ENM 540: Data-driven modeling and probabilistic scientific computing

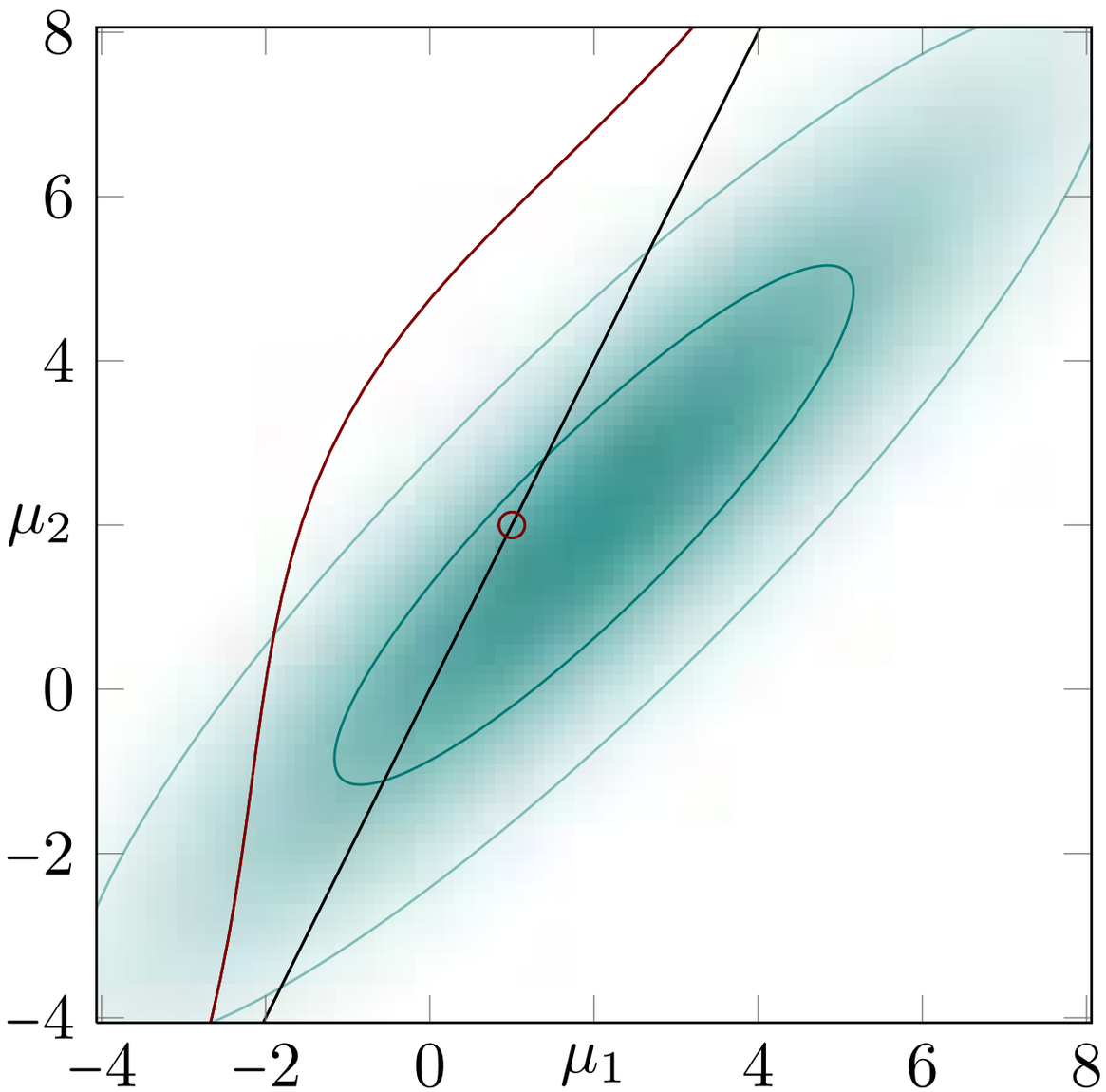
Probabilistic scientific computing with Gaussian processes

Paris Perdikaris

March 20, 2018

Closure under Linear Maps

Linear Maps of Gaussians are Gaussians



$$p(z) = \mathcal{N}(z; \mu, \Sigma)$$

$$\Rightarrow \quad p(Az) = \mathcal{N}(Az, A\mu, A\Sigma A^\top)$$

Here: $A = [1, -0.5]$

http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

Physics informed machine learning

$$\mathcal{L}_x u_2(x) = f_2(x)$$

$$\frac{\partial}{\partial x} u_2(x) + \int_0^x u_2(\xi) d\xi = f(x)$$

$$\sum_{d=1}^{10} \frac{\partial^2}{\partial x_d^2} u(x) = f(x)$$

$$u_2(t,x) + \frac{\partial}{\partial x} u_2(t,x) - \frac{\partial^2}{\partial x^2} u_2(t,x) - u_2(t,x) = f(t,x)$$

$$-_\infty D_x^\alpha u_2(x) - u_2(x) = f(x)$$

$$u_2(x) \sim \mathcal{GP}(0, g(x,x';\theta)) \xrightarrow{\textbf{Linearity}} f_2(x) \sim \mathcal{GP}(0, k(x,x';\theta)) \longrightarrow k(x,x';\theta) = \mathcal{L}_x \mathcal{L}_{x'} g(x,x';\theta)$$

Problem setup:

- $f_2(x)$ is an unknown, black box function
- only scattered, noisy, variable fidelity observations of $f_2(x)$ are available
- we have no data on $u_2(x)$ other than the necessary initial/boundary conditions
- no numerical discretization!

"once we accept that we don't know f, but we do know something, it becomes natural to take a Bayesian approach" *P. Diaconis, "Bayesian numerical analysis", 1988*

"stochastic methods will transform pure and applied mathematics in the beginning of the third millennium, as probability and statistics will come to be viewed as the natural tools to use in mathematical as well as scientific modeling" *D. Mumford, "The dawning age of stochasticity", 2000*

Revisiting numerical methods from a statistical inference viewpoint traces all the way back to the Poincare's courses on probability theory!

M. Raissi,., P. Perdikaris, and G.E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, http://128.84.21.199/abs/1607.04805, 2016

Inferring solutions to partial differential equations

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

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

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

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

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). Machine learning of linear differential equations using Gaussian processes. Journal of Computational Physics, 348, 683-693.

Numerical solution of PDEs via machine learning

Priors on $u(x)$, $f(x)$

+

Noisy data on $f(x)$

Posterior on $u(x)$

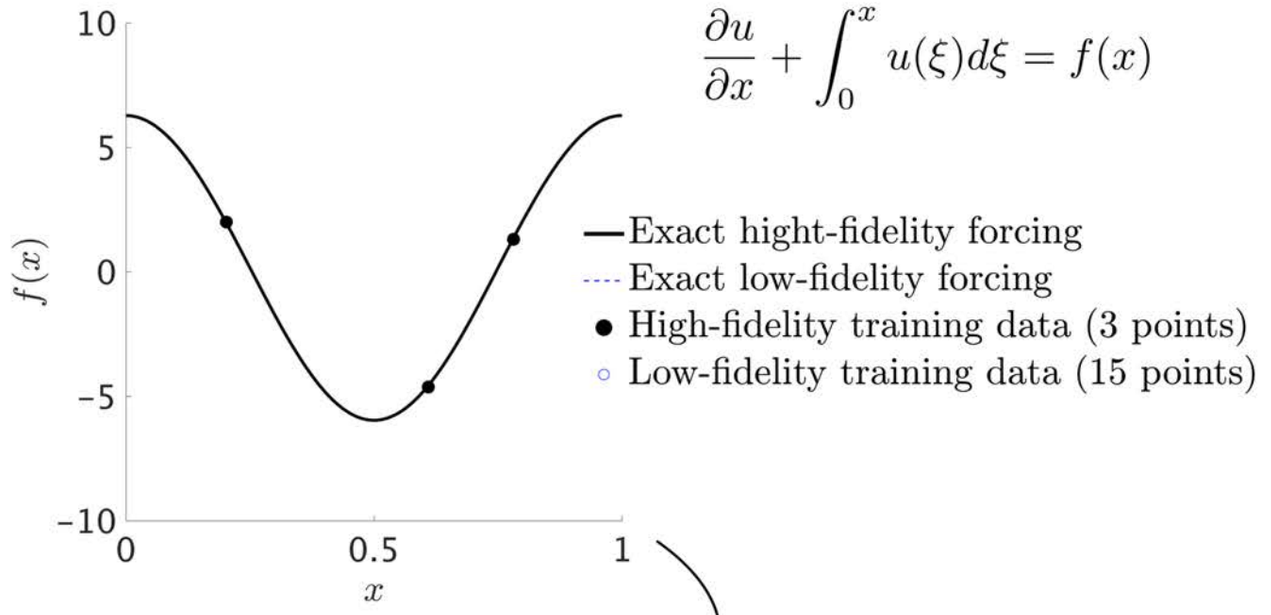
A

$$\left\{ \begin{array}{c} u(x) \sim \mathcal{GP}(0, g(x, x'; \theta)) \\ \mathcal{L}_x u(x) = f \sim \mathcal{GP}(0, k(x, x'; \theta)) \\ k(x, x'; \theta) = \mathcal{L}_x \mathcal{L}_{x'} g(x, x'; \theta) \end{array} \right\}$$

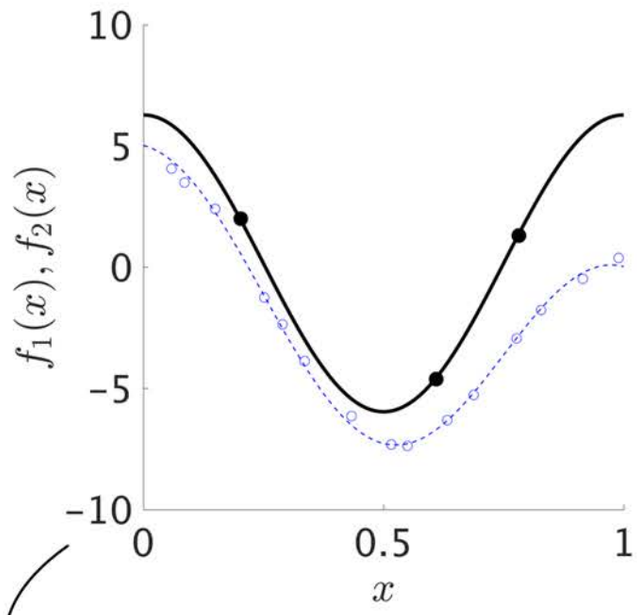
B Single-fidelity regression

1D example:

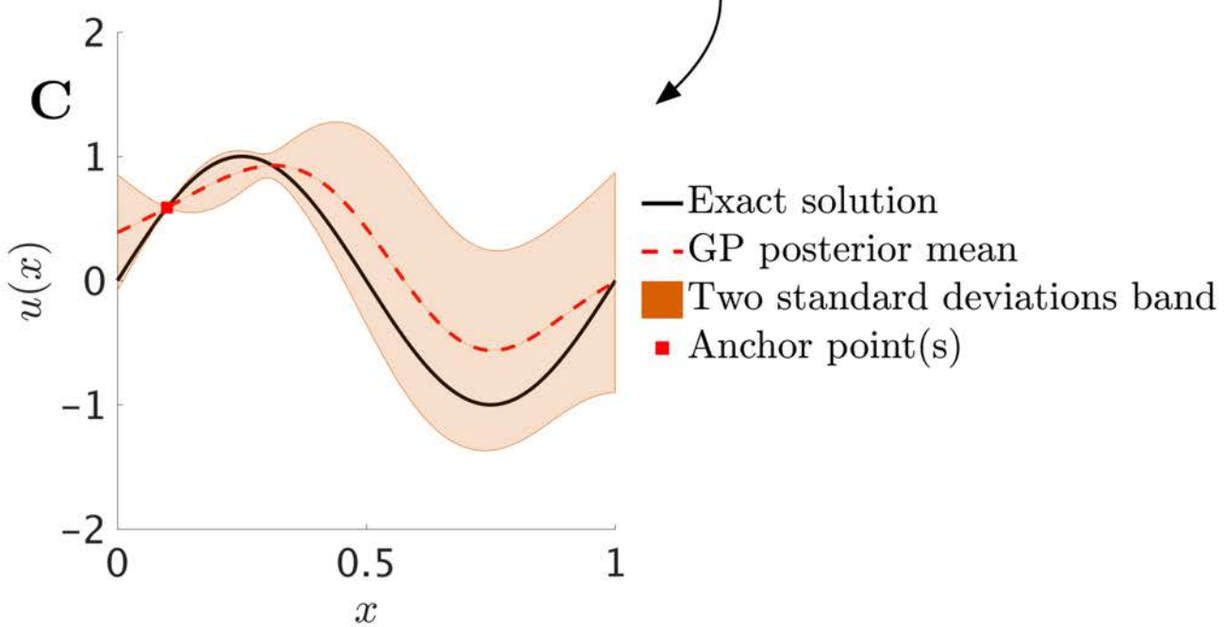
$$\frac{\partial u}{\partial x} + \int_0^x u(\xi) d\xi = f(x)$$



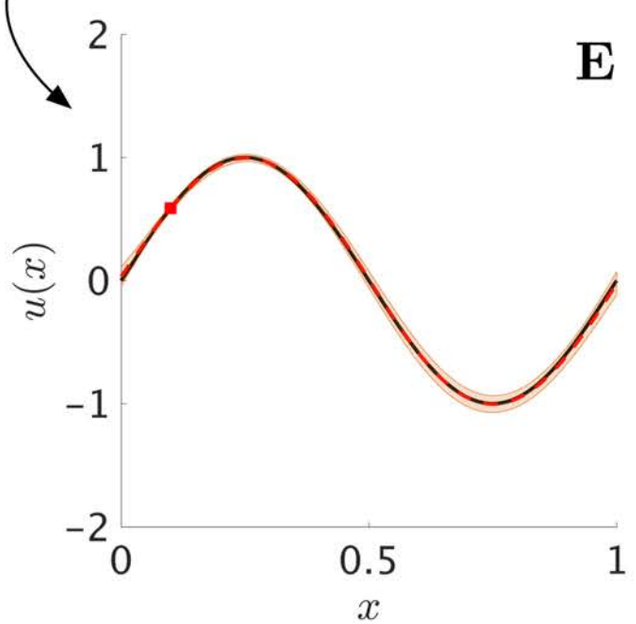
D Multi-fidelity regression



C

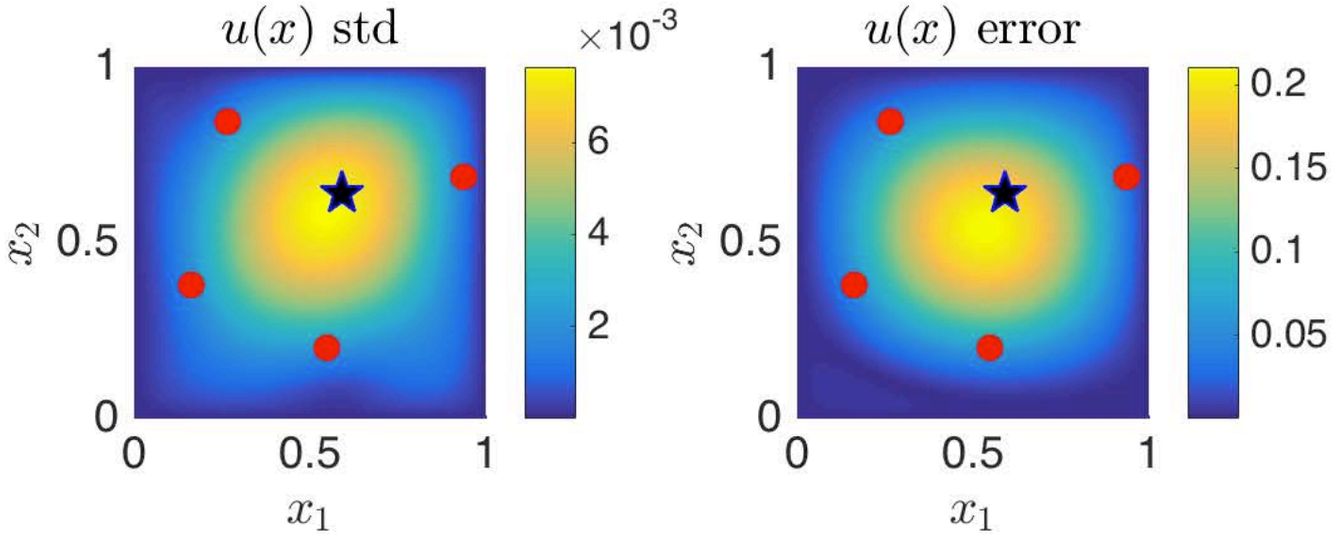


E

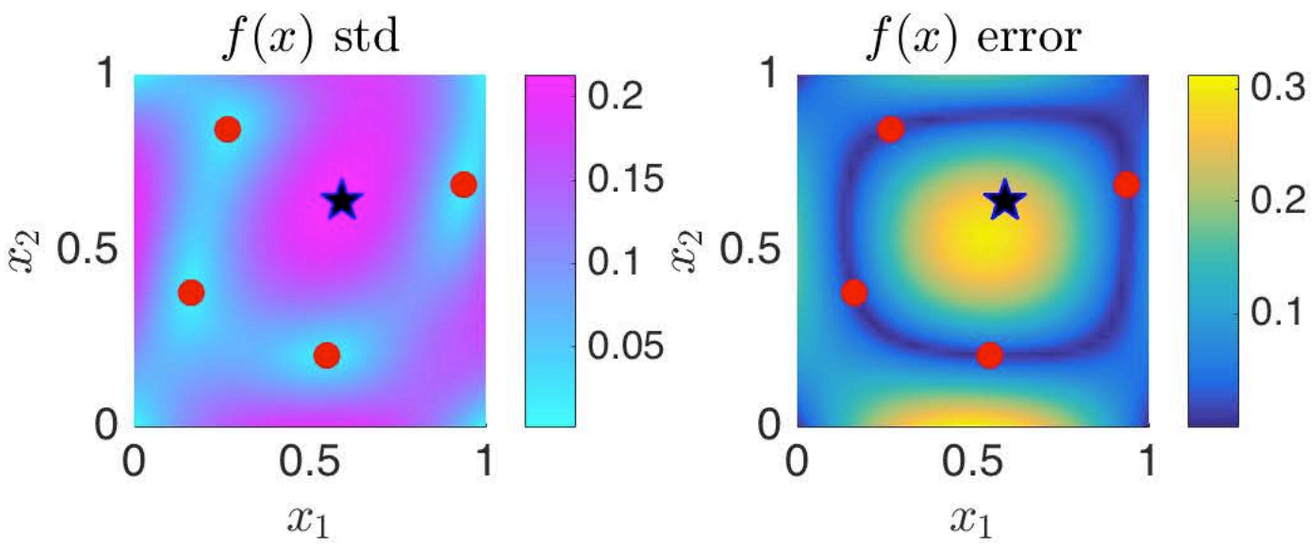


M. Raissi,., P. Perdikaris, and G.E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, http://128.84.21.199/abs/1607.04805, 2016

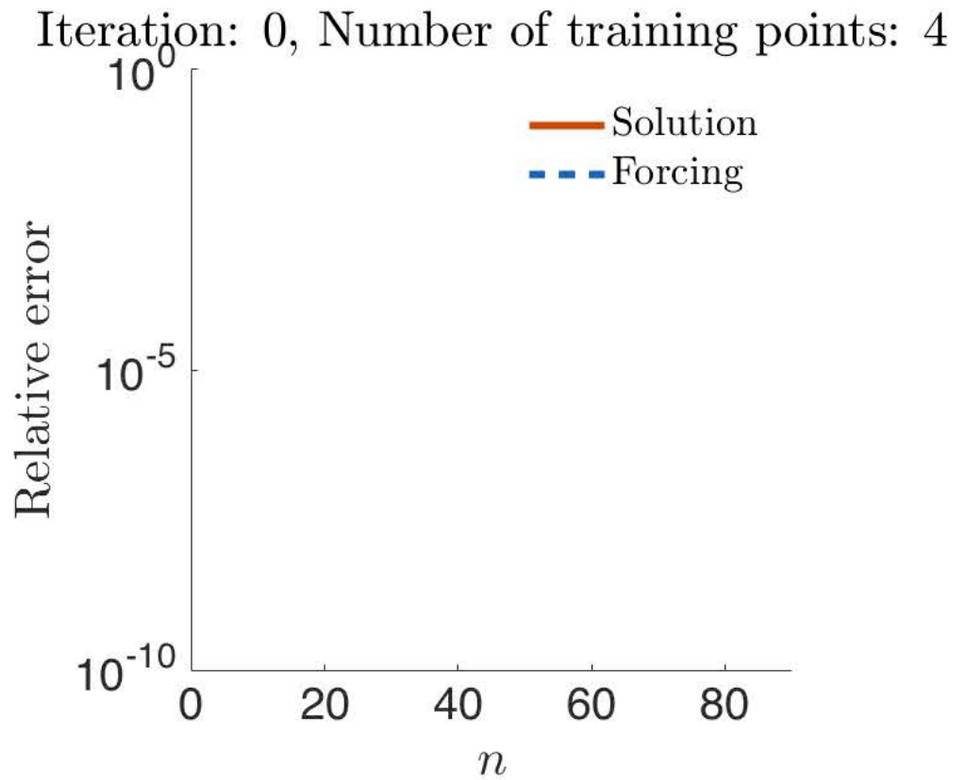
Adaptive refinement via active learning

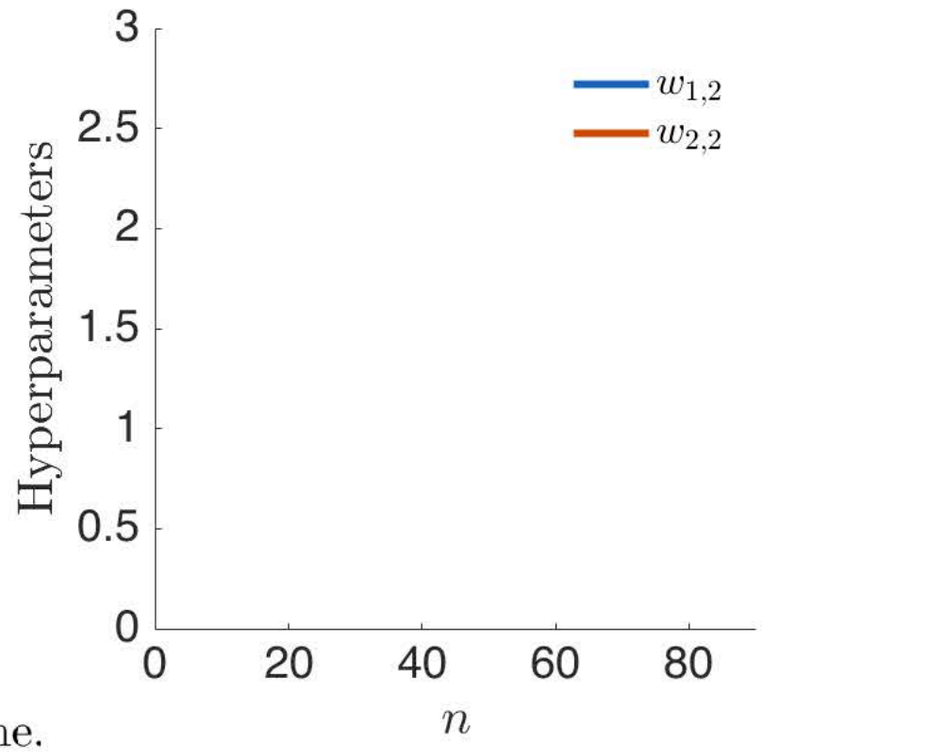


$$\frac{\partial^2 u}{\partial {x_1}^2} + \frac{\partial^2 u}{\partial {x_2}^2} = f(x_1, x_2)$$



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



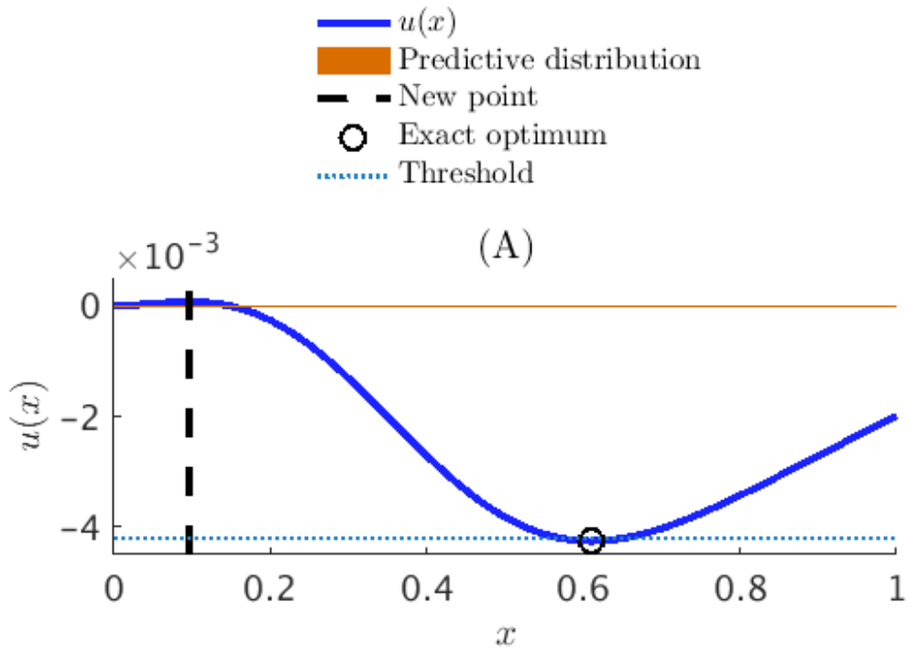


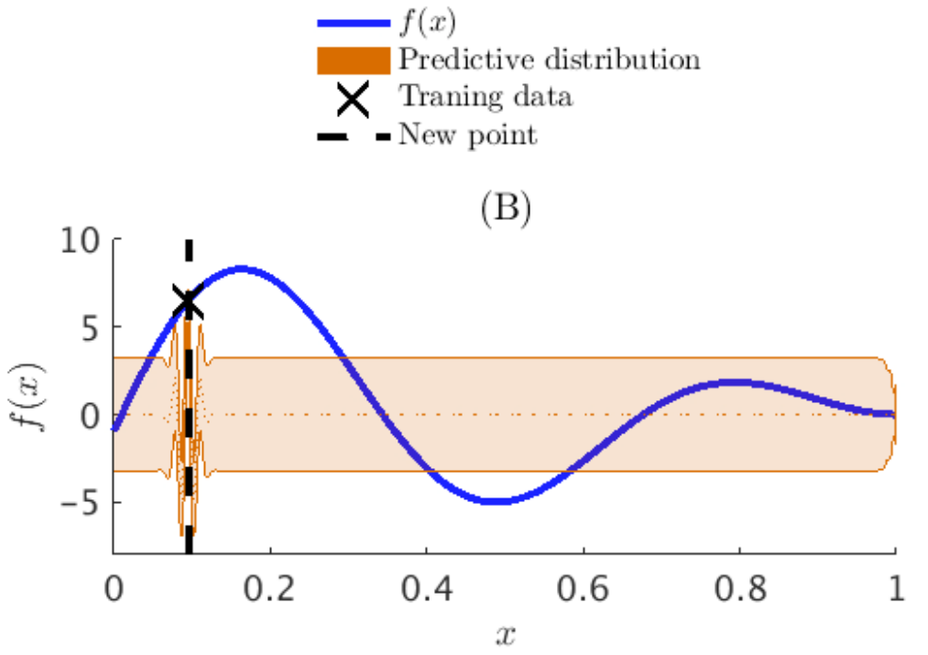
M. Raissi,., P. Perdikaris, and G.E. Karniadakis, Inferring solutions of differential equations using noisy multi-fidelity data, http://128.84.21.199/abs/1607.04805, 2016

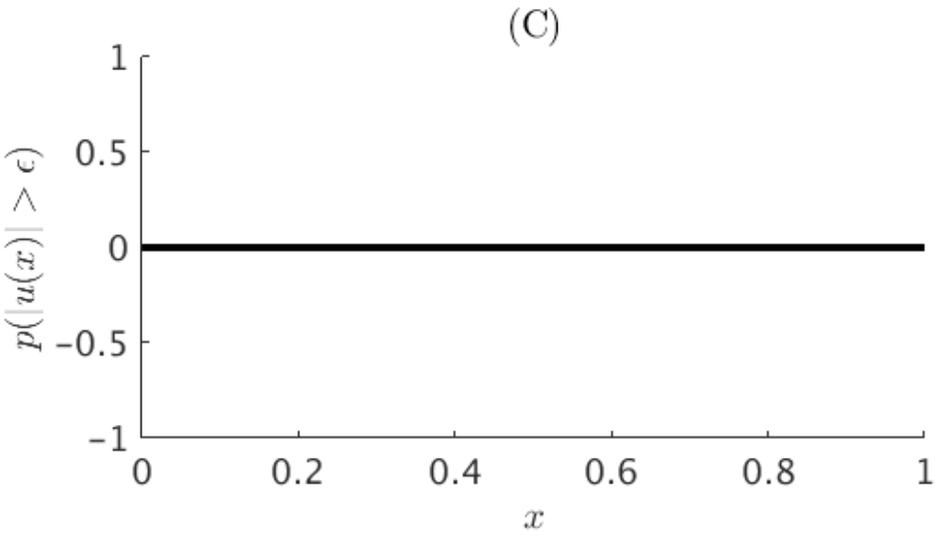
Estimating probabilities of failure in linear elasticity

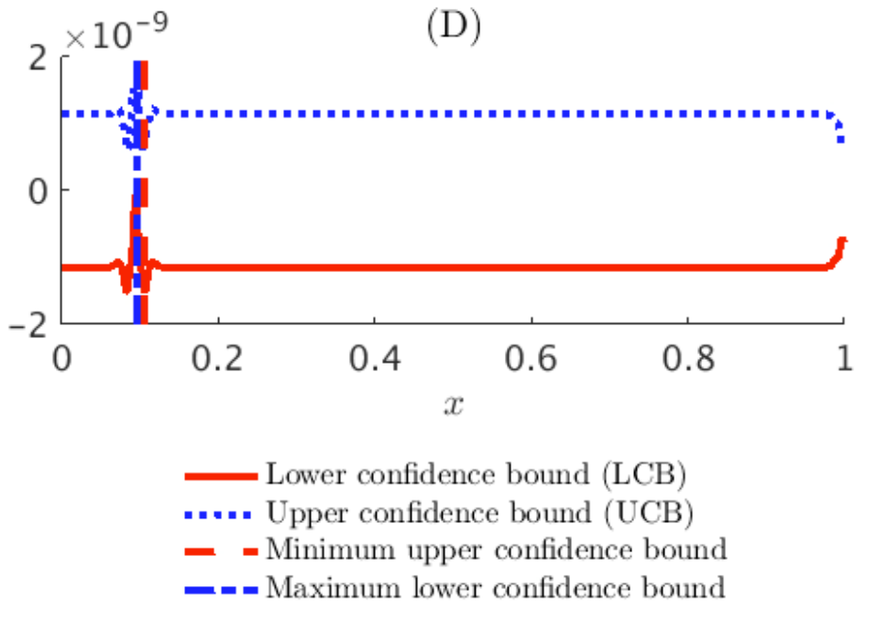
$$\mathcal{L}_x u(x) := \frac{d^4}{dx^4} u(x) = f(x)$$
$$u(0) = u'(0) = 0$$
$$u''(1) = 0$$
$$u'''(1) = f(1)$$

1. Given noisy observations of the loading $f(x)$, solve for the displacement $u(x)$.
2. Find the maximum displacement $|u(x)|$.
3. Given the threshold ϵ, find the probability of failure.









Identification of linear parametrized systems

$$u(x) \longrightarrow \boxed{\mathcal{L}_x^\phi : \phi = ?} \longrightarrow f(x)$$

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

$$u(x) \sim \mathcal{GP}(0, k_{uu}(x, x'; \theta)) \longrightarrow \mathcal{L}_x^\phi u(x) = f(x) \sim \mathcal{GP}(0, k_{ff}(x, x'; \theta, \phi))$$

$$k_{ff}(x, x'; \theta, \phi) = \mathcal{L}_x^\phi \mathcal{L}_{x'}^\phi k_{uu}(x, x'; \theta)$$

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

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

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

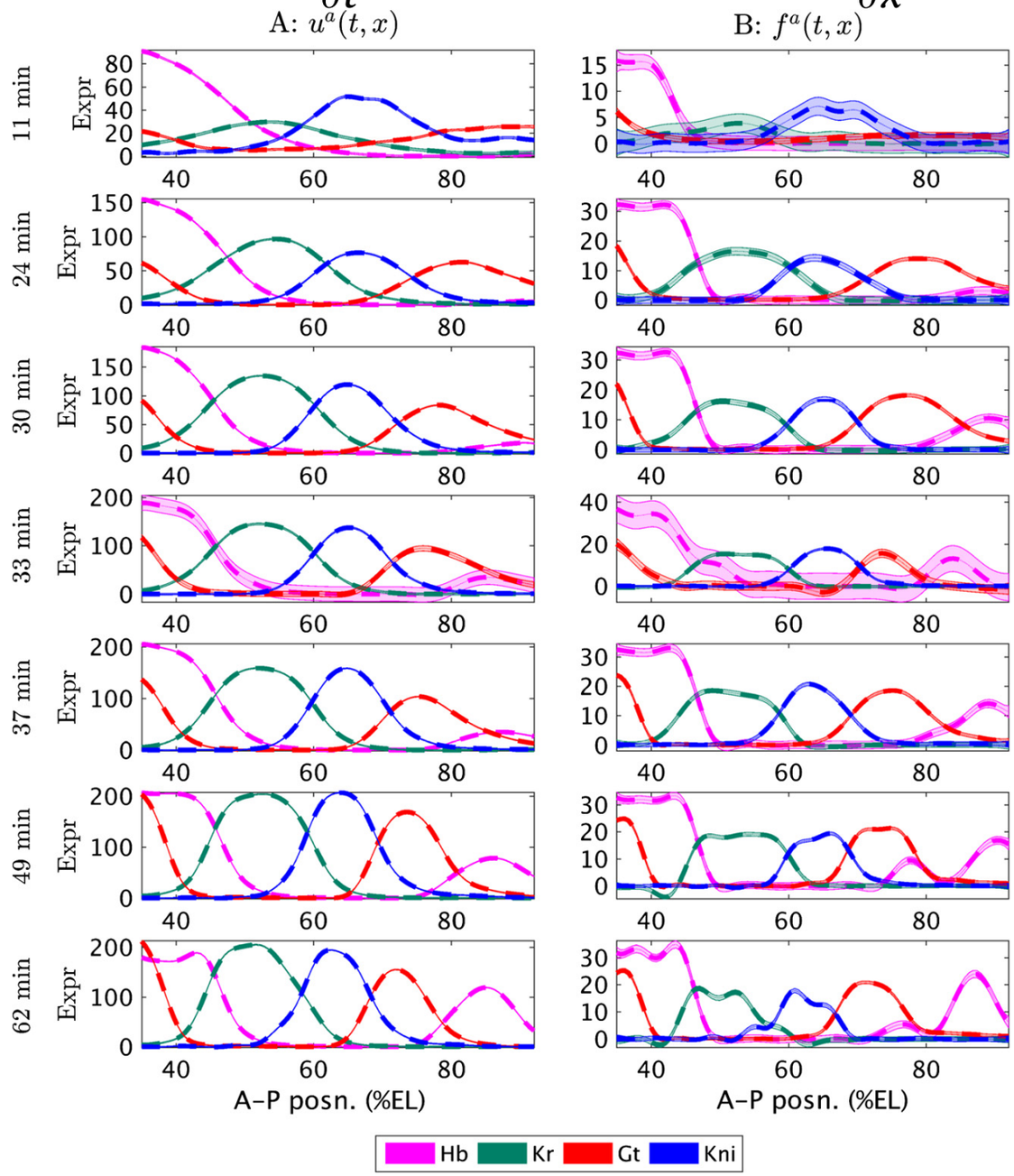
Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Machine learning of linear differential equations using Gaussian processes. Journal of Computational Physics, 348, 683-693.

Identification of linear parametrized systems

Example: Drosophila melanogaster gap gene dynamics

The gap gene dynamics of protein $a \in \{Hb, Kr, Gt, Kni\}$ (see Fig. 5) can be modeled using a reaction-diffusion partial differential equation

$$\mathcal{L}_{(t,x)}^{(\lambda^a, D^a)} u^a(t,x) = \frac{\partial}{\partial t} u^a(t,x) + \lambda^a u^a(t,x) - D^a \frac{\partial^2}{\partial x^2} u^a(t,x) = f^a(t,x),$$



Inferred parameter values for the decay λ^a and diffusion D^a rates of protein a.

Gene	Decay (λ^a)	Diff. (D^a)
Hb	0.1606	0.3669
Kr	0.0797	0.4490
Gt	0.1084	0.4543
Kni	0.0807	0.2683

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Machine learning of linear differential equations using Gaussian processes. Journal of Computational Physics, 348, 683-693.

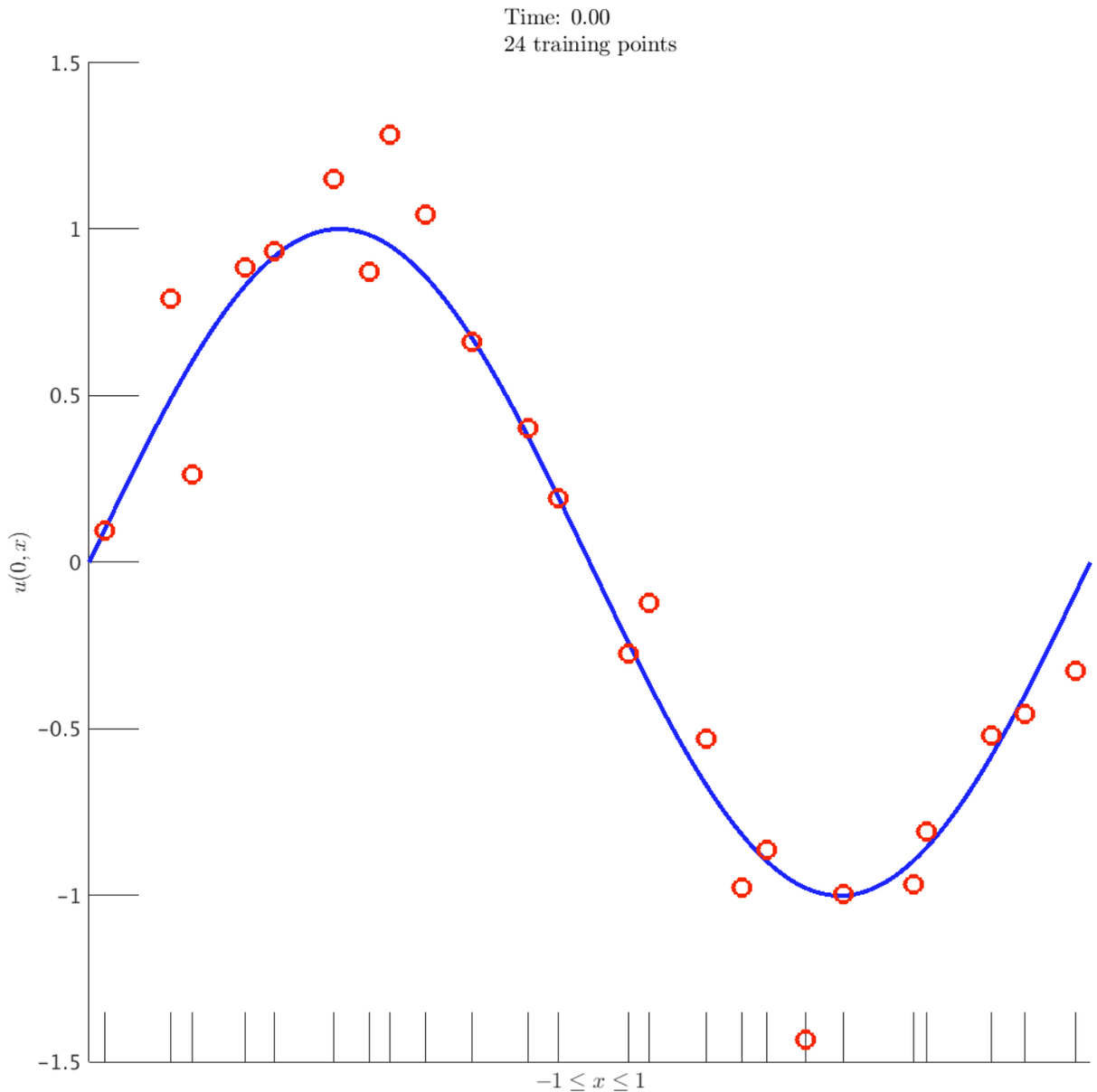
Extension to non-linear equations

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.

Identification of non-linear parametrized systems

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

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

$$\downarrow$$

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

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

$\longrightarrow$

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

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

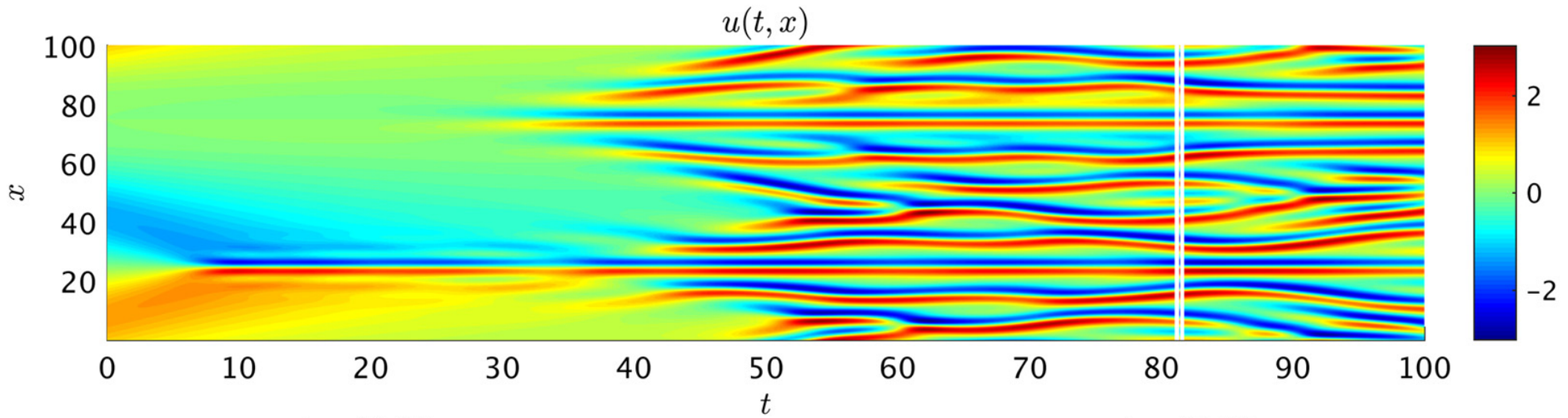
where $\boldsymbol{h} = \begin{bmatrix} \boldsymbol{h}^n \\ \boldsymbol{h}^{n-1} \end{bmatrix}$, $p(\boldsymbol{h}|\theta, \lambda, \sigma^2) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$, and $\boldsymbol{K}$ is given by

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

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

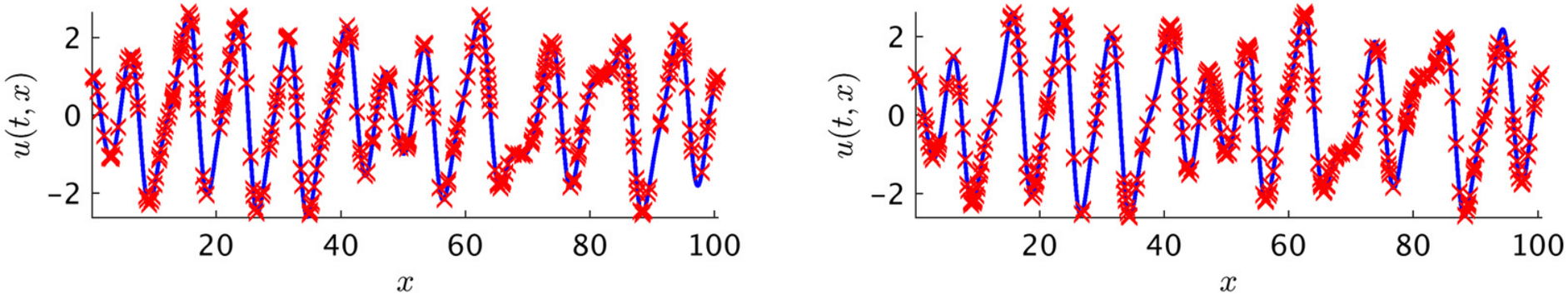
Identification of non-linear parametrized systems

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



$t = 81.20$
301 training data

$t = 81.60$
299 training data



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.