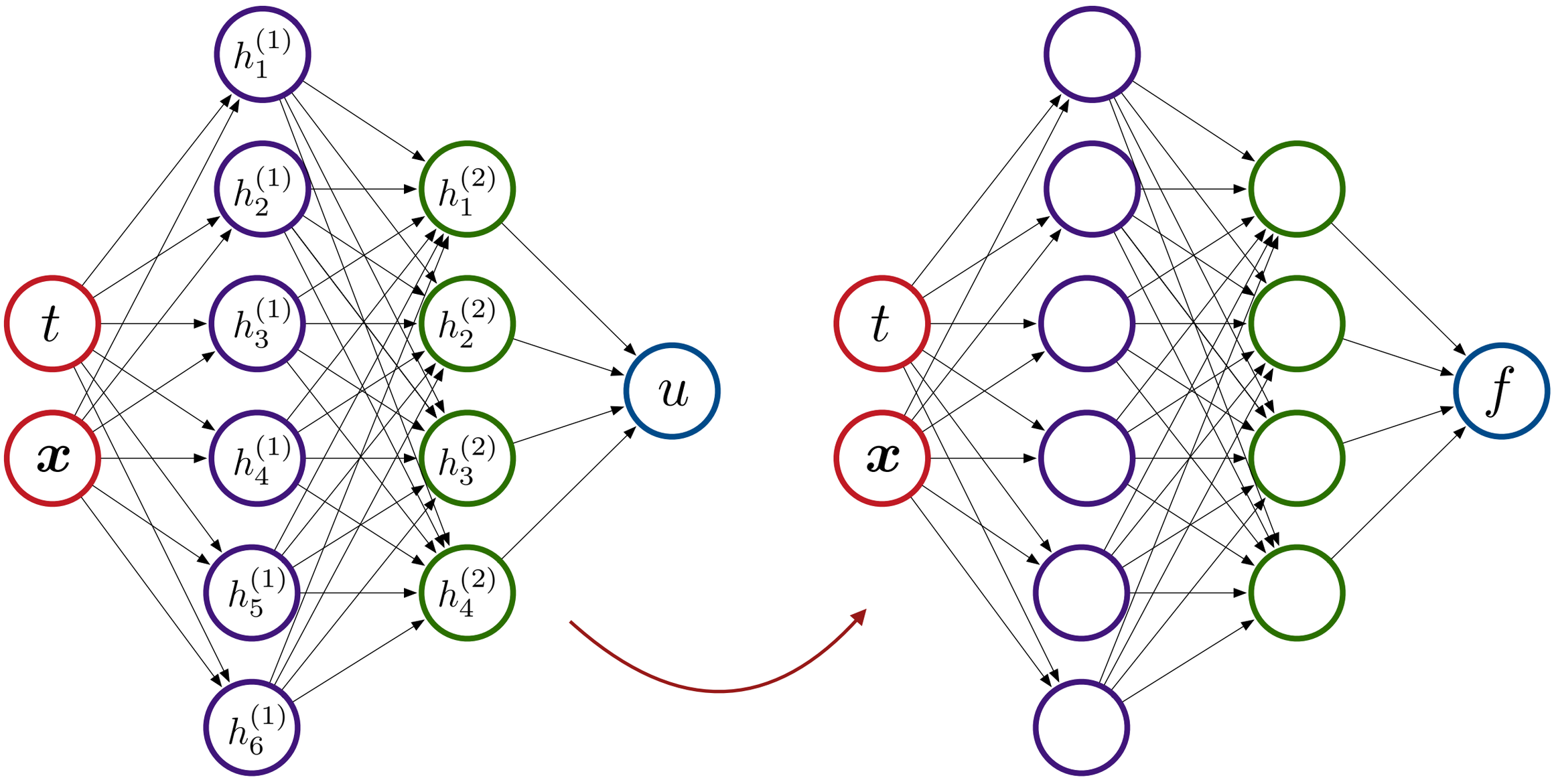
	Clean data			1% noise			5% noise		
	λ_1	λ_2	λ_3	λ_1	λ_2	λ_3	λ_1	λ_2	λ_3
First quartile	0.9603	0.9829	0.9711	0.7871	0.8095	0.5891	−0.0768	0.0834	−0.0887
Median	0.9885	1.0157	0.9970	0.8746	0.9124	0.8798	0.4758	0.5539	0.4086
Third quartile	1.0187	1.0550	1.0314	0.9565	0.9948	0.9553	0.6991	0.7644	0.7009

Raissi, M., & Karniadakis, G. E. (2018). Hidden physics models: Machine learning of nonlinear partial differential equations. Journal of Computational Physics, 357, 125-141.

Physics-informed neural networks

$u(t, \boldsymbol{x})$

$f = u_t + \mathcal{N}[u; \lambda]$



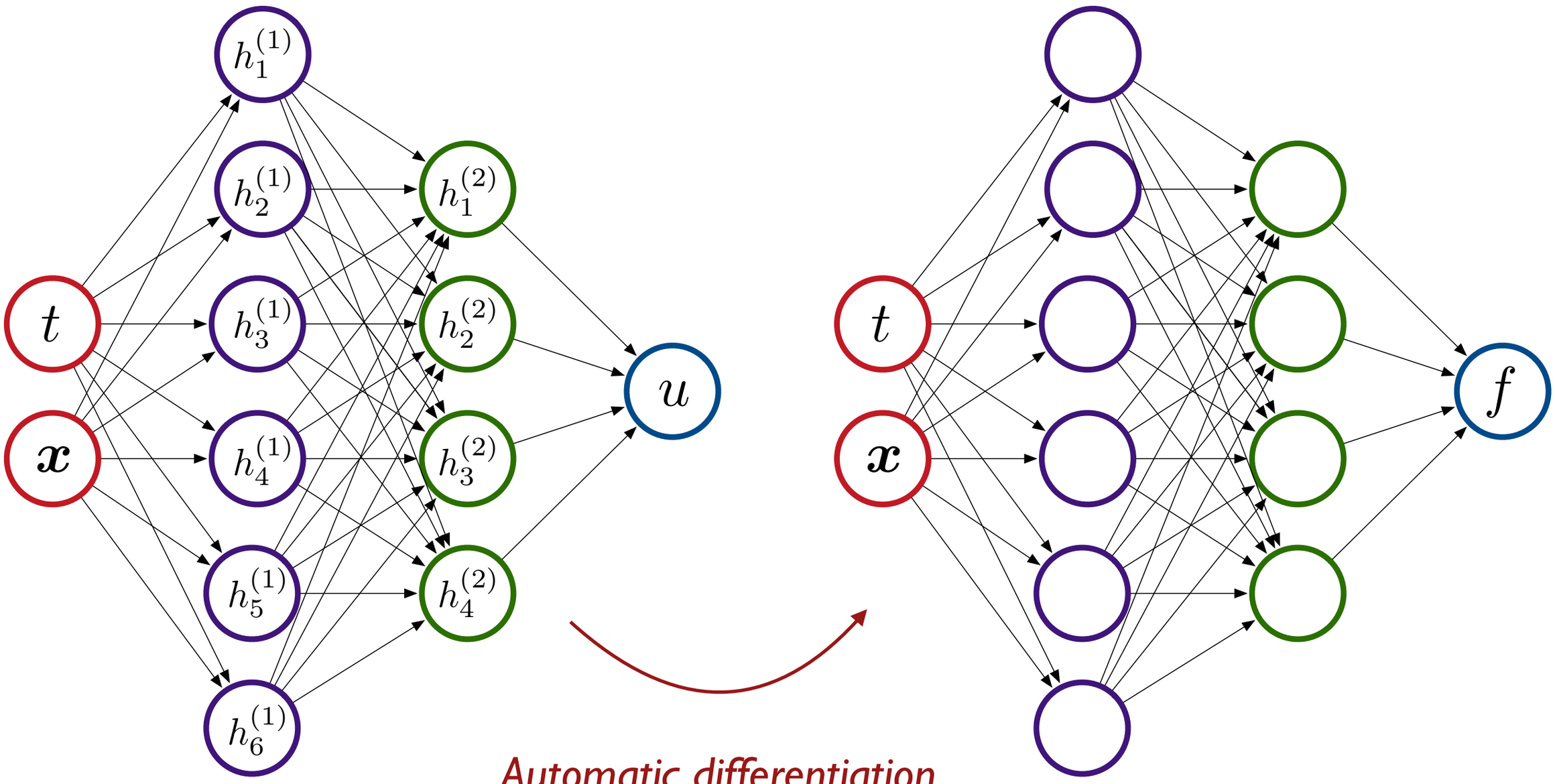
…some old ideas revisited with modern computational tools:

- *Psichogios, D. C., & Ungar, L. H. (1992). A hybrid neural network–first principles approach to process modeling. AIChE Journal, 38(10), 1499-1511.*
- *Lagaris, I. E., Likas, A., & Fotiadis, D. I. (1997). Artificial neural networks for solving ordinary and partial differential equations. arXiv preprint physics/9705023.*

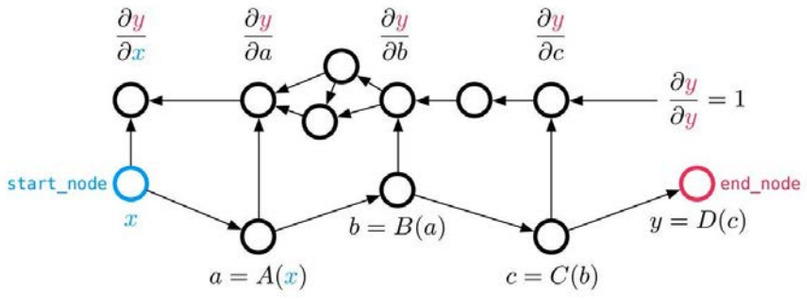
Physics-informed neural networks

$u(t, \boldsymbol{x})$

$f = u_t + \mathcal{N}[u; \lambda]$



Automatic differentiation



TensorFlow™

PYTORCH

Data-driven solution of PDEs

$$u_t + \mathcal{N}[u;\lambda] = 0,\ x \in \Omega,\ t \in [0,T]$$

We define $f(t,x)$ to be given by the left-hand-side of equation (1); i.e.,

$$f := u_t + \mathcal{N}[u;\lambda], \tag{2}$$

and proceed by approximating $u(t,x)$ by a deep neural network. This assumption along with equation (2) result in a *physics informed neural network* $f(t,x)$. This network can be derived by applying the chain rule for differentiating compositions of functions using automatic differentiation [11]. It is worth highlighting that the parameters of the differential operator λ turn into parameters of the *physics informed neural network* $f(t,x)$.

$$MSE = MSE_u + MSE_f,$$

$$MSE_u = \frac{1}{N}\sum_{i=1}^{N} |u(t_u^i, x_u^i) - u^i|^2,$$

$$MSE_f = \frac{1}{N}\sum_{i=1}^{N} |f(t_u^i, x_u^i)|^2.$$

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. arXiv preprint arXiv

Physics-informed neural networks

Example: Burgers' equation in 1D

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \tag{3}$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$

Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - (0.01/\pi)u_{xx},$$

```
def u(t, x):
    u = neural_net(tf.concat([t,x],1), weights, biases)
    return u
```

Correspondingly, the *physics informed neural network* $f(t, x)$ takes the form

```
def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. arXiv preprint arXiv

Physics-informed neural networks

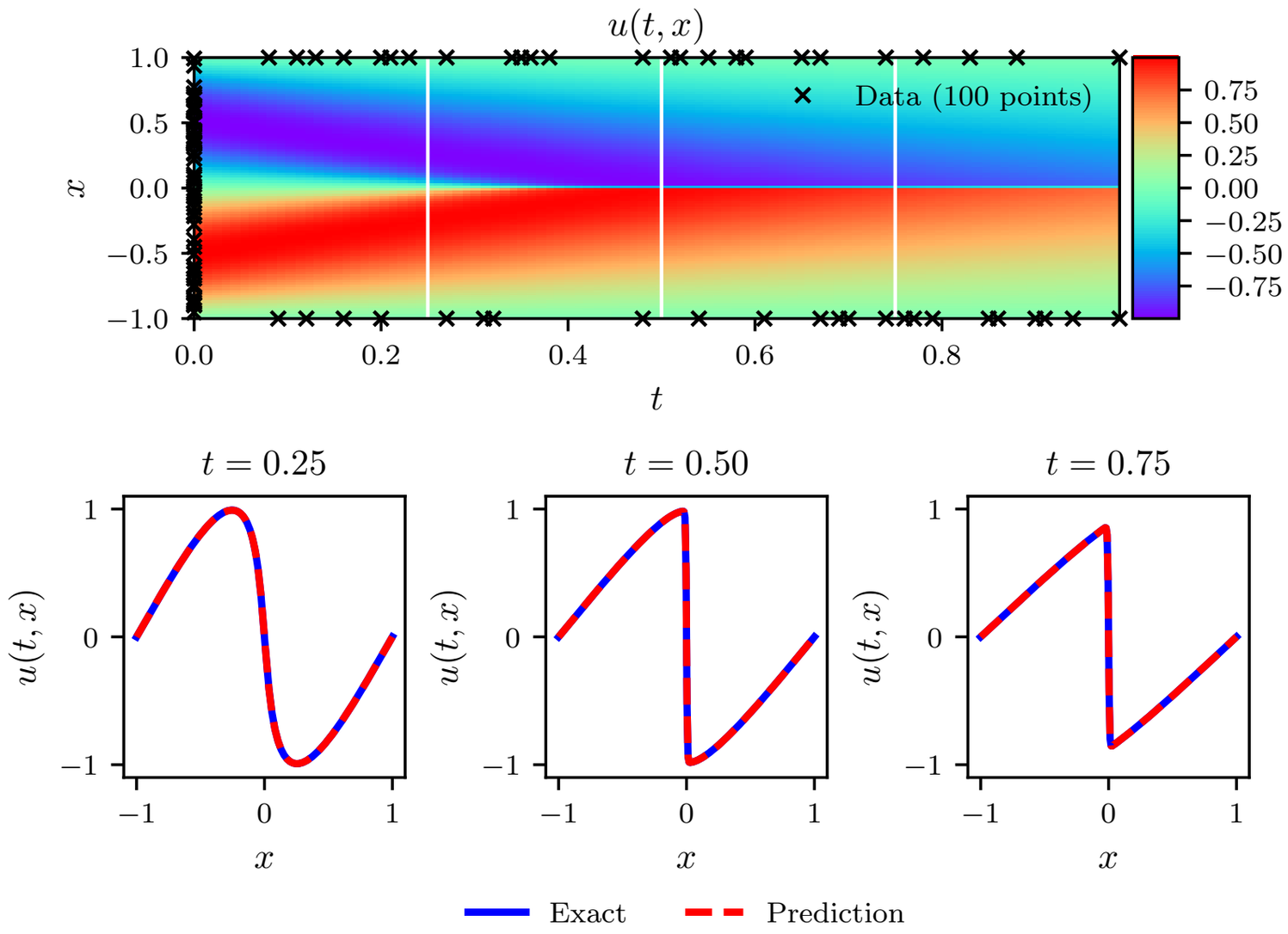
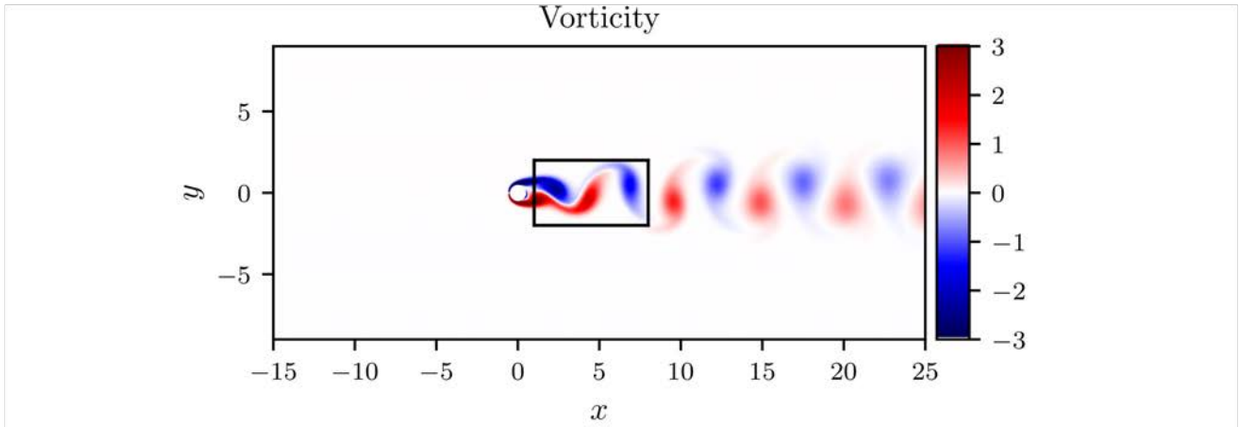
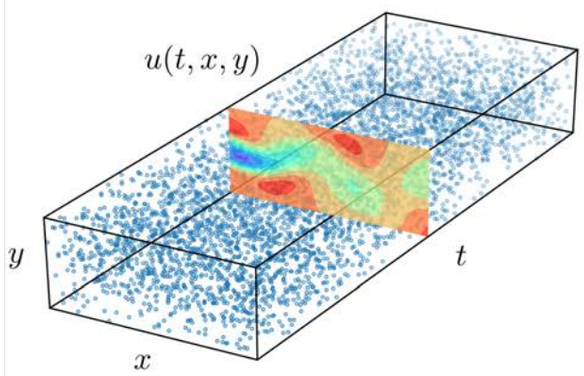


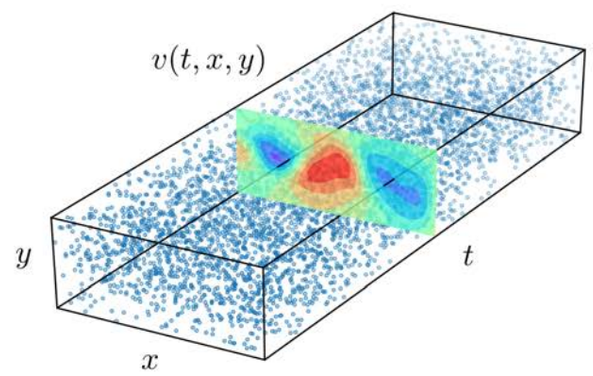
Figure 1: *Burgers' equation: Top:* Predicted solution $u(t,x)$ along with the initial and boundary training data. In addition we are using 10,000 collocation points generated using a Latin Hypercube Sampling strategy. *Bottom:* Comparison of the predicted and exact solutions corresponding to the three temporal snapshots depicted by the white vertical lines in the top panel. The relative $\mathcal{L}_2$ error for this case is $6.7 \cdot 10^{-4}$. Model training took approximately 60 seconds on a single NVIDIA Titan X GPU card.

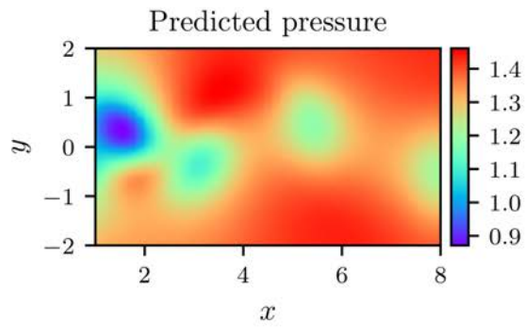
Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. arXiv preprint arXiv

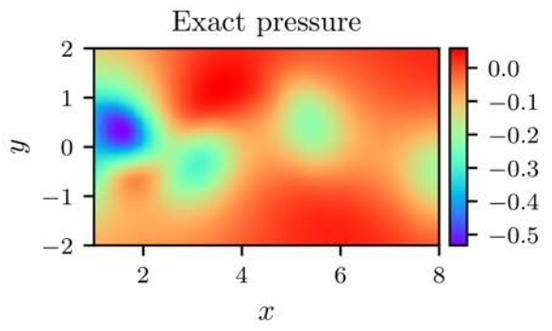
Physics-informed neural networks











Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy})$ $v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$

Data — *Models*

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. arXiv preprint

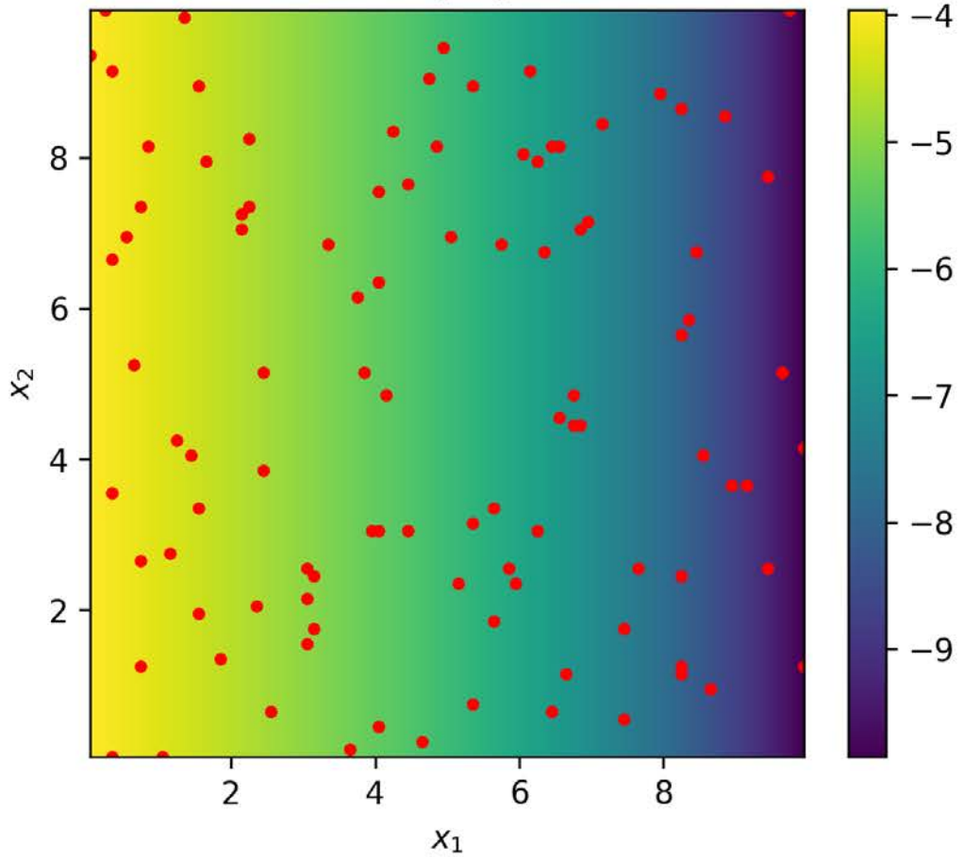
Learning constitutive relationships

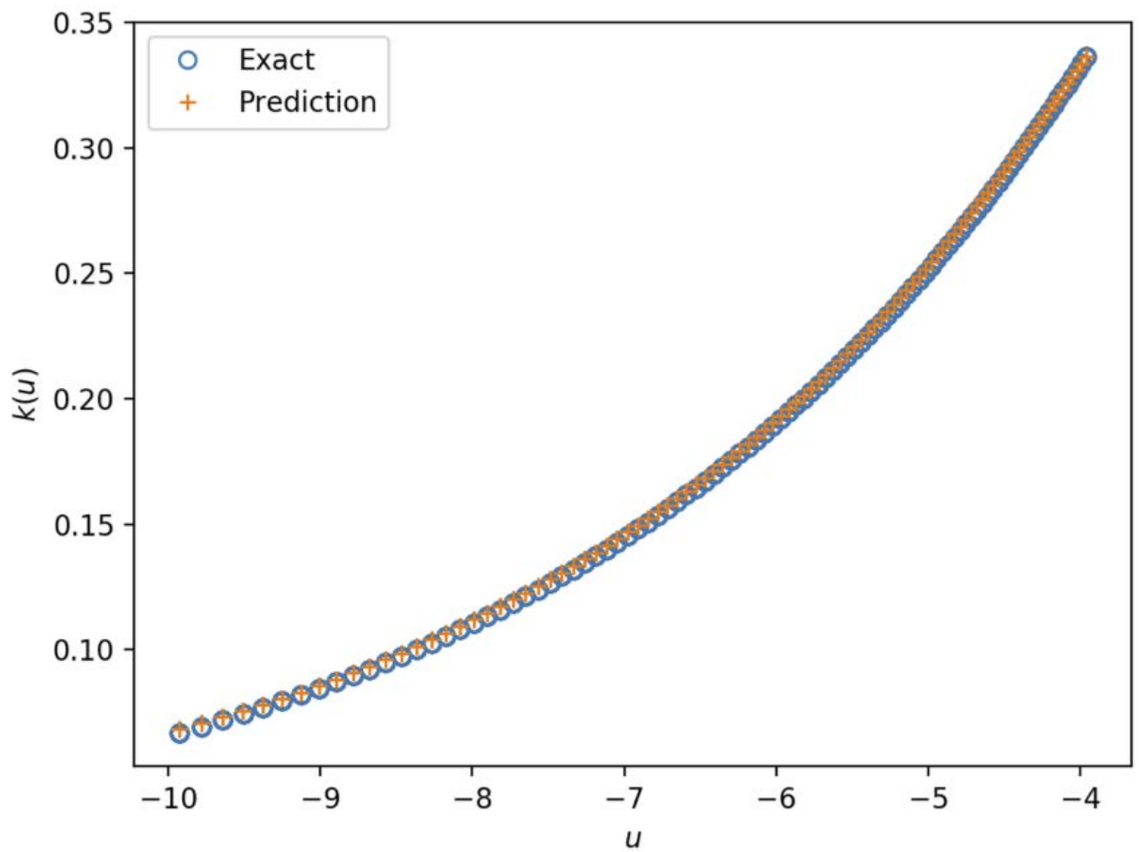
Non-linear diffusion: $\nabla \cdot [K(u)\nabla u(\mathbf{x})] = 0, \quad (x_1, x_2) \in (0, L_1) \times (0, L_2)$

subject to the boundary conditions

$$
\begin{aligned}
u(\mathbf{x}) &= u_0, && x_1 = L_1, \\
-K(u)\frac{\partial u(\mathbf{x})}{\partial x_1} &= q, && x_1 = 0 \\
\frac{\partial u(\mathbf{x})}{\partial x_2} &= 0, && x_2 = \{0, L_2\}.
\end{aligned}
$$

Goal: Infer k(u) from scattered noisy measurements of u(x).

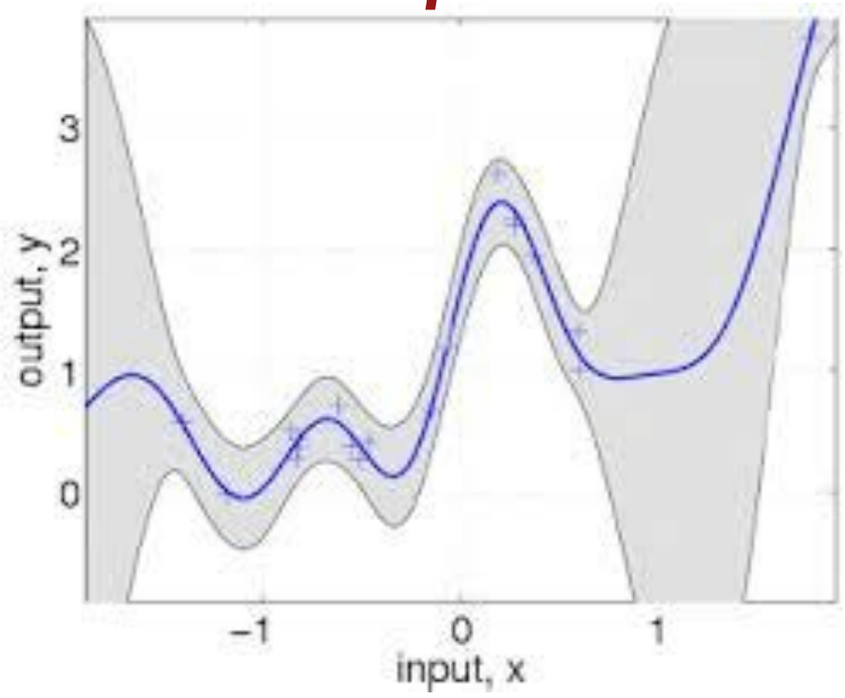




Tartakovsky, A. M., Marrero, C. O., Perdikaris, P., Tartakovsky, G., & Barajas-Solano, D. (2018). Learning Parameters and Constitutive Relationships with Physics Informed Deep Neural Networks. arXiv preprint arXiv:1808.03398

Probabilistic representations

Gaussian processes



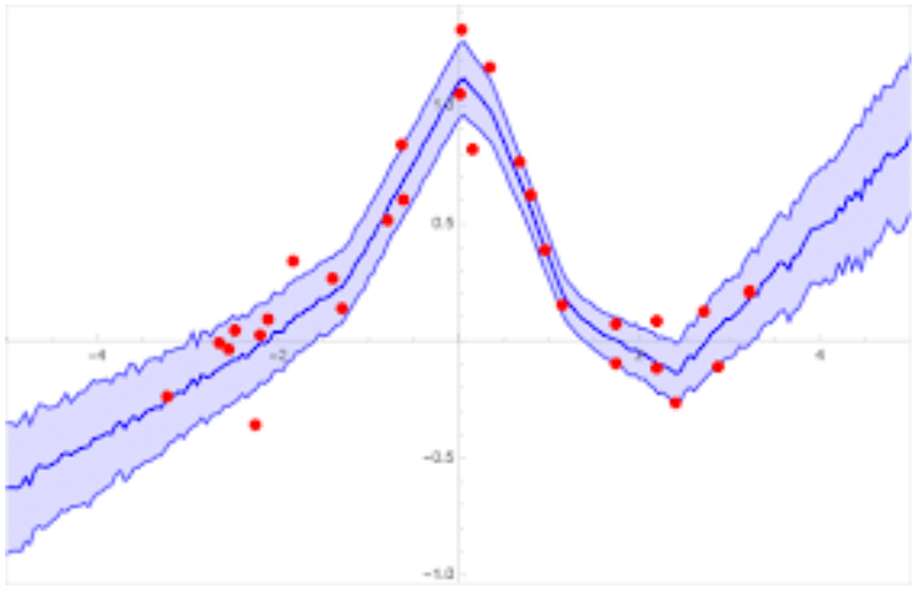
Capabilties:

- Elegant mathematical formulation and exact inference
- Flexible encoding of (linear) constraints
- Robust in small data regimes
- Built-in uncertainty quantification that enables effective policies for data acquisition/experimental design

Limitations:

- Limited expressivity
- Scalability to large data-sets
- Scalability to high-dimensions
- Robustness with respect to prior assumptions

Bayesian neural networks



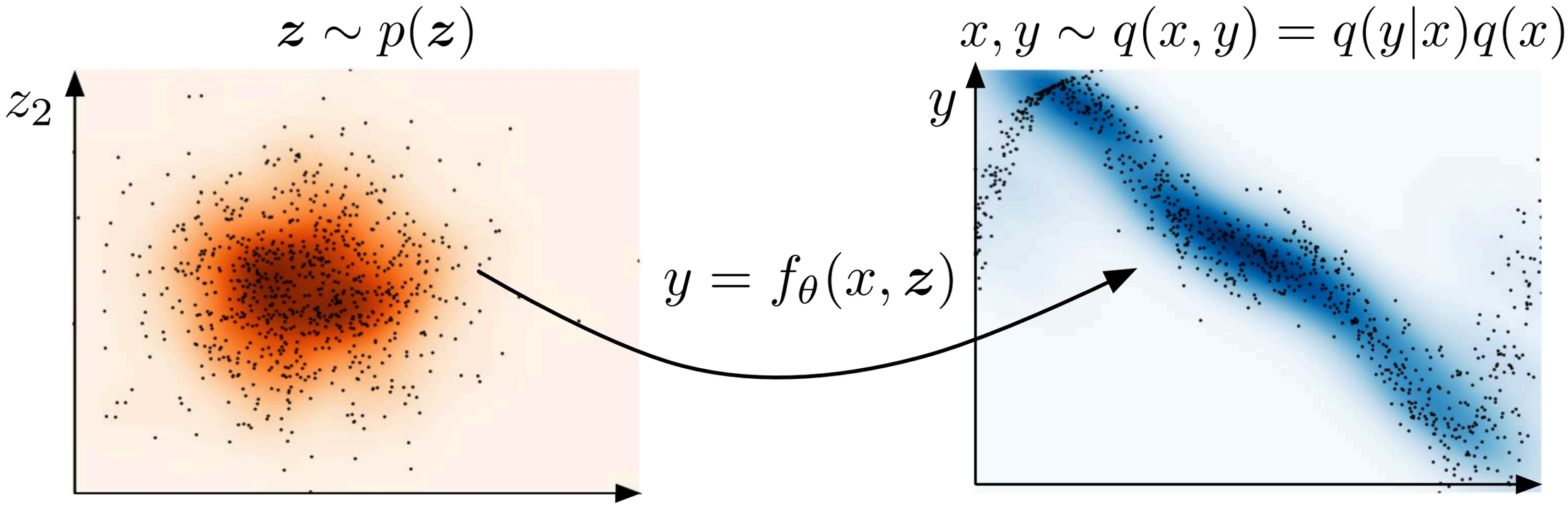
Capabilties:

- Highly flexible representations
- Scalable to high-dimensional data
- Scalable to large data sets
- Flexible encoding of (algebraic & differential) constraints

Limitations:

- Complex high-dimensional posteriors and intractable inference
- Robustness of uncertainty estimates
- Robustness in small data regimes
- Interpretability

Probabilistic latent variable models



Latent space ***Physical space***

$$y = f_\theta(x, \boldsymbol{z}),\ \boldsymbol{z} \sim p(\boldsymbol{z}) \ \Leftrightarrow\ y \sim p_\theta(y|x, \boldsymbol{z})$$

vs

$$y = f_\theta(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I)$$

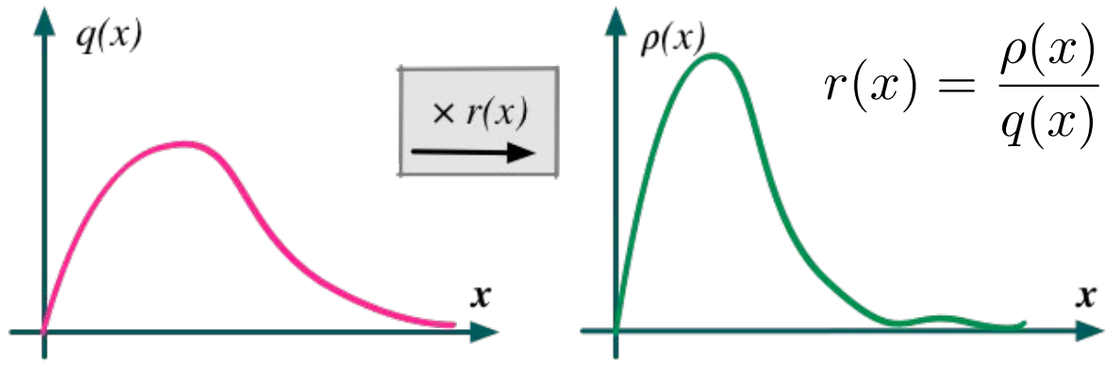
Density ratio estimation by probabilistic classification

$$\mathbb{KL}[p(\boldsymbol{x})||q(\boldsymbol{x})] := \int \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} p(\boldsymbol{x}) d\boldsymbol{x} = \mathbb{E}_{p(\boldsymbol{x})} \left[\log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \right]$$

Estimating density ratios is a challenging task:

- Each part of the ratio may itself involve intractable integrals
- We often deal with high-dimensional quantities.
- We may only have samples drawn from the two distributions, not their analytical forms.

This is where the ***density ratio trick*** enters: it allows us to construct a binary classifier that distinguishes between samples from the two distributions.



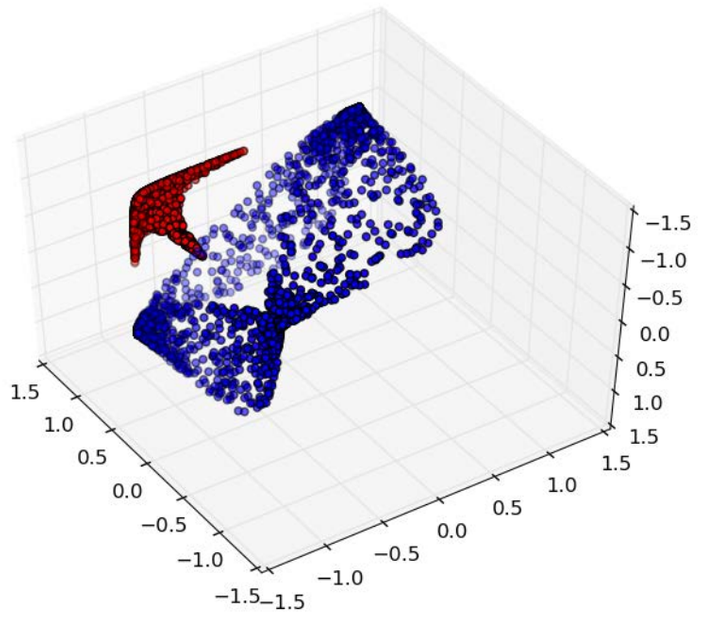
$$r(x) = \frac{\rho(x)}{q(x)}$$

The density ratio gives the correction factor needed to make two distributions equal.

$$r(x) = \frac{\rho(x)}{q(x)} = \frac{p(x|y=+1)}{p(x|y=-1)}$$

$$= \frac{p(y=+1|x)p(x)}{p(y=+1)} \Big/ \frac{p(y=-1|x)p(x)}{p(y=-1)}$$

$$= \frac{p(y=+1|x)}{p(y=-1|x)} = \frac{p(y=+1|x)}{1-p(y=+1|x)} = \frac{\mathcal{S}(x)}{1-\mathcal{S}(x)}$$



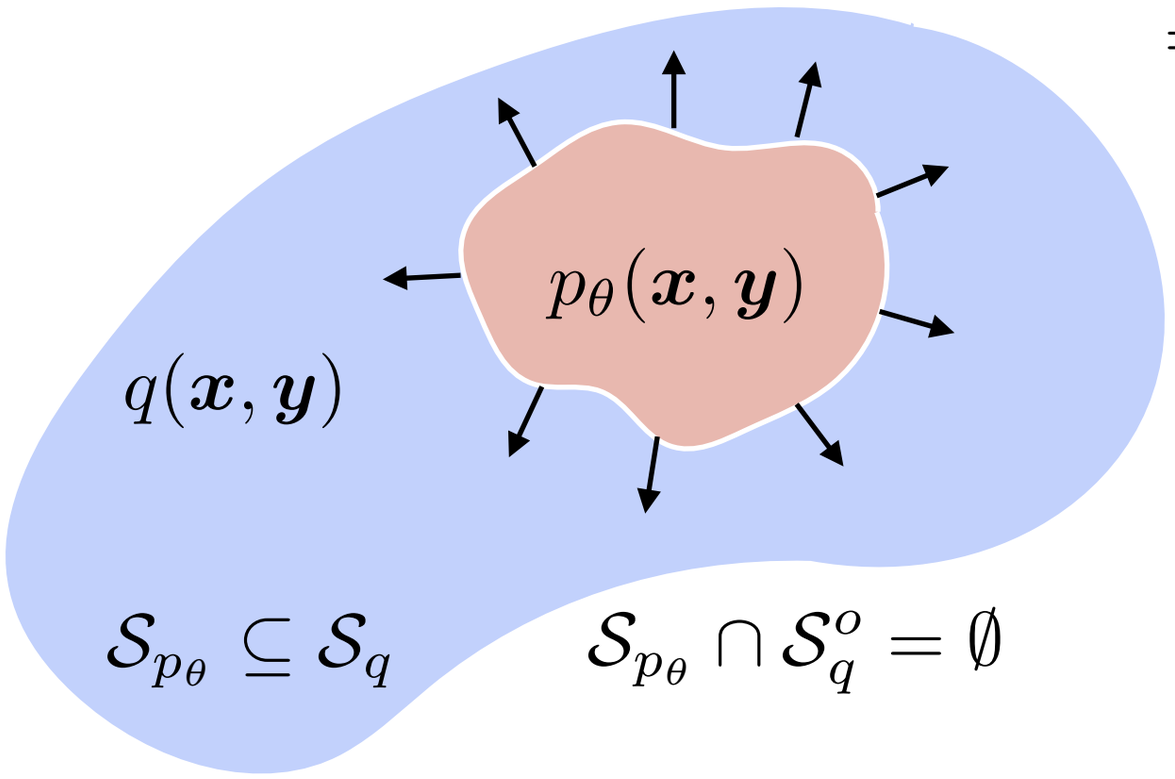
Joint distribution matching

Reverse KL: $\mathbb{KL}\left[p_\theta(\boldsymbol{x},\boldsymbol{y})||q(\boldsymbol{x},\boldsymbol{y})\right] = -\mathbb{H}\left[p_\theta(\boldsymbol{x},\boldsymbol{y})\right] - \mathbb{E}_{p_\theta(\boldsymbol{x},\boldsymbol{y})}\left[\log q(\boldsymbol{x},\boldsymbol{y})\right]$

$$= -\mathbb{H}\left[p_\theta(\boldsymbol{x},\boldsymbol{y})\right]$$

$$- \int_{\mathcal{S}_{p_\theta}\cap\mathcal{S}_q} \log q(\boldsymbol{x},\boldsymbol{y}) p_\theta(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y}$$

$$- \int_{\mathcal{S}_{p_\theta}\cap\mathcal{S}_q^o} \log q(\boldsymbol{x},\boldsymbol{y}) p_\theta(\boldsymbol{x},\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y}$$



$q(\boldsymbol{x},\boldsymbol{y})$: ***Observed data distribution***

$p_\theta(\boldsymbol{x},\boldsymbol{y})$: ***Generated data distribution***

$f_\theta(\boldsymbol{x},\boldsymbol{z}) \longrightarrow$ ***generator***

$T_\psi(\boldsymbol{x},\boldsymbol{y}) \longrightarrow$ ***discriminator***

$q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) \longrightarrow$ ***encoder***

$$\min_\psi -\mathbb{E}_{p(\boldsymbol{z})}[\log(T_\psi(\boldsymbol{x}, f_\theta(\boldsymbol{x},\boldsymbol{z})))] - \mathbb{E}_{q(\boldsymbol{x},\boldsymbol{y})}[\log(1 - T_\psi(\boldsymbol{x},\boldsymbol{y})]$$

$$\min_{\theta,\phi} \mathbb{E}_{p(\boldsymbol{z})}[T_\psi(\boldsymbol{x}, f_\theta(\boldsymbol{x},\boldsymbol{z}))] + (1-\lambda)\mathbb{E}_{p(\boldsymbol{z})}[\log q_\phi(\boldsymbol{z}|\boldsymbol{x}, f_\theta(\boldsymbol{x},\boldsymbol{z}))]$$

Adversarial objective

Li, C., Li, J., Wang, G., & Carin, L. (2018). Learning to Sample with Adversarially Learned Likelihood-Ratio.

Uncertainty propagation in physical systems

Inputs
(random)

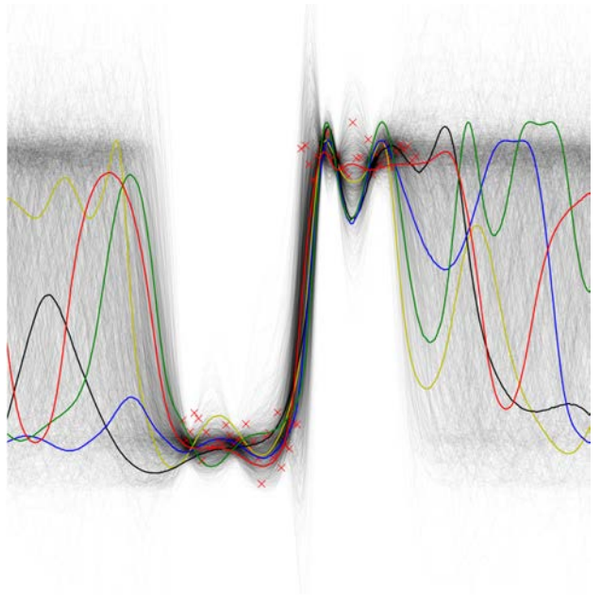
$\boldsymbol{z} \sim p(\boldsymbol{z})$ →

$\boldsymbol{x}, t$ →

Inputs
(deterministic)

$$\boldsymbol{u}_t + \mathcal{N}_{\boldsymbol{x}}\boldsymbol{u} = 0$$

→



$p(\boldsymbol{u}|\boldsymbol{x}, t, \boldsymbol{z})$

Outputs

Adversarial UQ in physics-informed neural networks

Model:

$$p(\boldsymbol{u}|\boldsymbol{x},t,\boldsymbol{z}),\quad \boldsymbol{z}\sim p(\boldsymbol{z}),\quad \text{s.t}\quad \boldsymbol{u}_t+\mathcal{N}_{\boldsymbol{x}}\boldsymbol{u}=0$$

Training:

$$\max_{\psi}\ \mathcal{L}_{\mathcal{D}}(\psi)$$

$$\min_{\theta,\phi}\ \mathcal{L}_{\mathcal{G}}(\theta,\phi)+\beta\mathcal{L}_{PDE}(\theta)$$

$$\mathcal{L}_{\mathcal{D}}(\psi)=\mathbb{E}_{q(\boldsymbol{x},t)p(\boldsymbol{z})}[\log\sigma(T_\psi(\boldsymbol{x},t,f_\theta(\boldsymbol{x},t,\boldsymbol{z})))]+\mathbb{E}_{q(\boldsymbol{x},t,\boldsymbol{u})}[\log(1-\sigma(T_\psi(\boldsymbol{x},t,\boldsymbol{u})))]$$

$$\mathcal{L}_{\mathcal{G}}(\theta,\phi)=\mathbb{E}_{q(\boldsymbol{x},t)p(\boldsymbol{z})}[T_\psi(\boldsymbol{x},t,f_\theta(\boldsymbol{x},t,\boldsymbol{z}))+(1-\lambda)\log(q_\phi(\boldsymbol{z}|\boldsymbol{x},t,f_\theta(\boldsymbol{x},t,\boldsymbol{z})))]$$

$$\mathcal{L}_{PDE}(\theta)=\frac{1}{N_r}\sum_{i=1}^{N_r}[\boldsymbol{r}_\theta(\boldsymbol{x}_i,t_i)-\boldsymbol{r}_i]^2$$

Prediction:

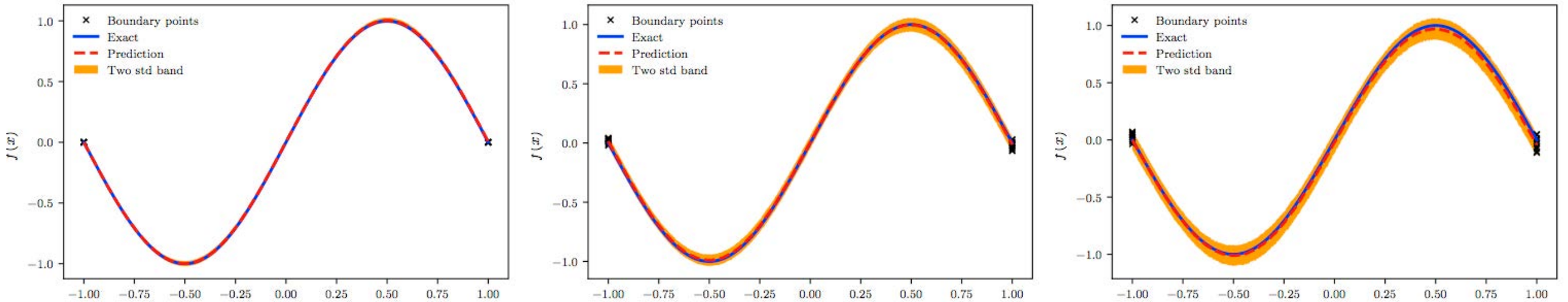
$$\mu_{\boldsymbol{u}}(\boldsymbol{x}^*,t^*)=\mathbb{E}_{p_\theta}[\boldsymbol{u}|\boldsymbol{x}^*,t^*,\boldsymbol{z}]\approx\frac{1}{N_s}\sum_{i=1}^{N_s}f_\theta(\boldsymbol{x}^*,t^*,\boldsymbol{z}_i),$$

$$\sigma^2_{\boldsymbol{u}}(\boldsymbol{x}^*,t^*)=\mathbb{V}\text{ar}_{p_\theta}[\boldsymbol{u}|\boldsymbol{x}^*,t^*,\boldsymbol{z}]\approx\frac{1}{N_s}\sum_{i=1}^{N_s}[f_\theta(\boldsymbol{x}^*,t^*,\boldsymbol{z}_i)-\mu_{\boldsymbol{u}}(\boldsymbol{x}^*,t^*)]^2$$

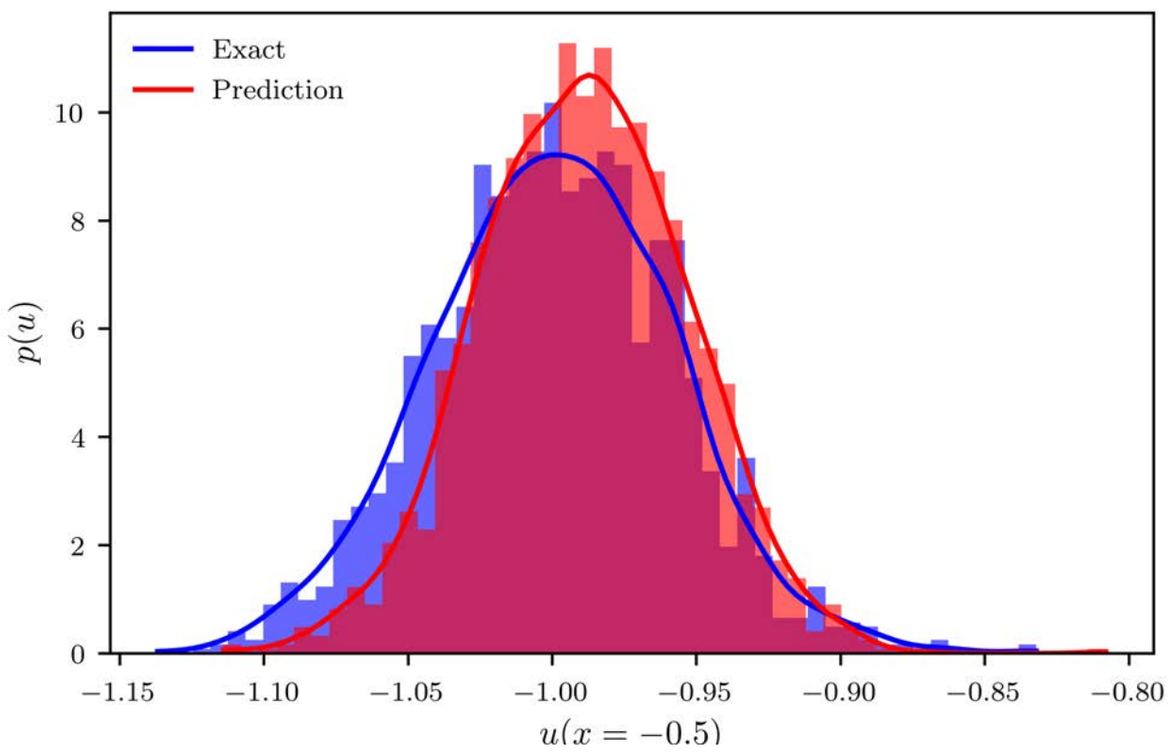
Adversarial UQ in physics-informed neural networks

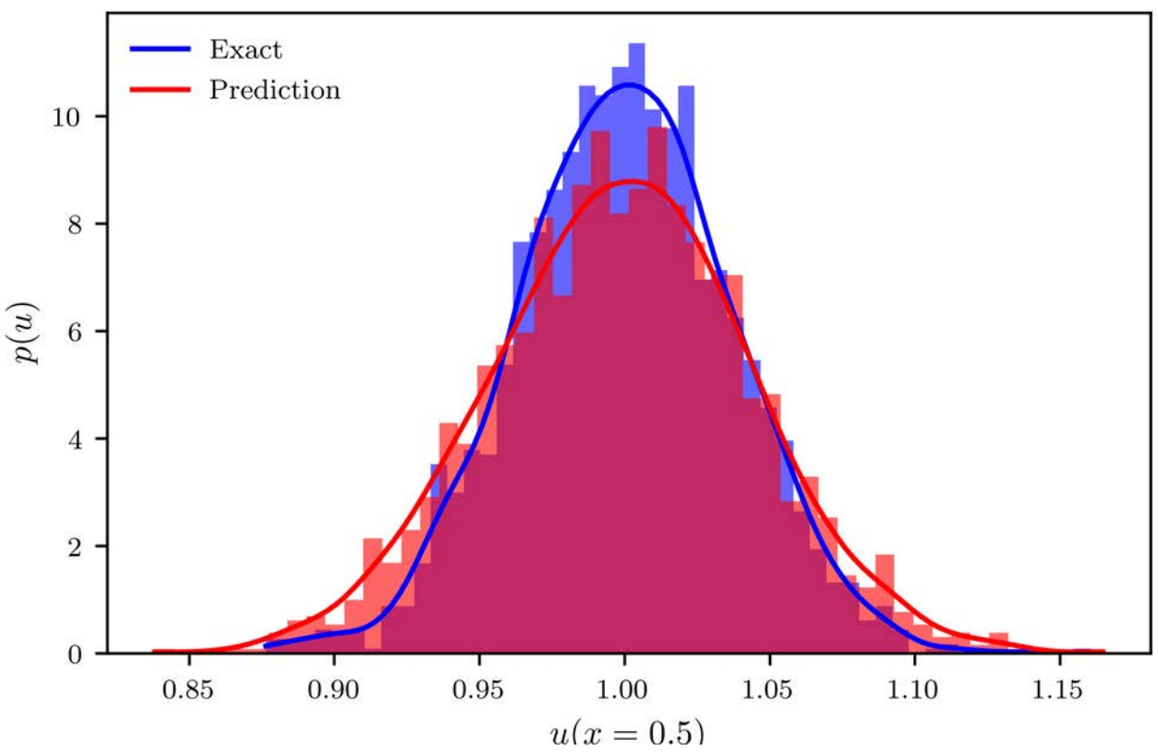
Example in 1D: $u_{xx} - u^2 u_x = f(x)$ *subject to random boundary conditions*

Noise-free data *3% noise* *5% noise*



(a) (b) (c)



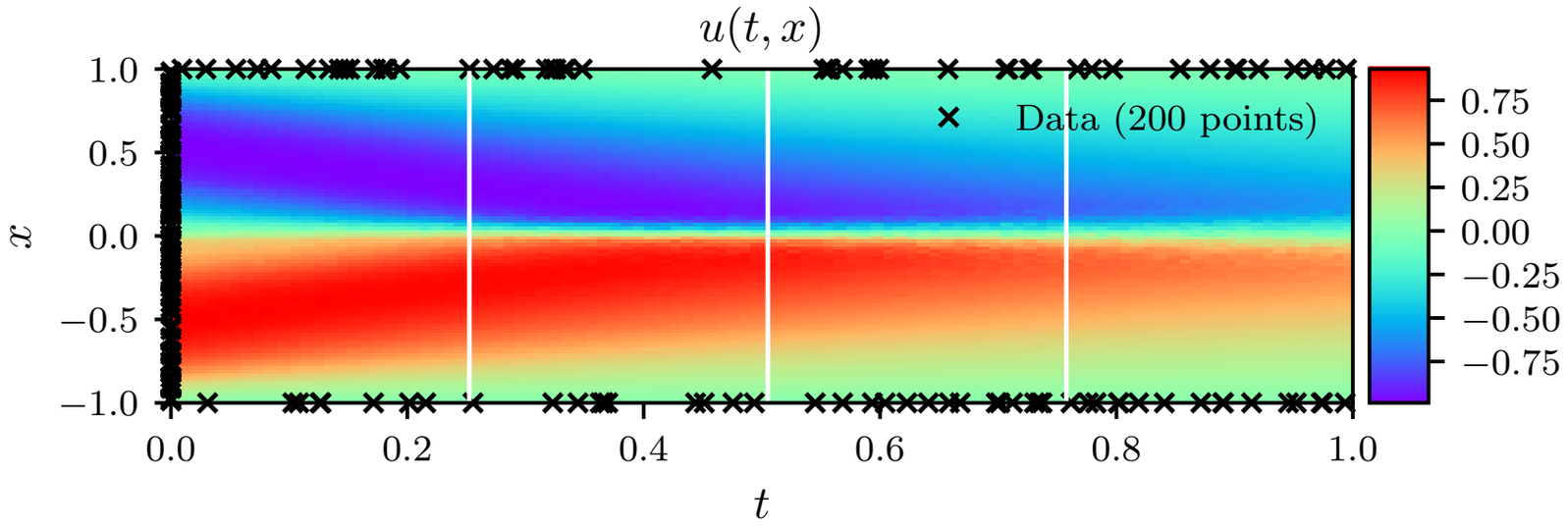


Adversarial UQ in physics-informed neural networks

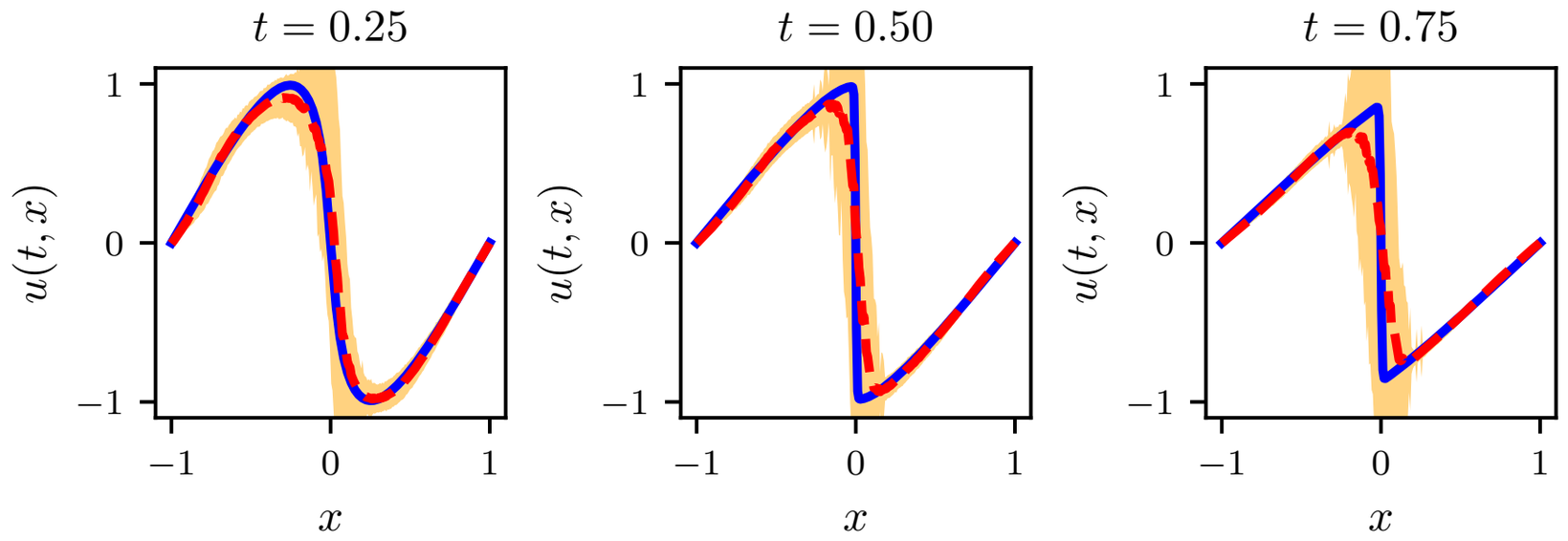
$$u = f_\theta(x, t, \boldsymbol{z}), \boldsymbol{z} \sim p(\boldsymbol{z})$$

Burgers equation:

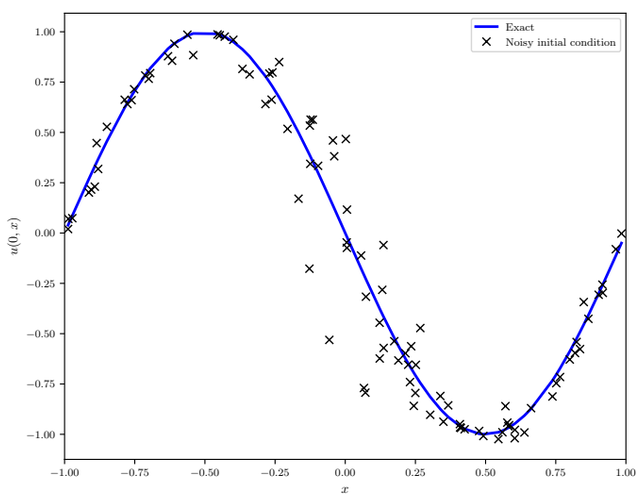
$$u_t + uu_x - (0.01/\pi)u_{xx} = 0,$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$

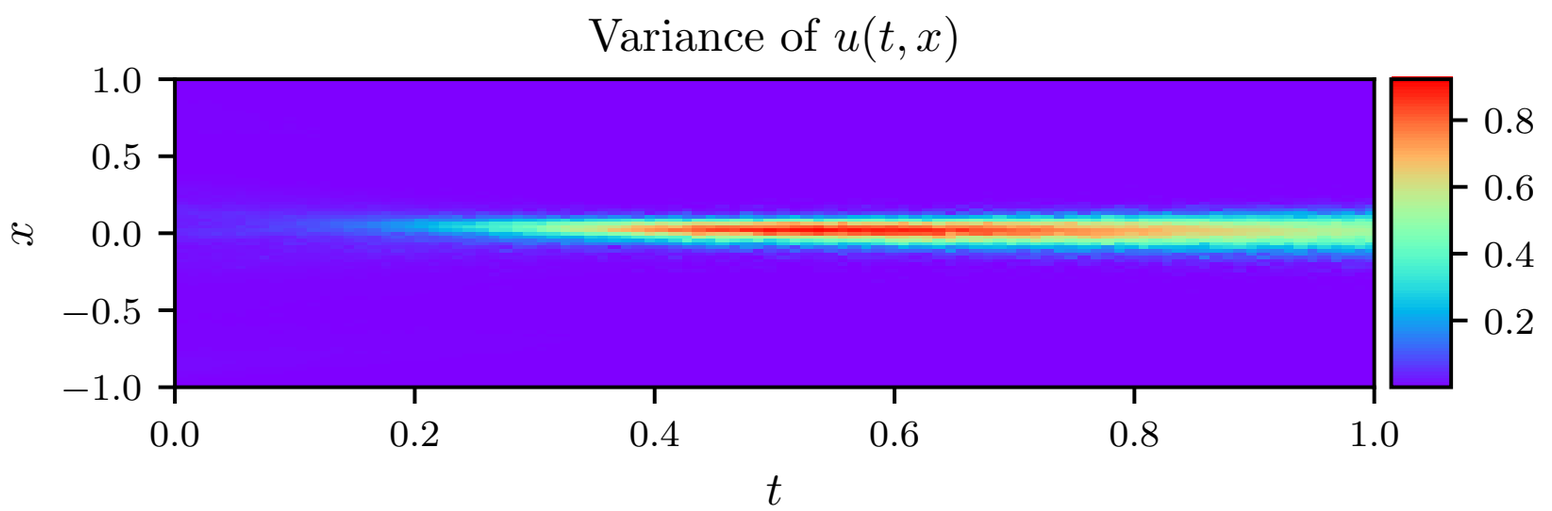


$$p_\theta(u|x, t, \boldsymbol{z})$$



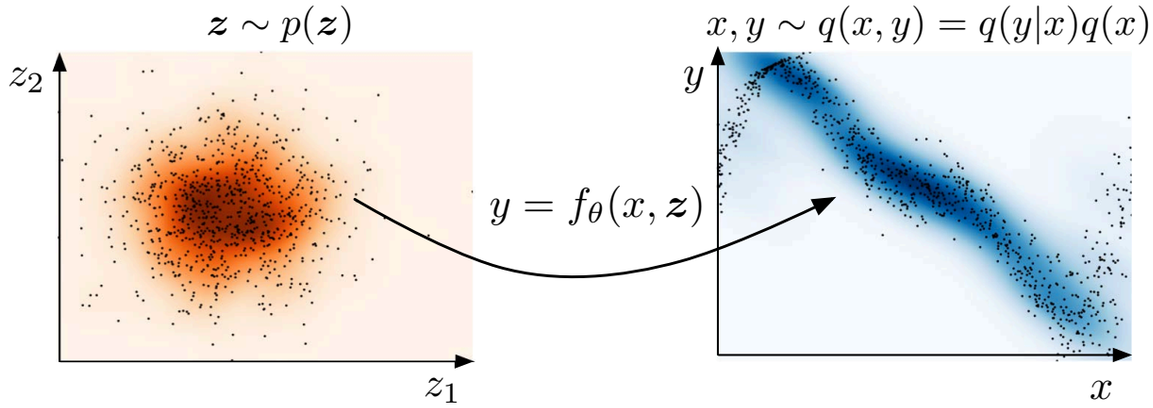
Noisy initial condition



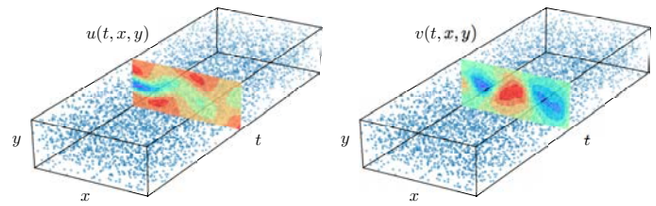


We develop application-driven computational tools for modeling, analyzing, and optimizing complex systems using:

- First mathematical principles
- Machine learning
- High-performance computing

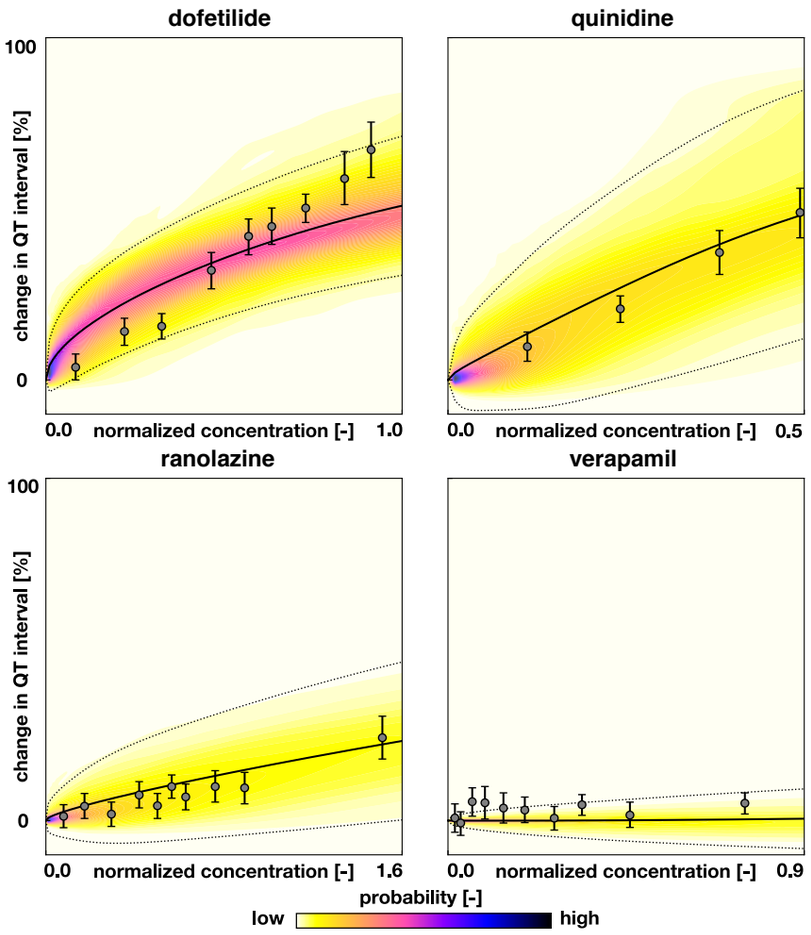


$$y = f_\theta(x, z), \;\; z \sim p(z) \;\; \Leftrightarrow \;\; y \sim p_\theta(y|x, z)$$

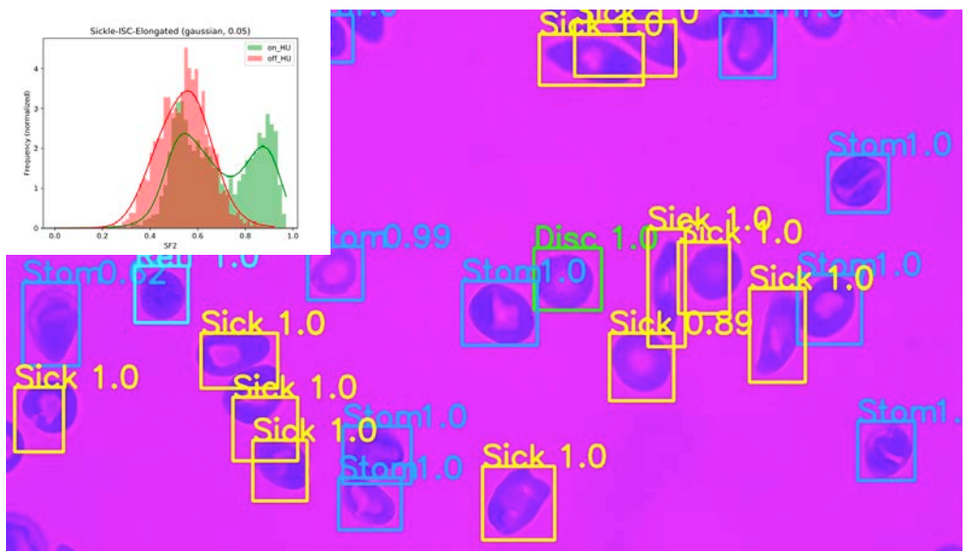


$$\underbrace{p(\boldsymbol{z}|\boldsymbol{u}, \boldsymbol{x})}_{\text{posterior}} = \frac{\overbrace{p(\boldsymbol{u}|\boldsymbol{x}, \boldsymbol{z})}^{\text{likelihood}} \overbrace{p(\boldsymbol{z}|\boldsymbol{x})}^{\text{prior}}}{\underbrace{p(\boldsymbol{u}|\boldsymbol{x})}_{\text{marginal likelihood}}},$$

Data-driven discovery using physics-informed deep learning



Uncertainty quantification in drug-induced arrhythmias



Deep learning for personalized risk assessment in health and disease

Data ⟷ Models

Questions?

Funding:

U.S. DEPARTMENT OF ENERGY

Collaborators:

Yibo Yang (UPenn)
Alex Tartakovsky (PNNL)
George Karniadakis (Brown)
Maziar Raissi (Brown)
Luca Bonfiglio (MIT)

Email: pgp@seas.upenn.edu