Hidden Physics Models

Maziar Raissi

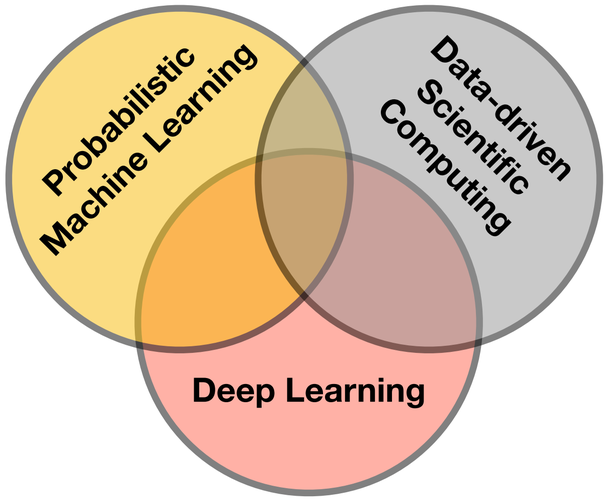
Assistant Professor

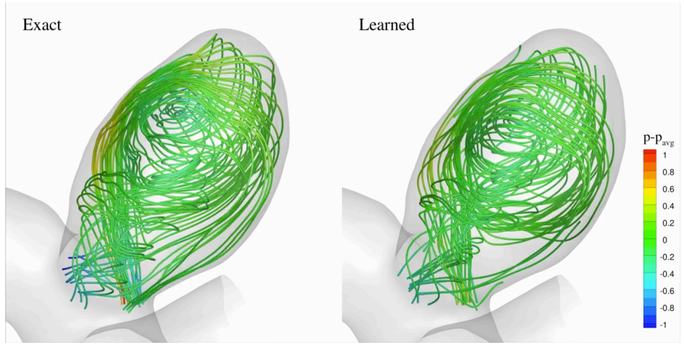
Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu

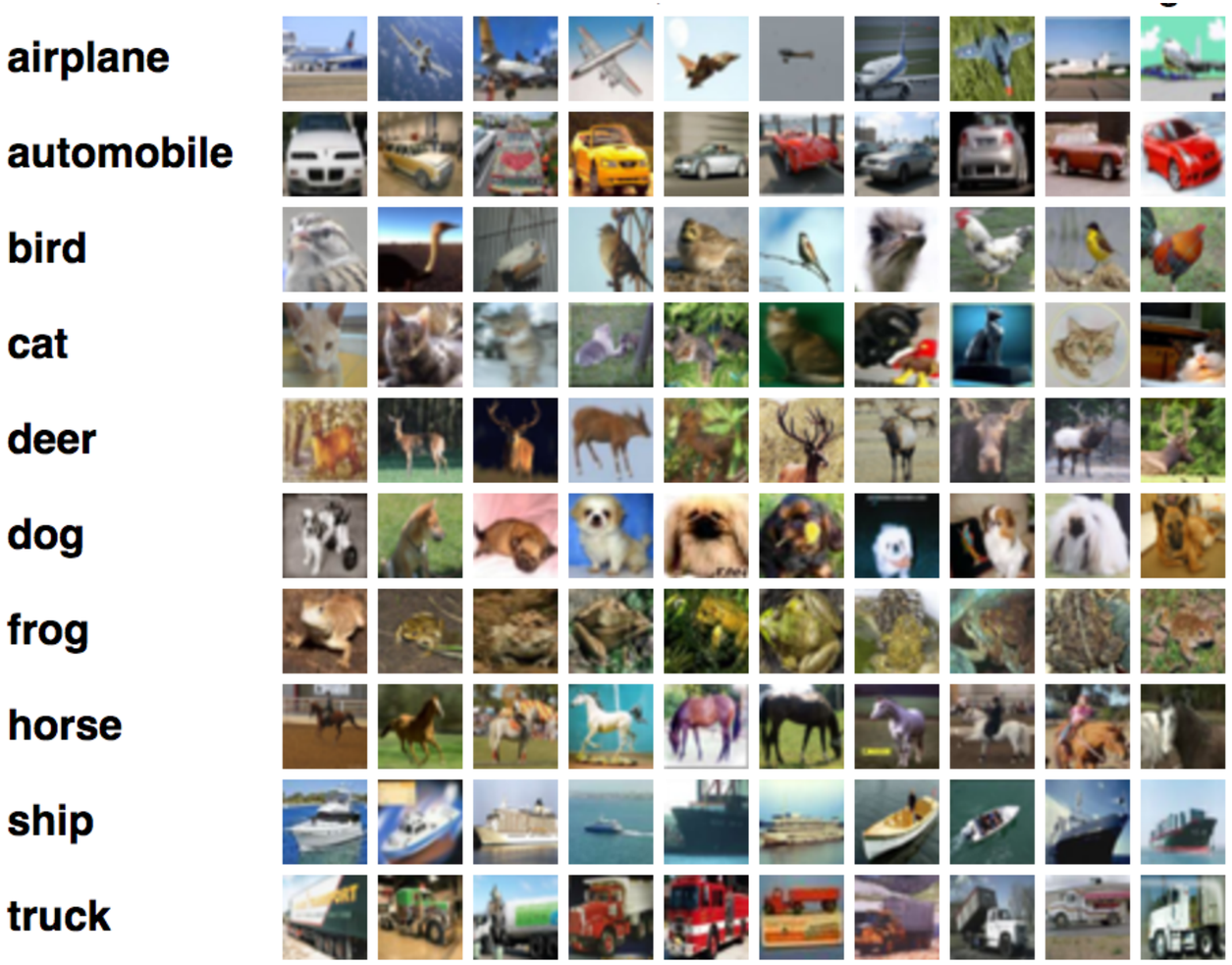
The ORACLE TEAM USA crew flying their AC72 “17” on foils.

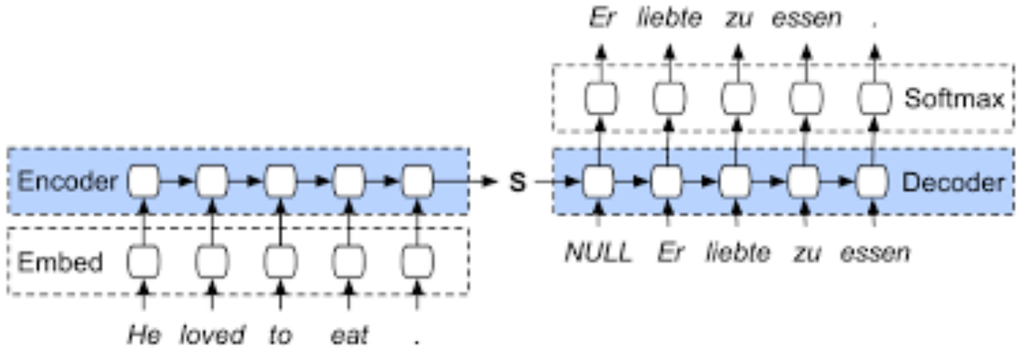


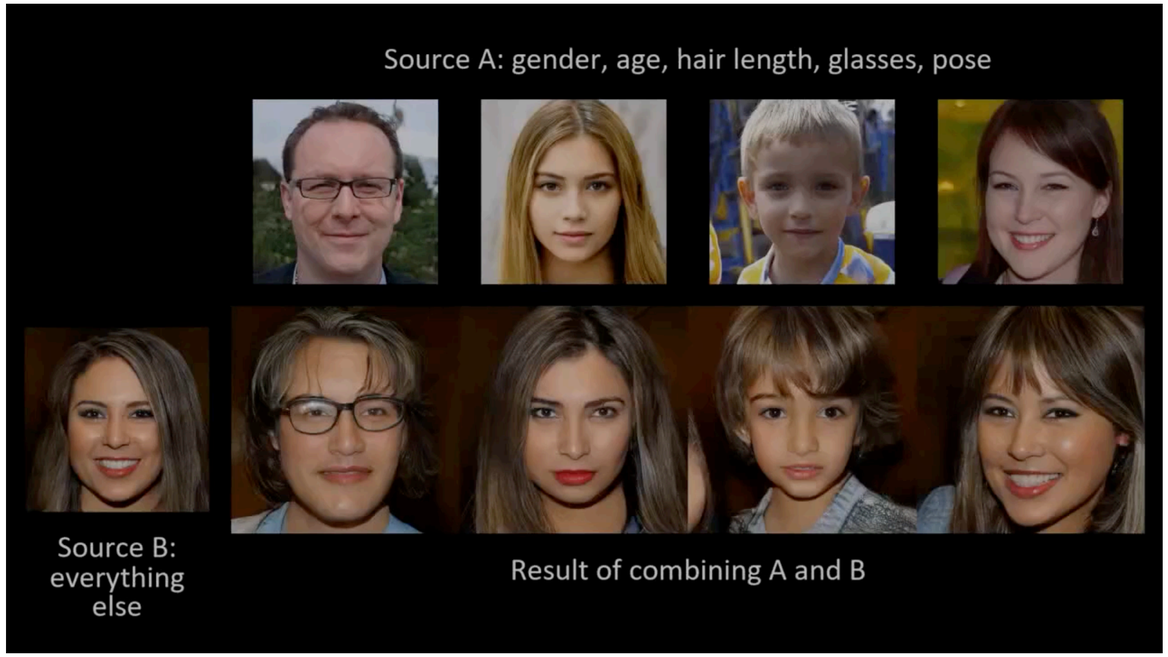


Boulder

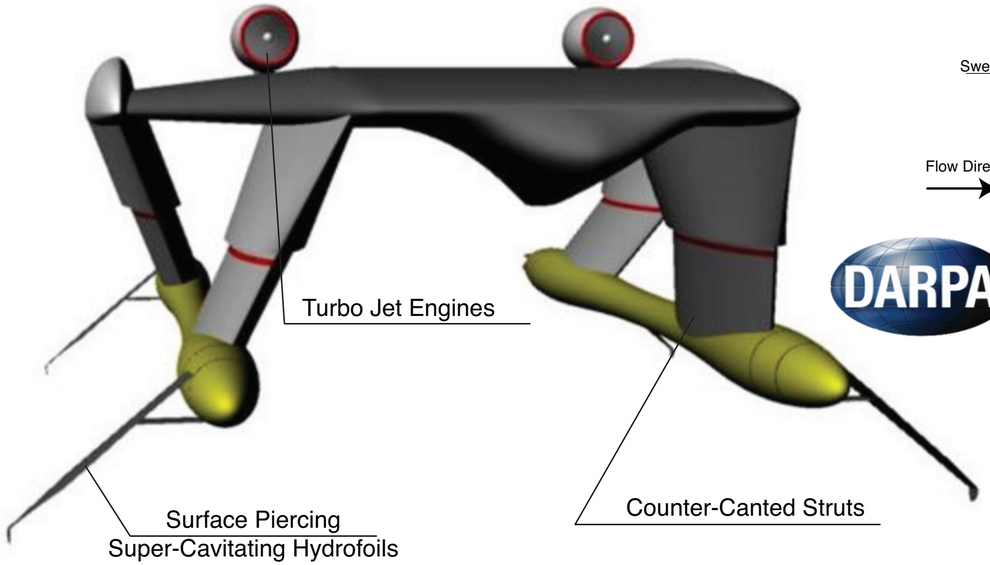
Machine/Deep Learning

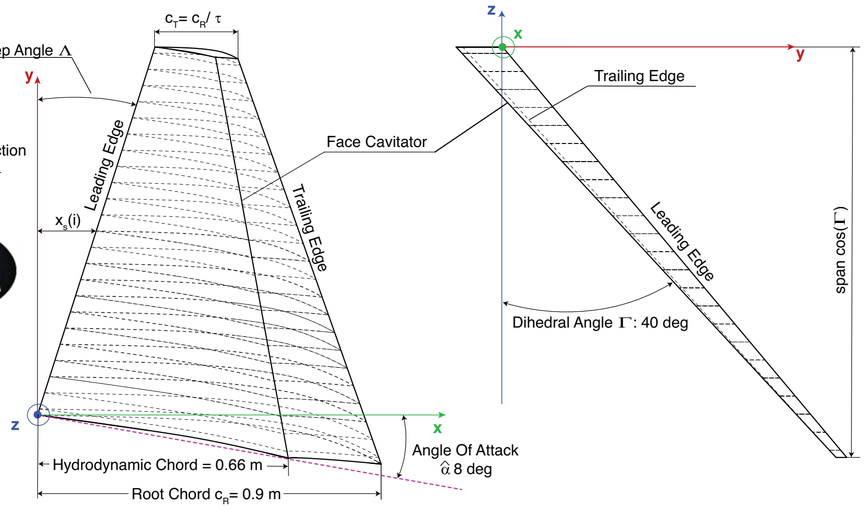


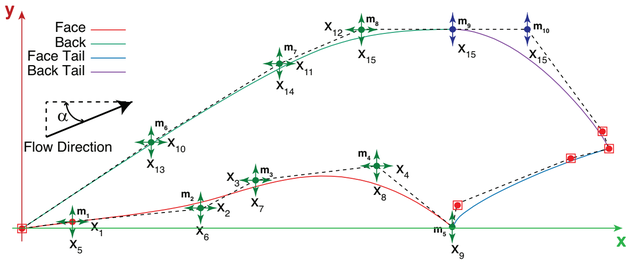


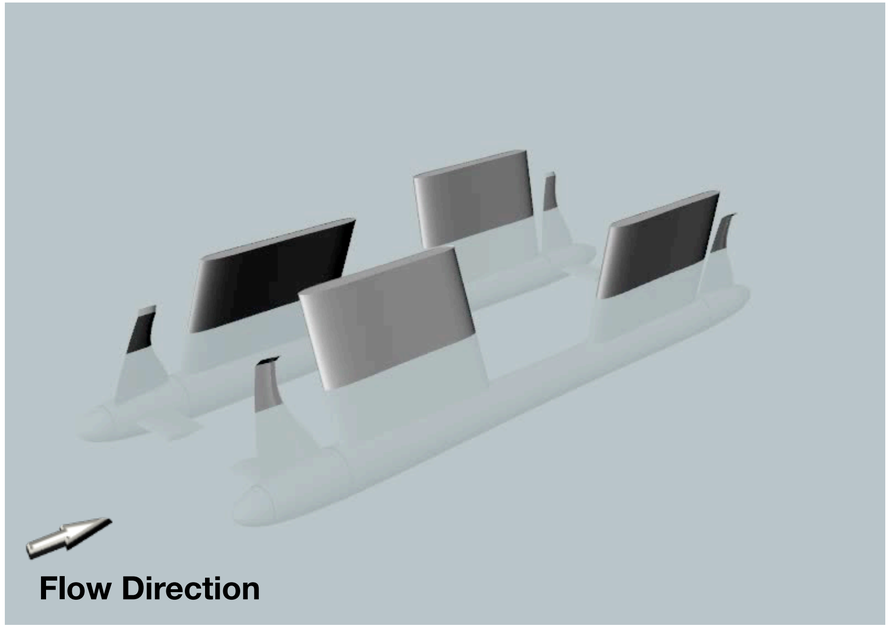


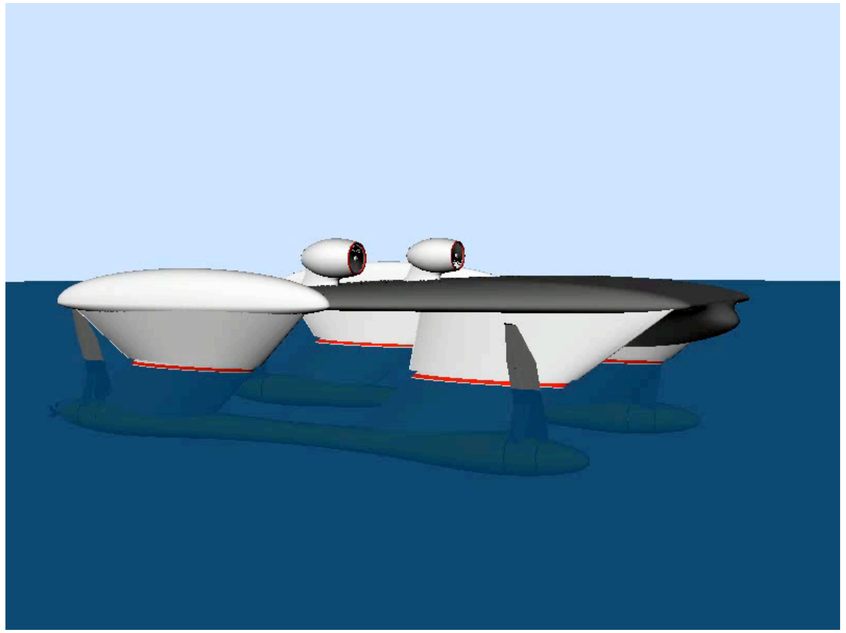
Design Optimization











lift/drag

Boulder

Gaussian Process Regression

Data

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right), \quad i = 1, \ldots, N$$

$$\mathbf{y} = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}\right)$$

Training

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}\right), \quad \mathbf{K} = k(\mathbf{x}, \mathbf{x}; \theta) + \sigma^2 \mathbf{I}$$

$$\min_{\theta, \sigma} \ \mathbf{y}' \mathbf{K}^{-1} \mathbf{y} + \log |\mathbf{K}|$$

Prior

$$f(x) \sim \mathcal{GP}\left(0, k(x, x'; \theta)\right)$$

$$\begin{bmatrix} f(x) \\ f(x') \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k(x, x; \theta) & k(x, x'; \theta) \\ k(x', x; \theta) & k(x', x'; \theta) \end{bmatrix} \right)$$

$$k(x, x'; \gamma, \omega) = \gamma^2 \exp\left(-0.5 \ \omega^2 (x - x')^2\right)$$

Prediction

$$\begin{bmatrix} f(x^*) \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} k(x^*, x^*) & k(x^*, \mathbf{x}) \\ k(\mathbf{x}, x^*) & \mathbf{K} \end{bmatrix} \right)$$

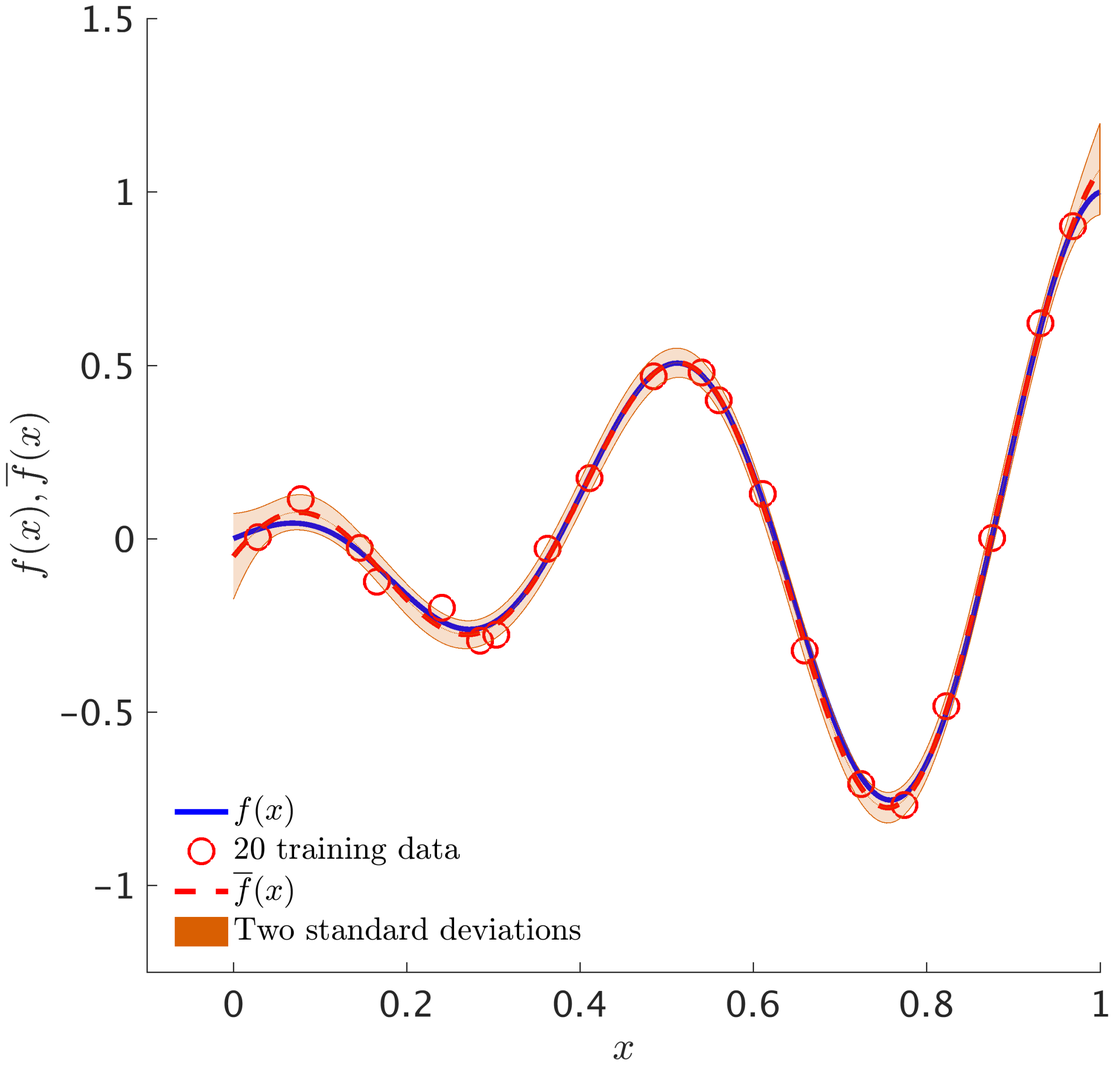
$$f(x^*) | \mathbf{x}, \mathbf{y} \sim \mathcal{N}\left(m(x^*), S(x^*, x^*)\right)$$

$$m(x^*) = k(x^*, \mathbf{x}) \mathbf{K}^{-1} \mathbf{y}$$

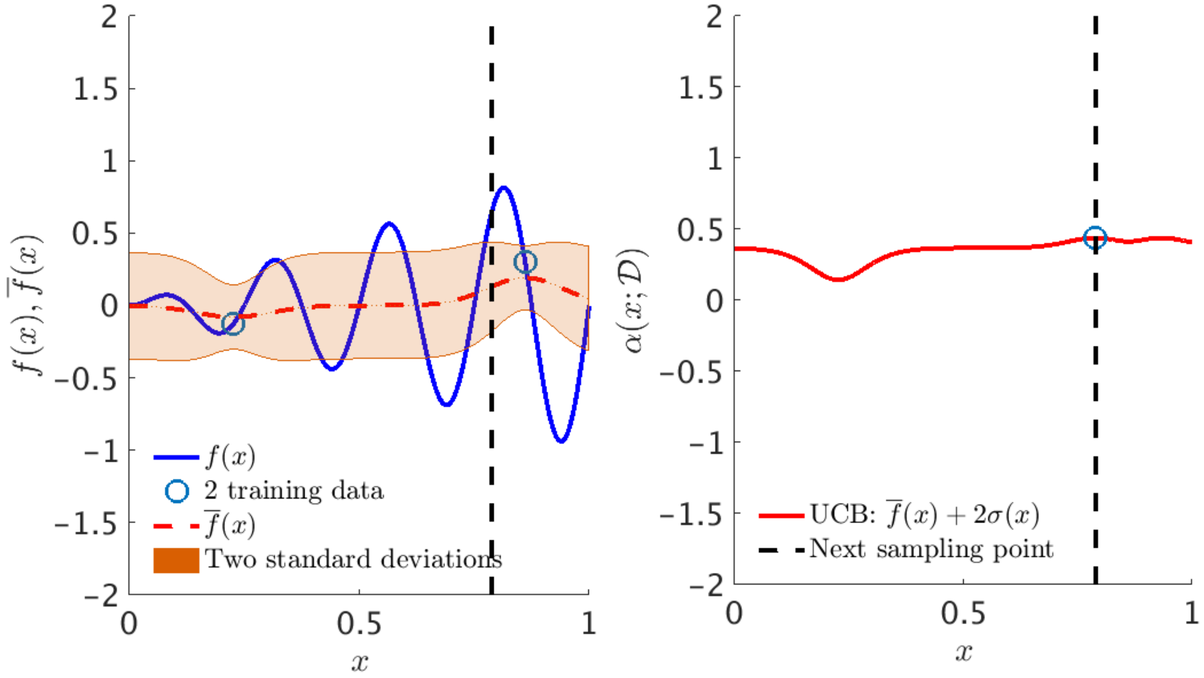
$$S(x^*, x^*) = k(x^*, x^*) - k(x^*, \mathbf{x}) \mathbf{K}^{-1} k(\mathbf{x}, x^*)$$

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



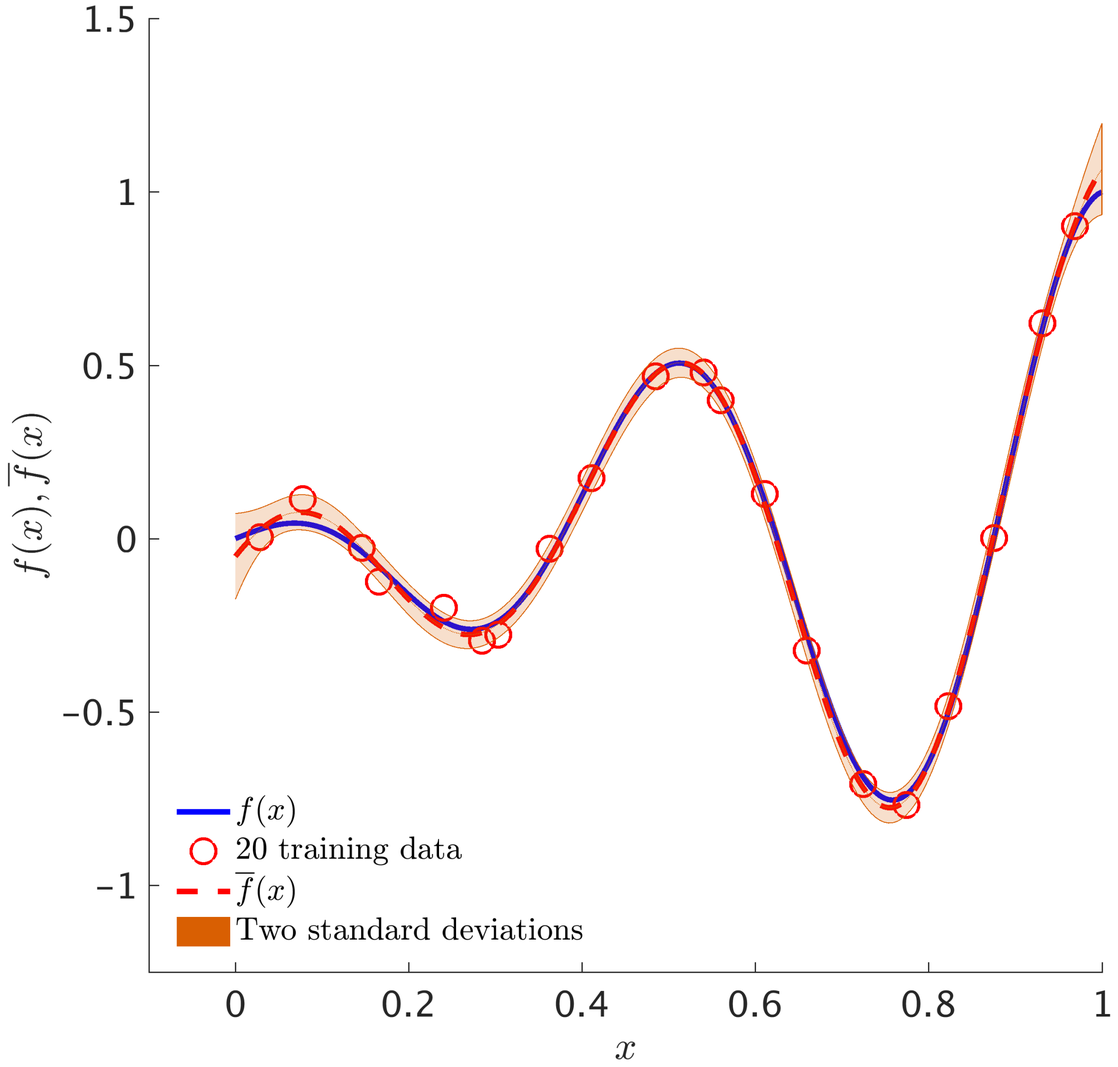
Bayesian Optimization



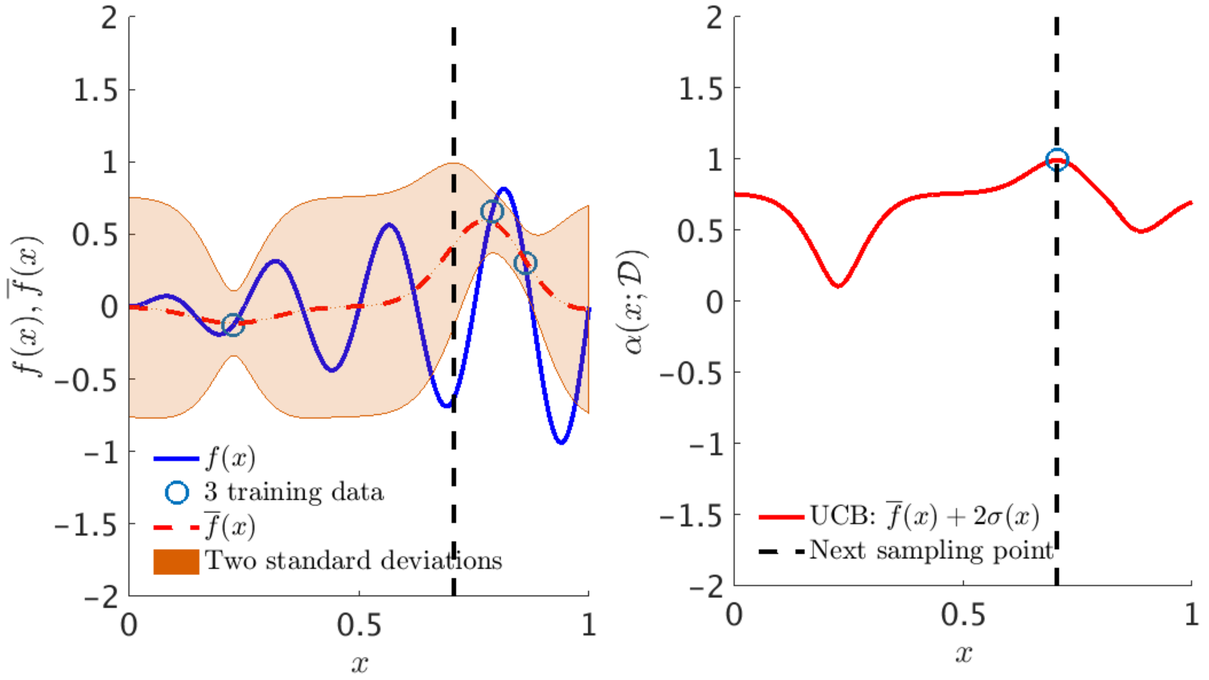
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



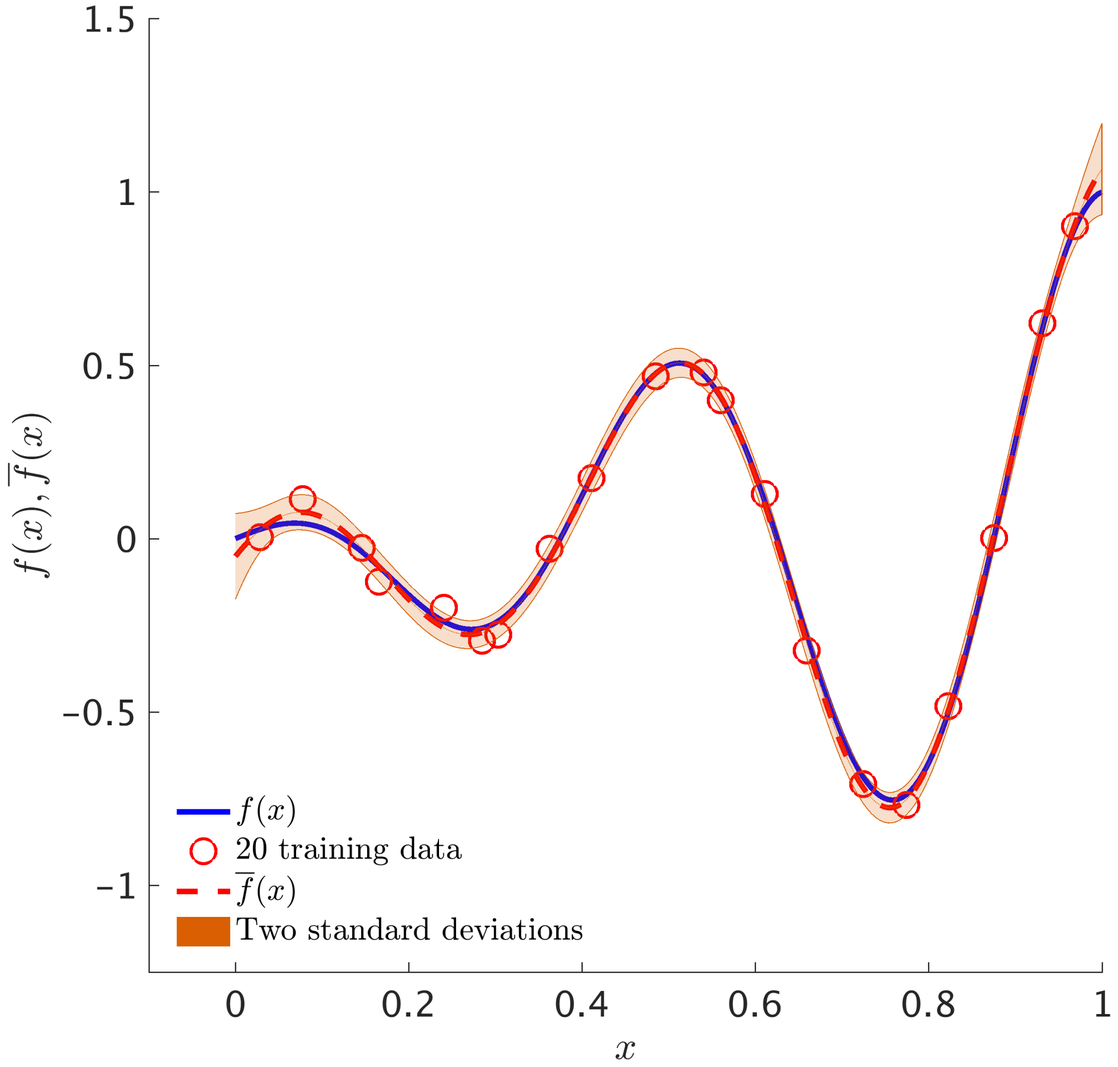
Bayesian Optimization



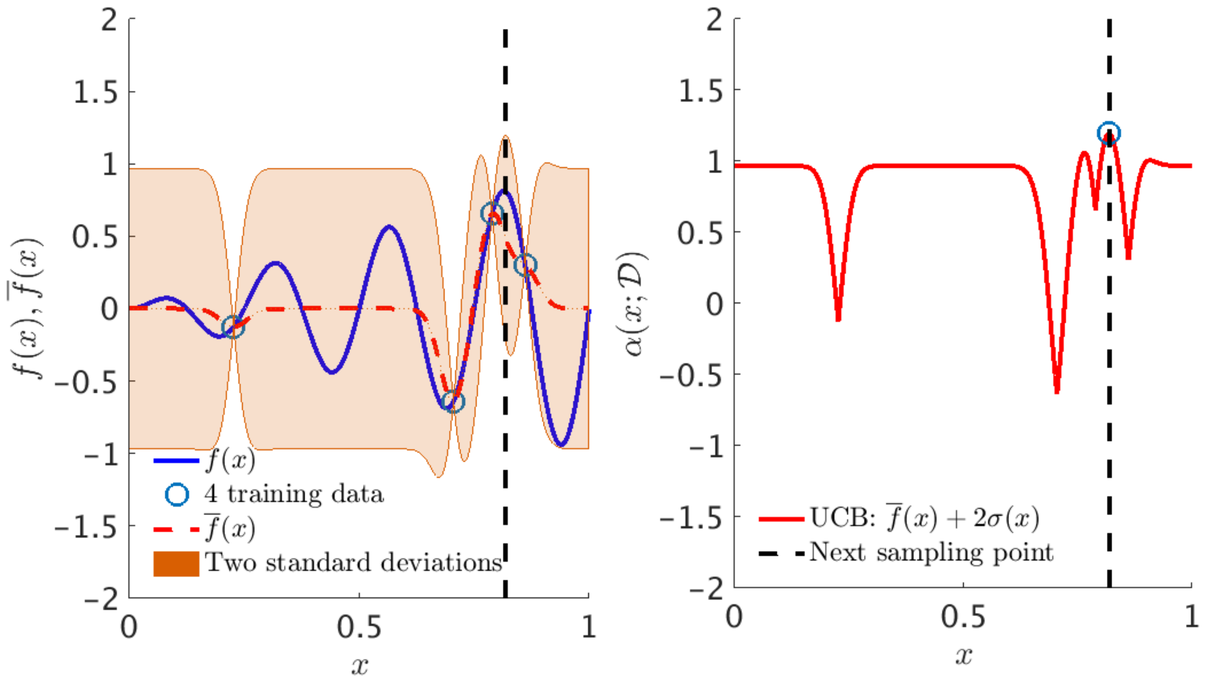
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



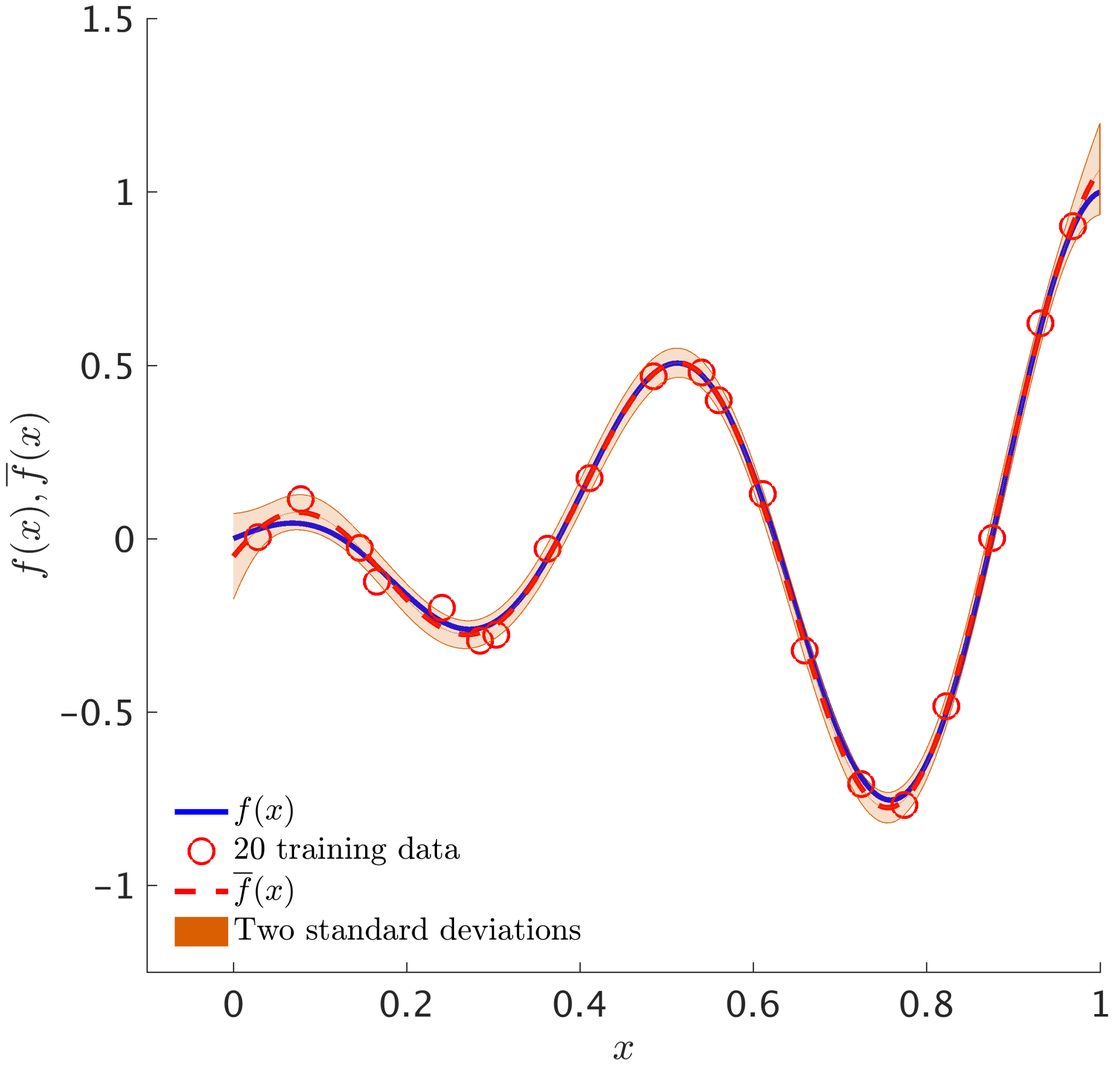
Bayesian Optimization



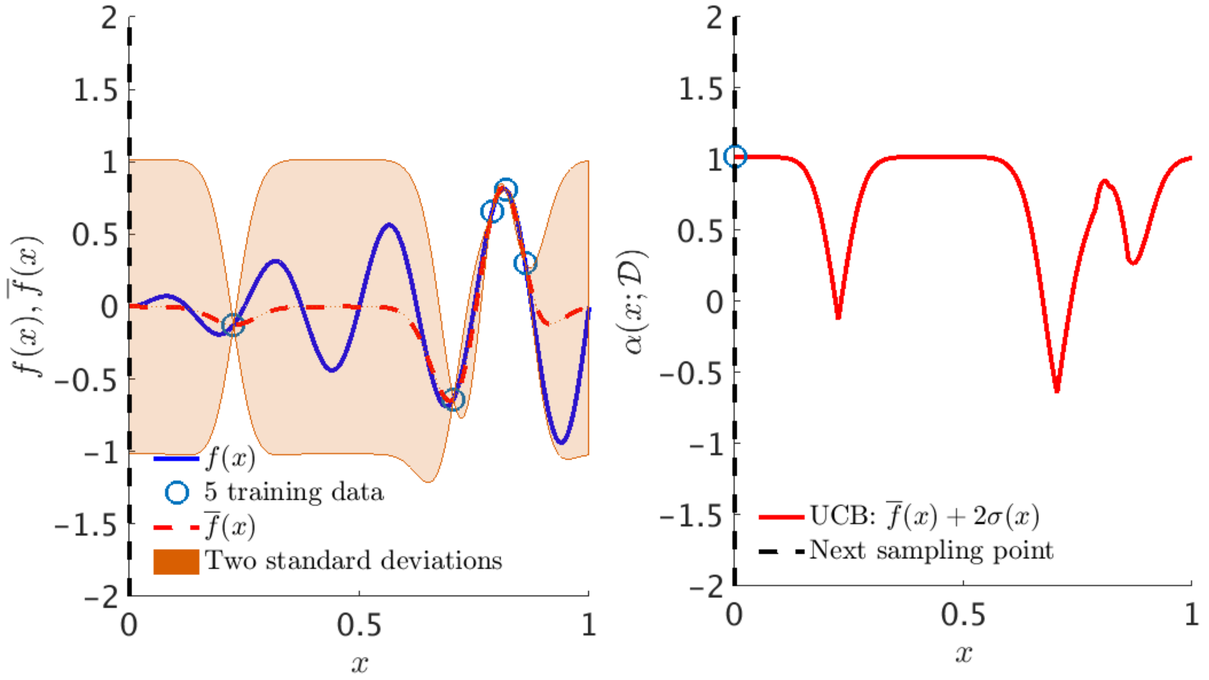
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



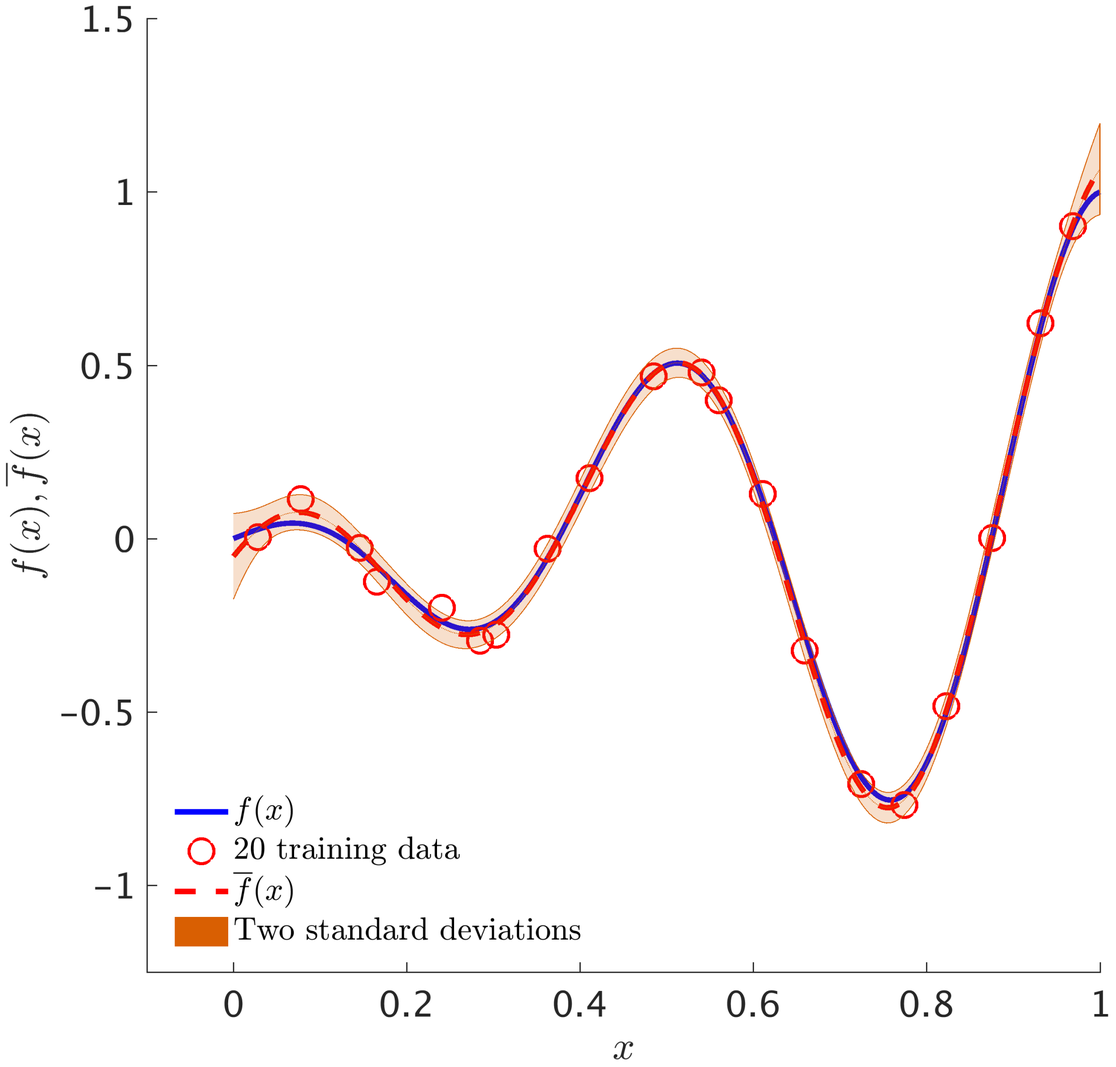
Bayesian Optimization



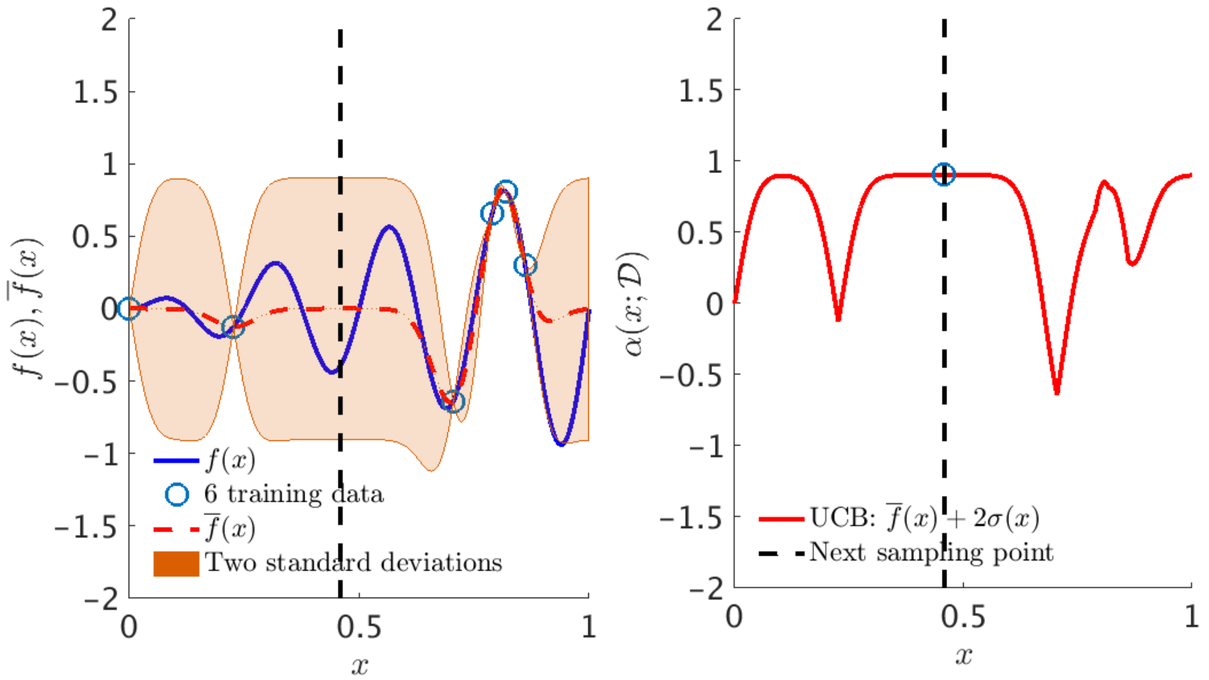
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



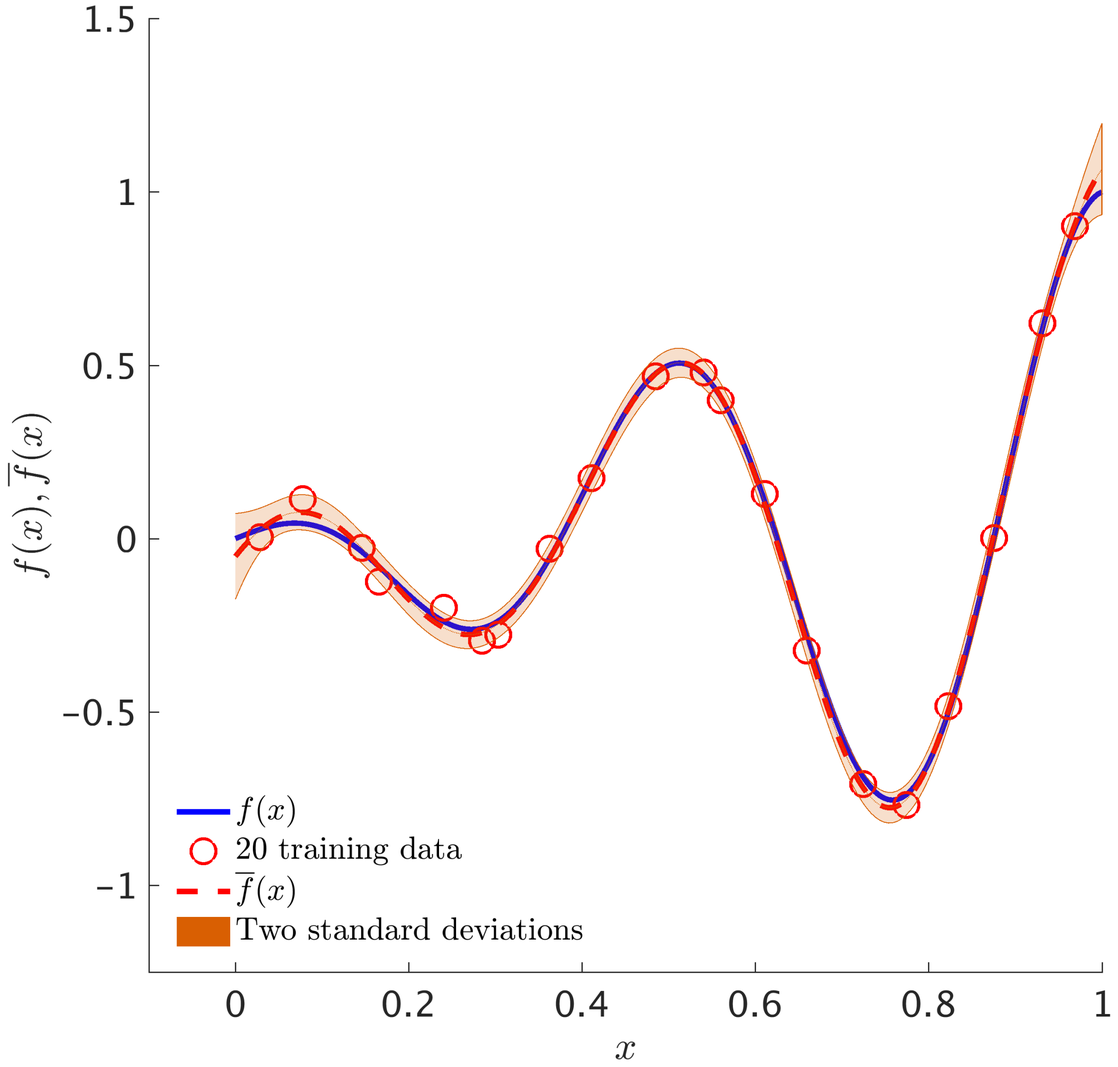
Bayesian Optimization



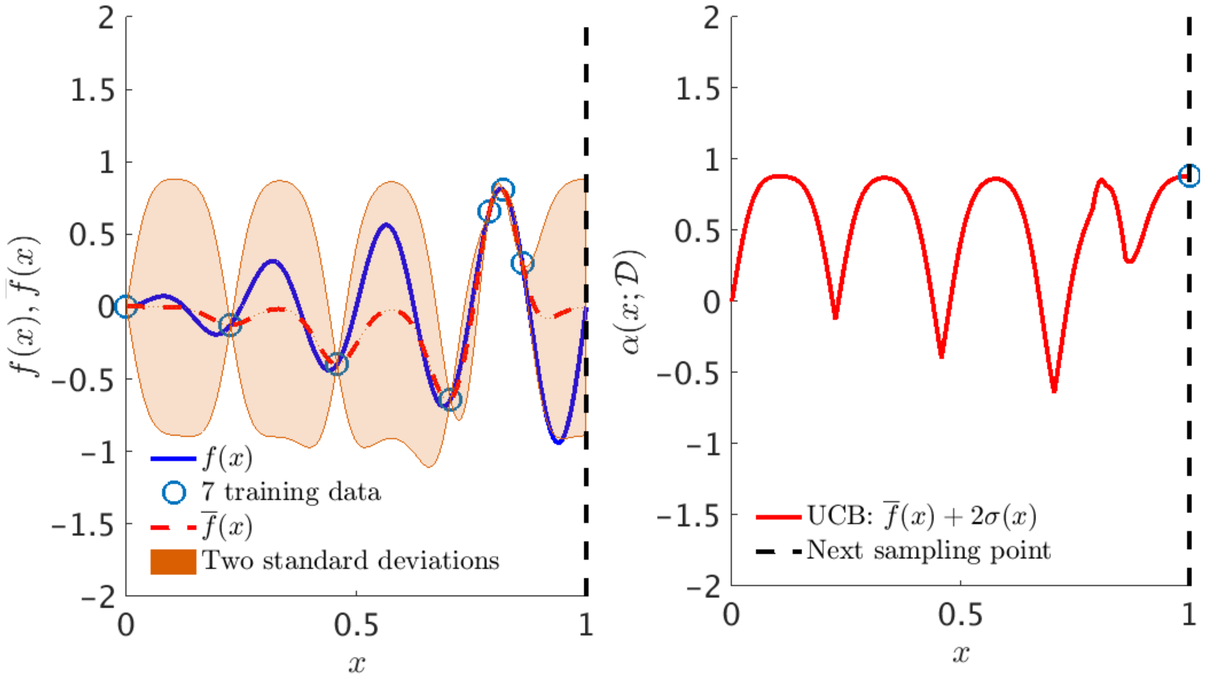
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



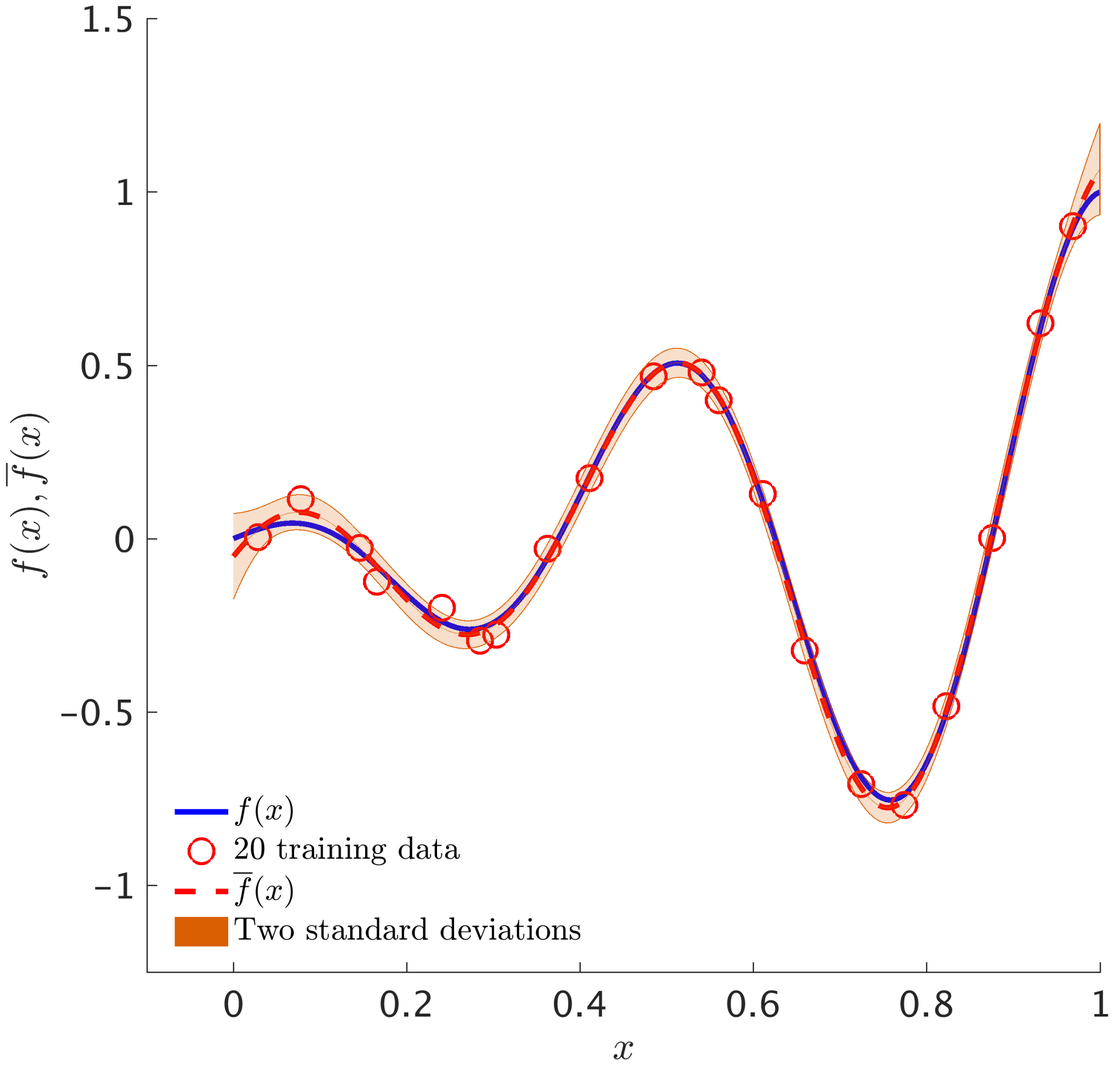
Bayesian Optimization



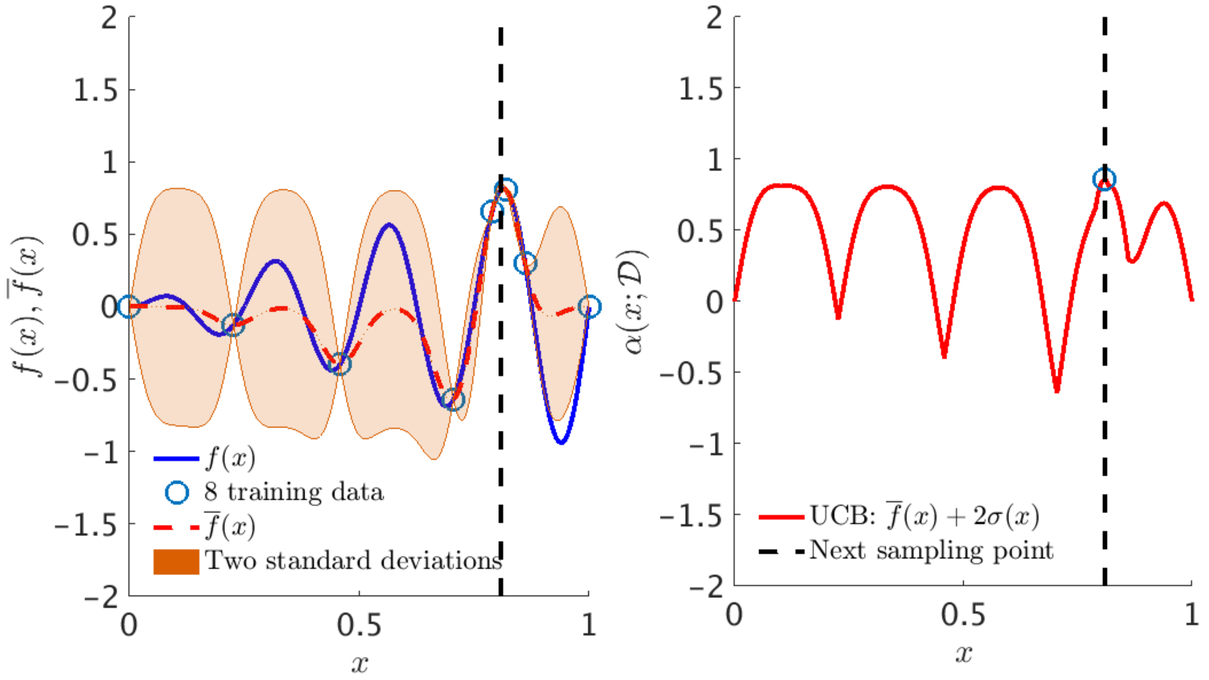
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



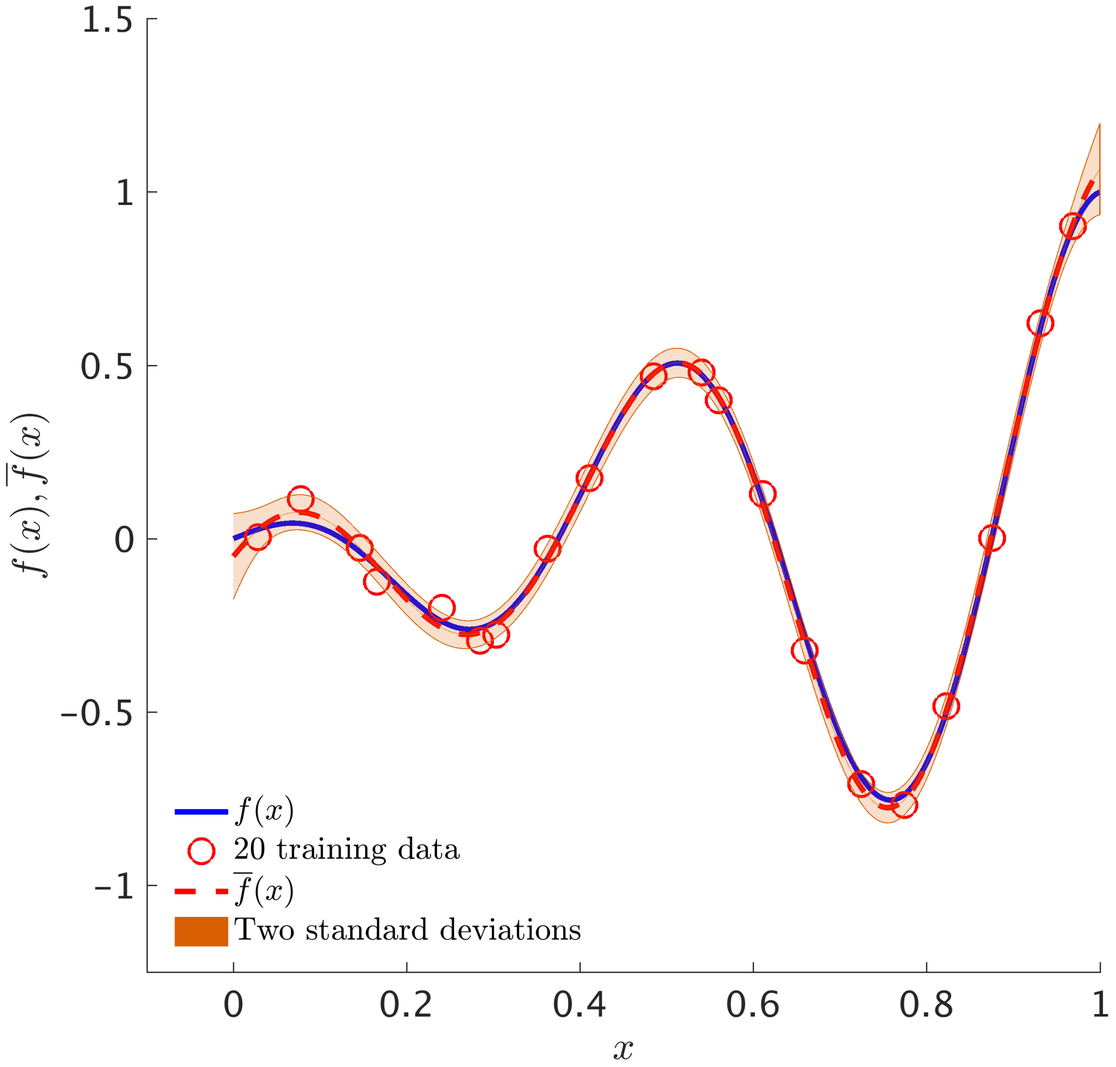
Bayesian Optimization



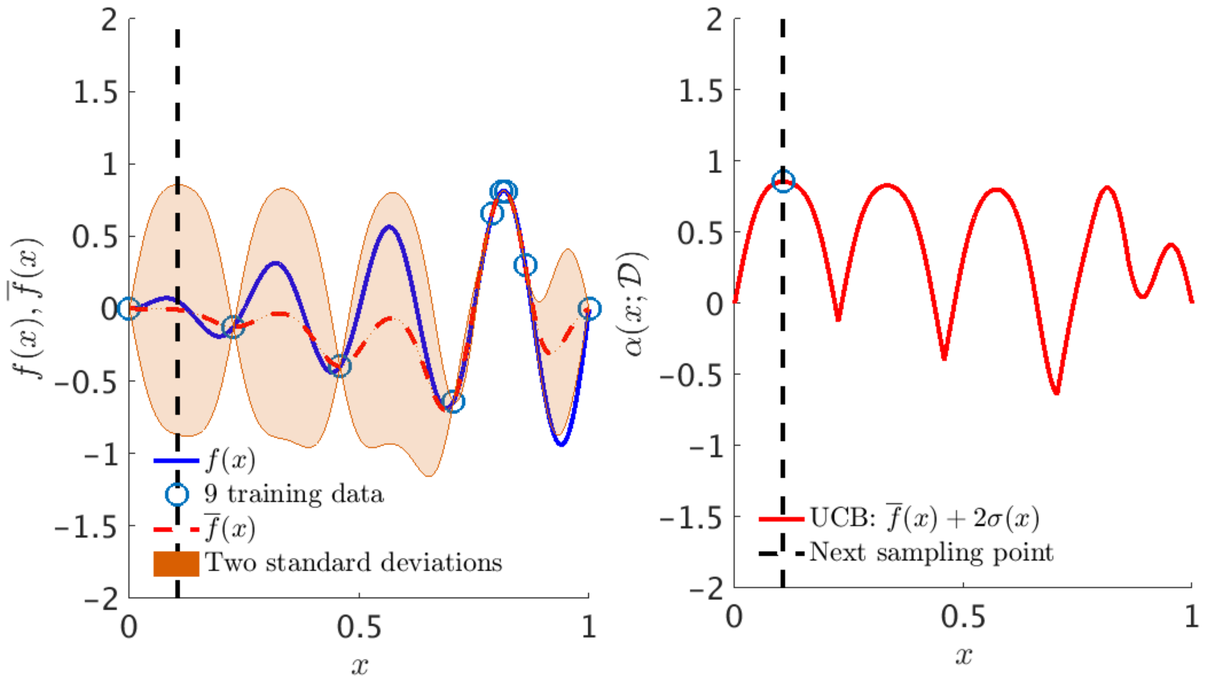
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



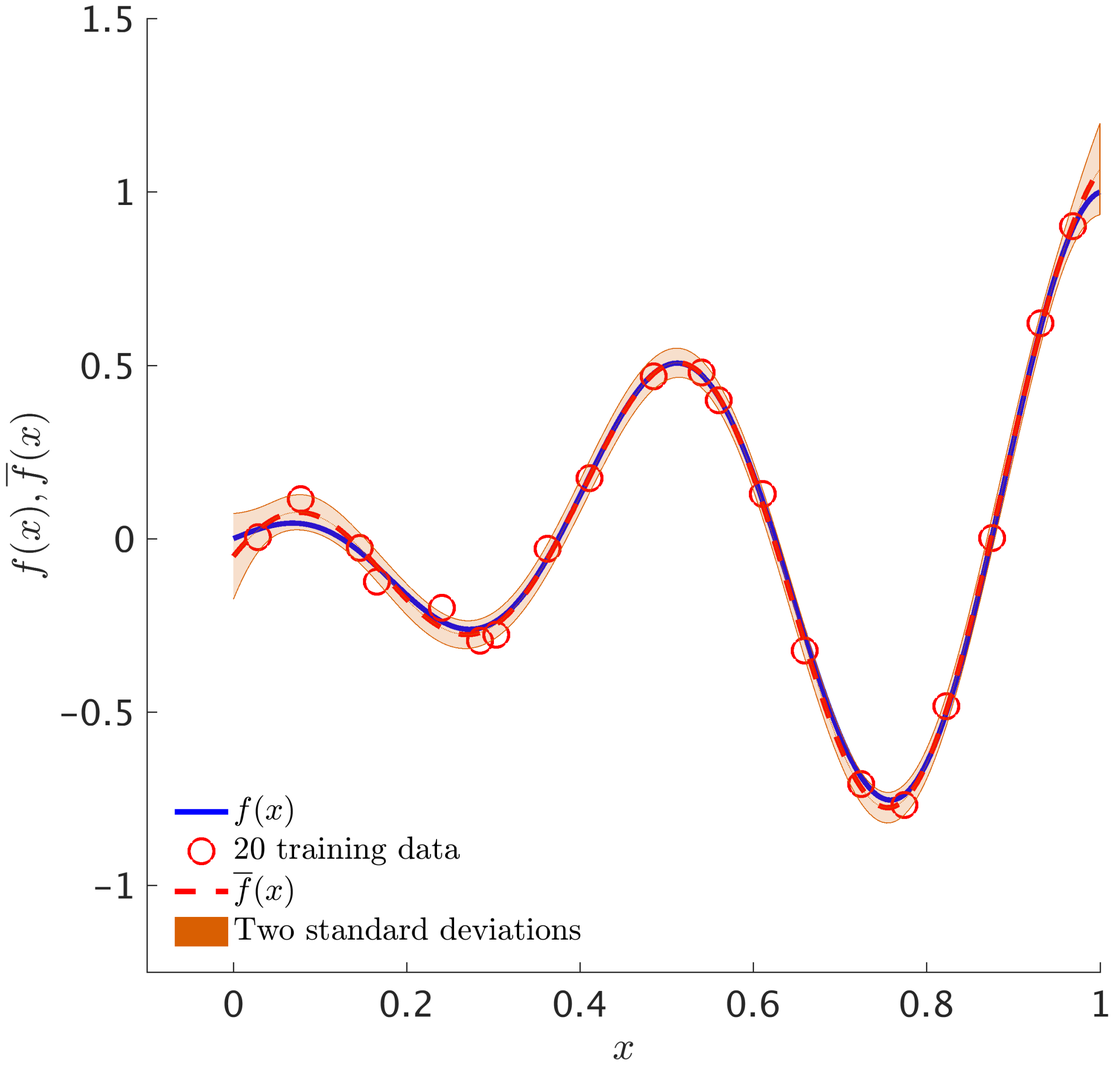
Bayesian Optimization



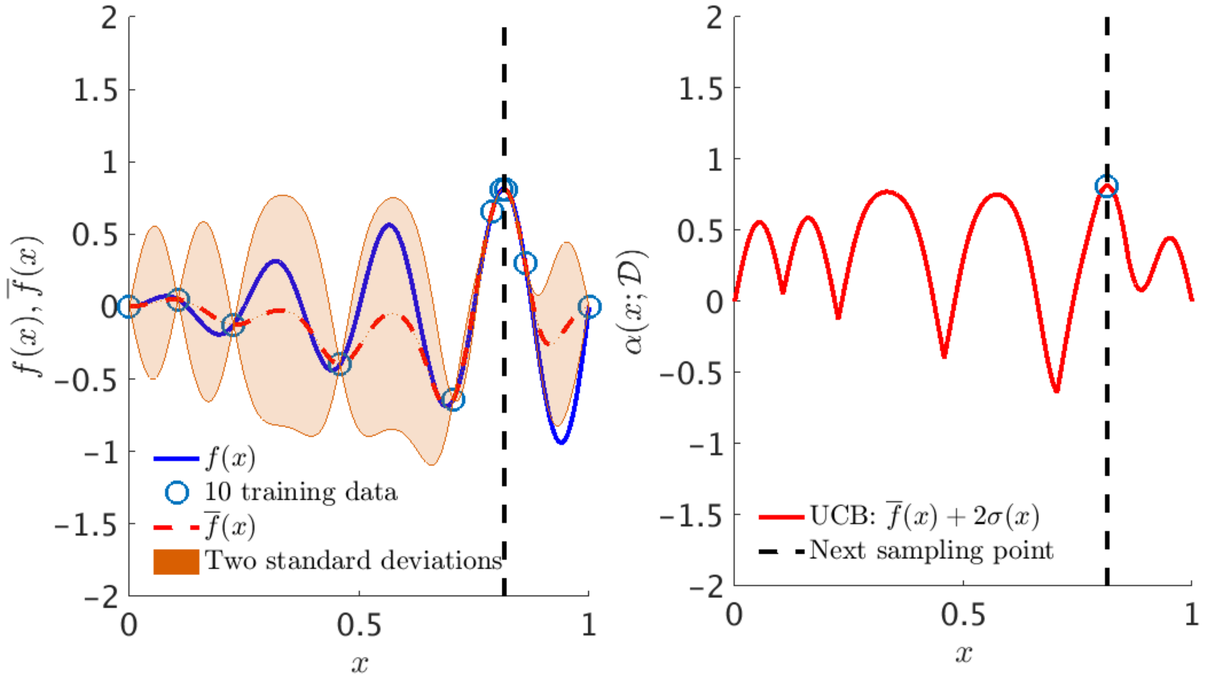
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



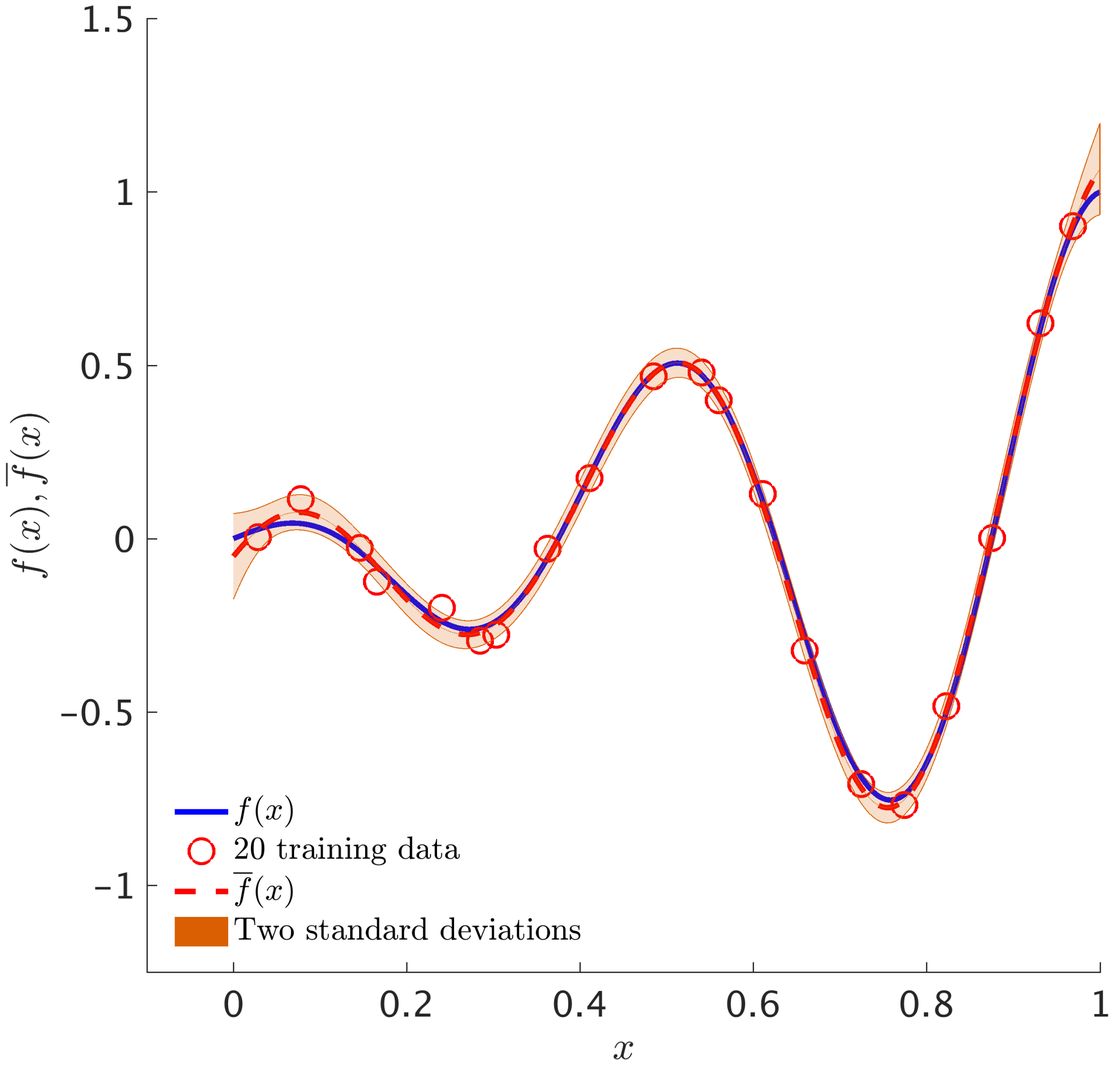
Bayesian Optimization



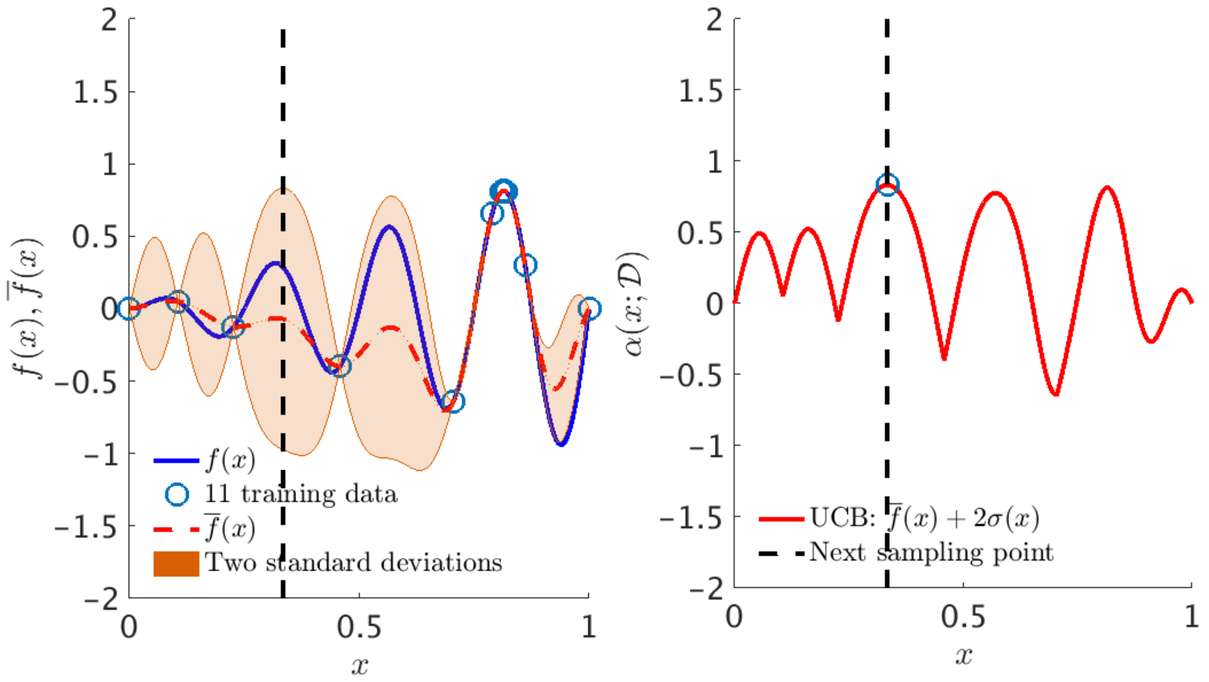
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



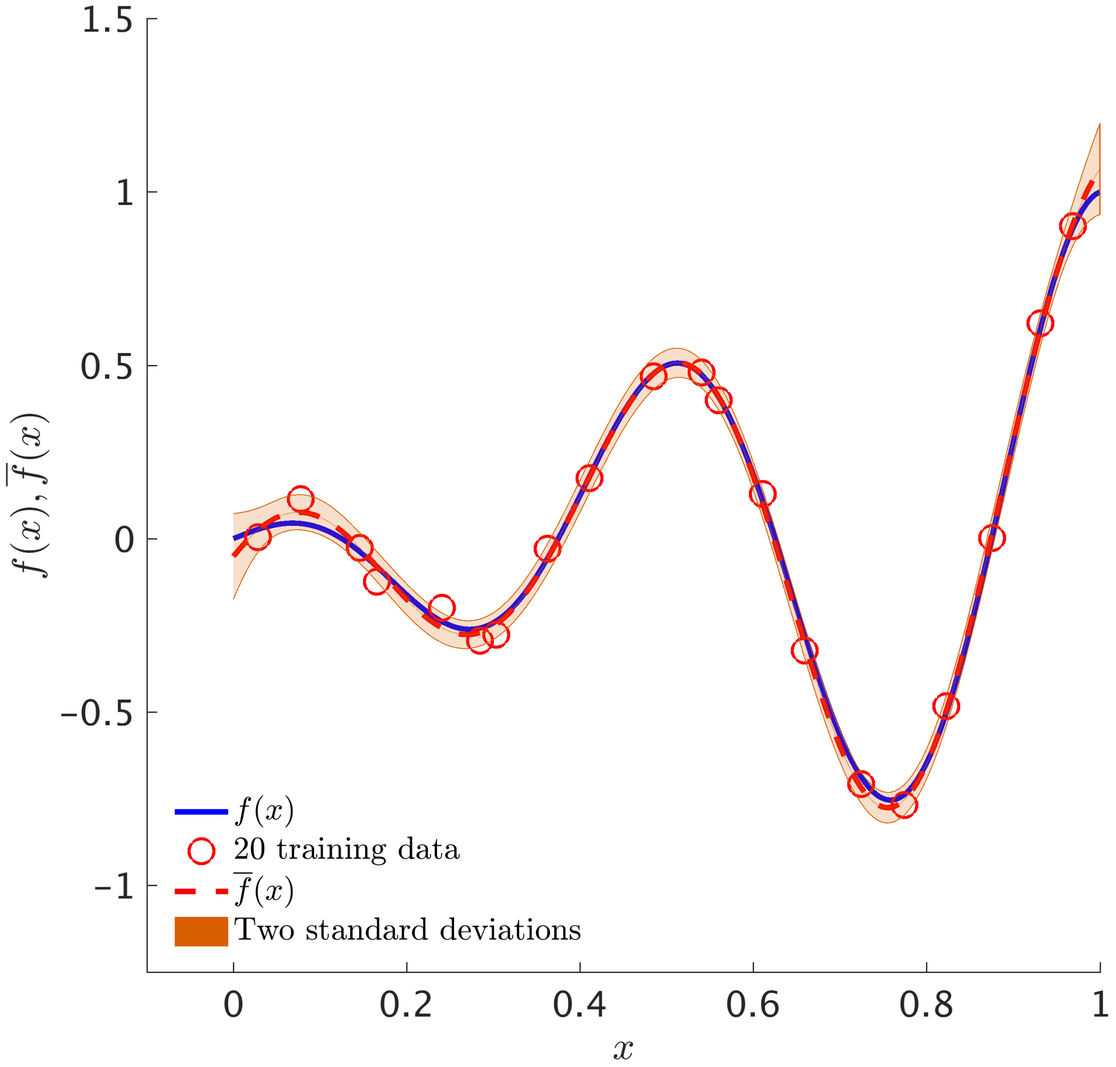
Bayesian Optimization



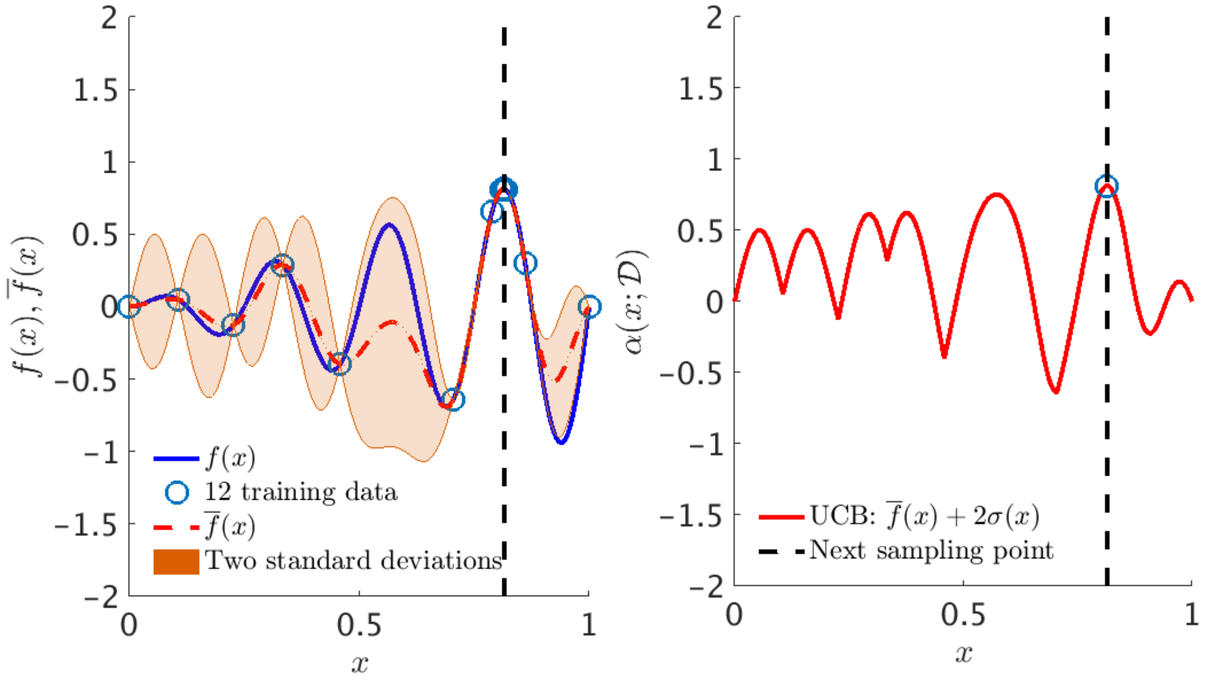
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



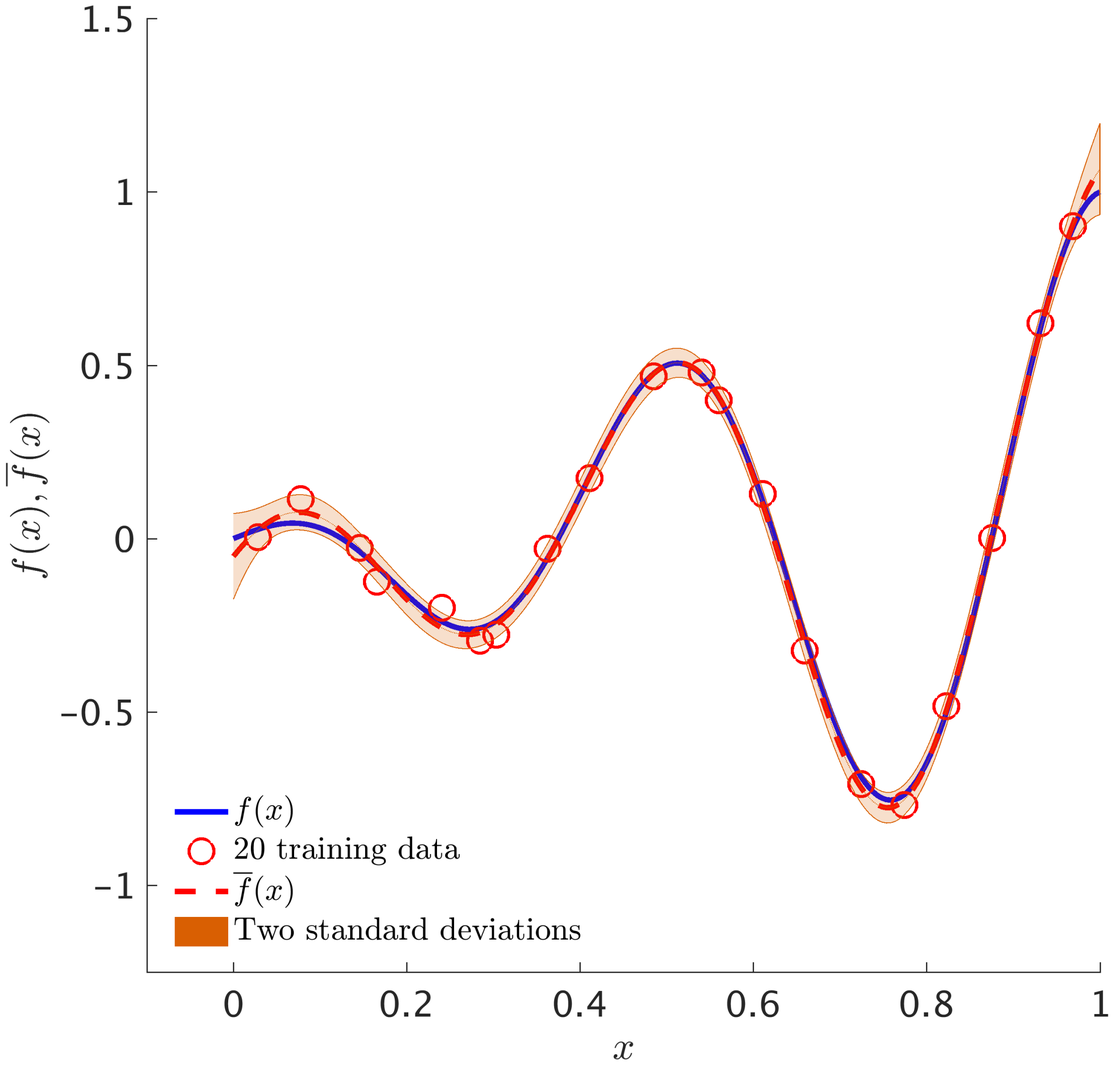
Bayesian Optimization



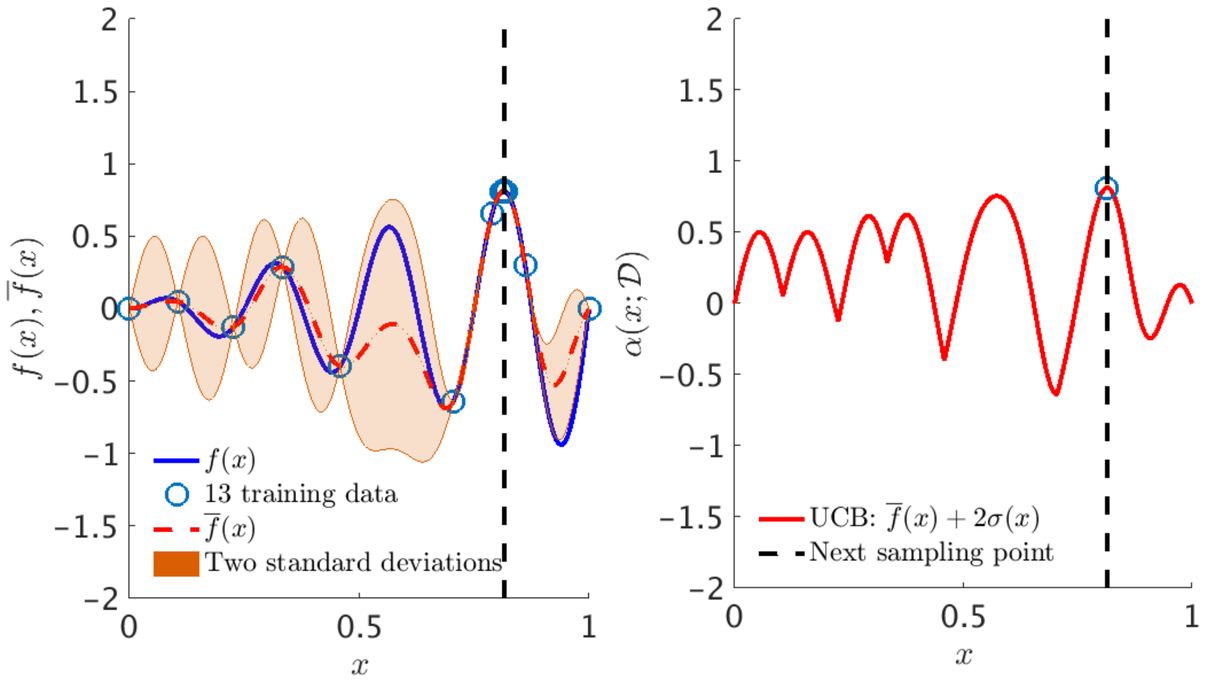
Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

Gaussian Process Regression



Bayesian Optimization



Shahriari, Bobak, et al. “Taking the human out of the loop: A review of bayesian optimization.” *Proceedings of the IEEE* 104.1 (2016): 148-175.

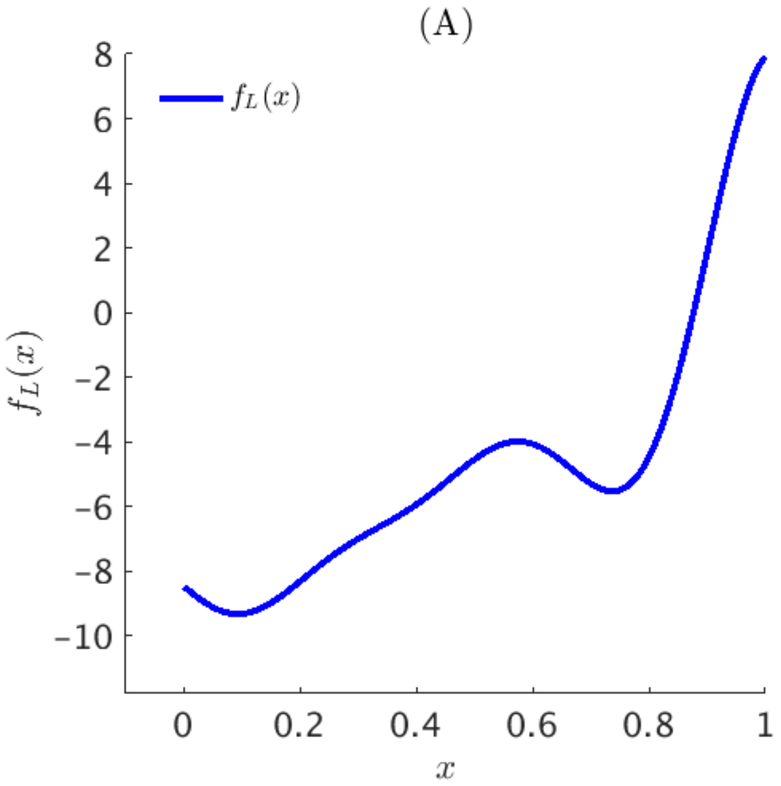
Rasmussen, Carl Edward, and Christopher KI Williams. *Gaussian processes for machine learning*. Vol. 1. Cambridge: MIT press, 2006.

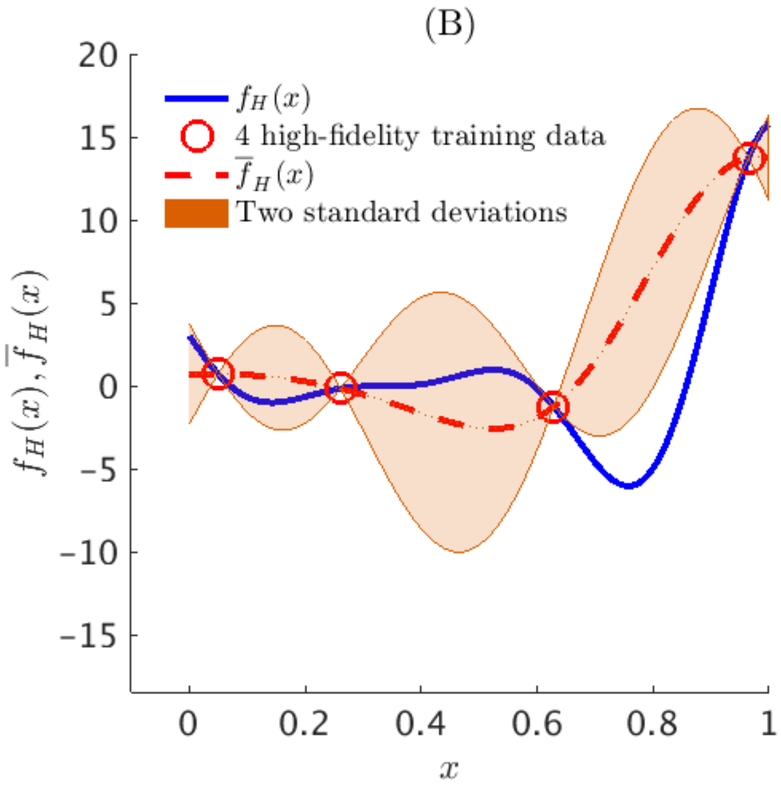
Multi-fidelity Modeling

$$f_H(x) = \rho f_L(x) + \delta(x),$$
$$f_L(x) \sim \mathcal{GP}\left(0, k_1(x, x'; \theta_1)\right),$$
$$\delta(x) \sim \mathcal{GP}\left(0, k_2(x, x'; \theta_2)\right).$$

$$\begin{bmatrix} f_H \\ f_L \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{HH} & k_{HL} \\ k_{LH} & k_{LL} \end{bmatrix}\right),$$

$$k_{HL}(x, x'; \theta_1, \rho) = \rho k_1(x, x'; \theta_1).$$





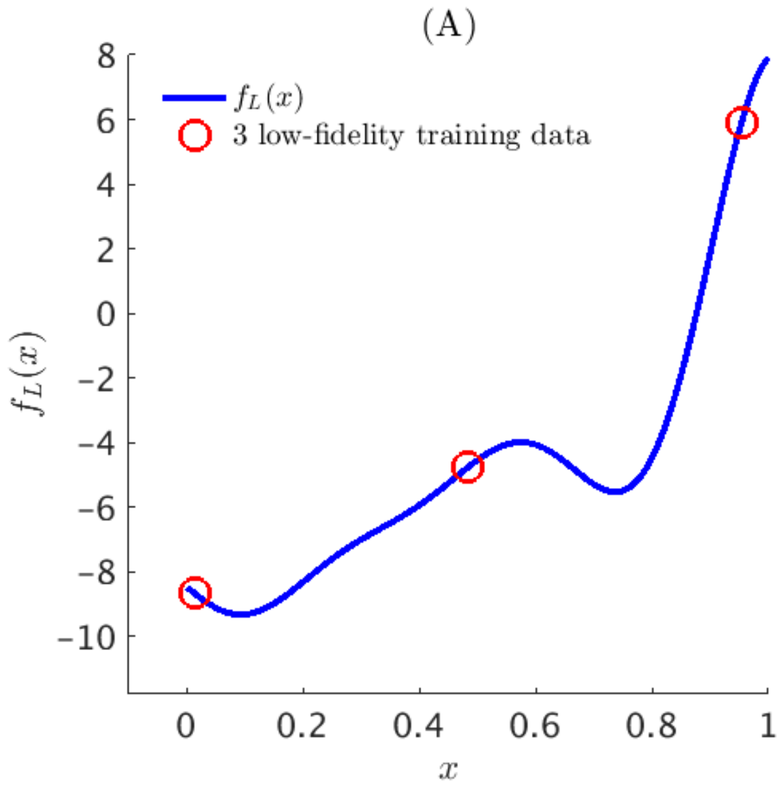
Kennedy, Marc C., and Anthony O'Hagan. “Predicting the output from a complex computer code when fast approximations are available.” *Biometrika* 87.1 (2000): 1-13.

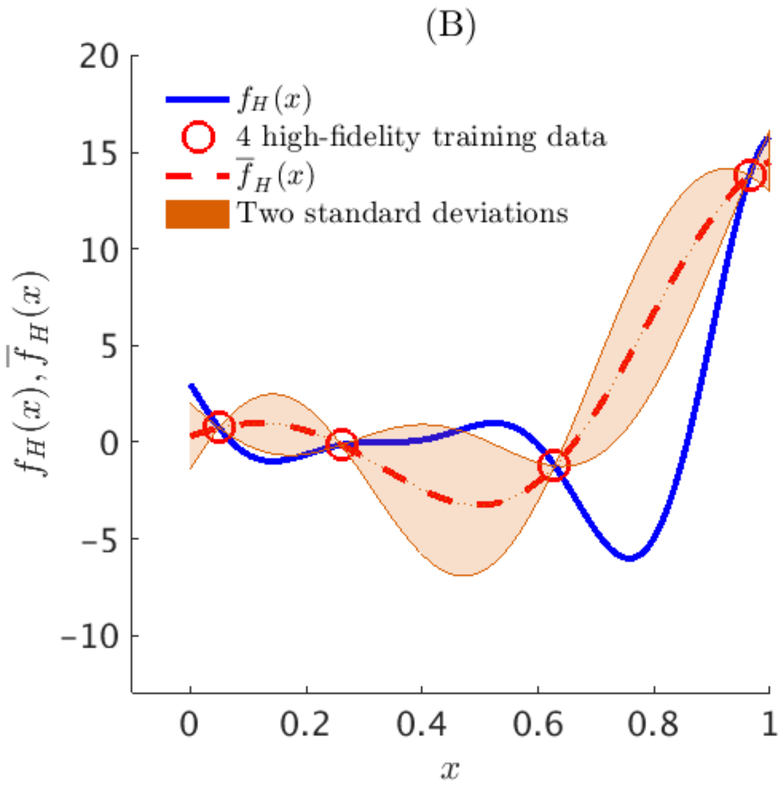
Multi-fidelity Modeling

$$f_H(x) = \rho f_L(x) + \delta(x),$$
$$f_L(x) \sim \mathcal{GP}\left(0, k_1(x, x'; \theta_1)\right),$$
$$\delta(x) \sim \mathcal{GP}\left(0, k_2(x, x'; \theta_2)\right).$$

$$\begin{bmatrix} f_H \\ f_L \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{HH} & k_{HL} \\ k_{LH} & k_{LL} \end{bmatrix}\right),$$

$$k_{HL}(x, x'; \theta_1, \rho) = \rho k_1(x, x'; \theta_1).$$





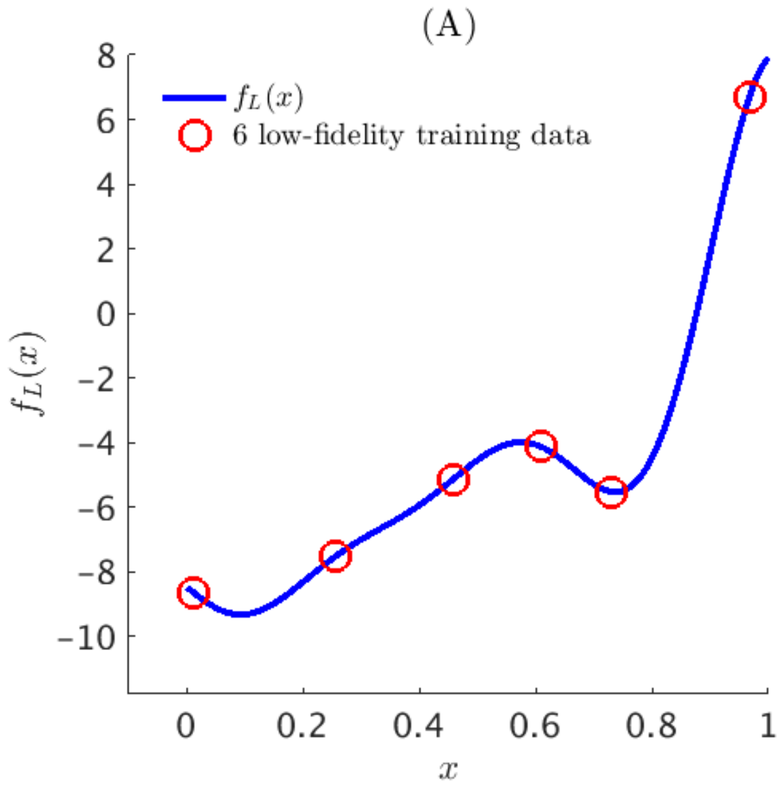
Kennedy, Marc C., and Anthony O'Hagan. “Predicting the output from a complex computer code when fast approximations are available.” *Biometrika* 87.1 (2000): 1-13.

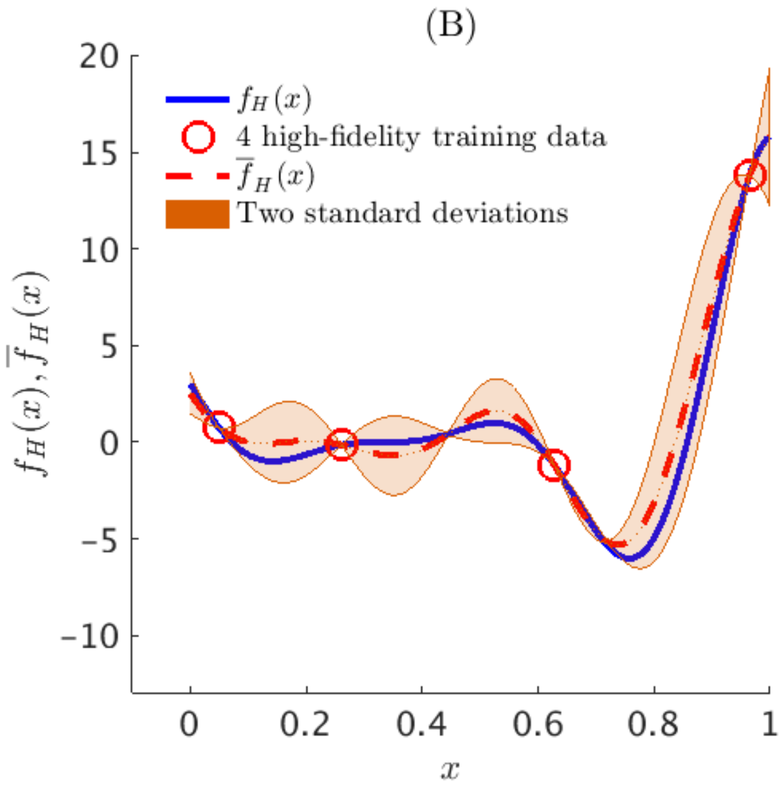
Multi-fidelity Modeling

$$f_H(x) = \rho f_L(x) + \delta(x),$$
$$f_L(x) \sim \mathcal{GP}\left(0, k_1(x, x'; \theta_1)\right),$$
$$\delta(x) \sim \mathcal{GP}\left(0, k_2(x, x'; \theta_2)\right).$$

$$\begin{bmatrix} f_H \\ f_L \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{HH} & k_{HL} \\ k_{LH} & k_{LL} \end{bmatrix}\right),$$

$$k_{HL}(x, x'; \theta_1, \rho) = \rho k_1(x, x'; \theta_1).$$





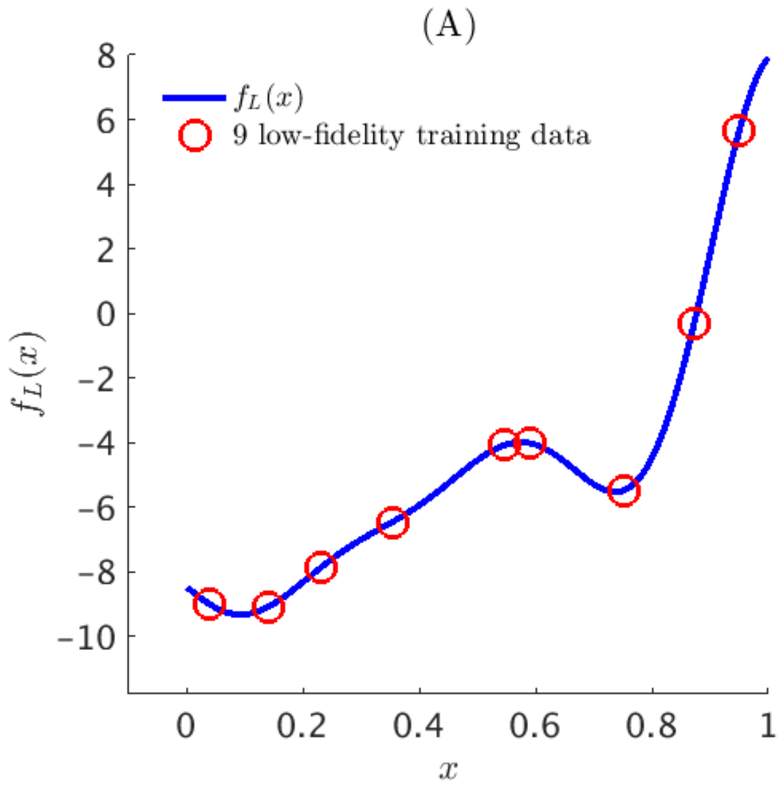
Kennedy, Marc C., and Anthony O'Hagan. “Predicting the output from a complex computer code when fast approximations are available.” *Biometrika* 87.1 (2000): 1-13.

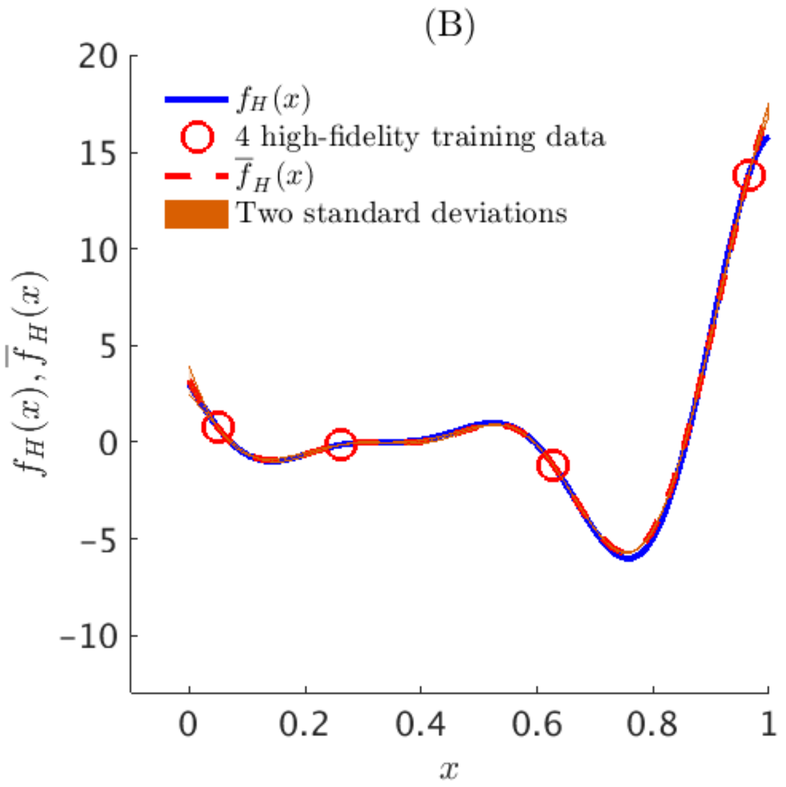
Multi-fidelity Modeling

$$f_H(x) = \rho f_L(x) + \delta(x),$$
$$f_L(x) \sim \mathcal{GP}\left(0, k_1(x, x'; \theta_1)\right),$$
$$\delta(x) \sim \mathcal{GP}\left(0, k_2(x, x'; \theta_2)\right).$$

$$\begin{bmatrix} f_H \\ f_L \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{HH} & k_{HL} \\ k_{LH} & k_{LL} \end{bmatrix}\right),$$

$$k_{HL}(x, x'; \theta_1, \rho) = \rho k_1(x, x'; \theta_1).$$





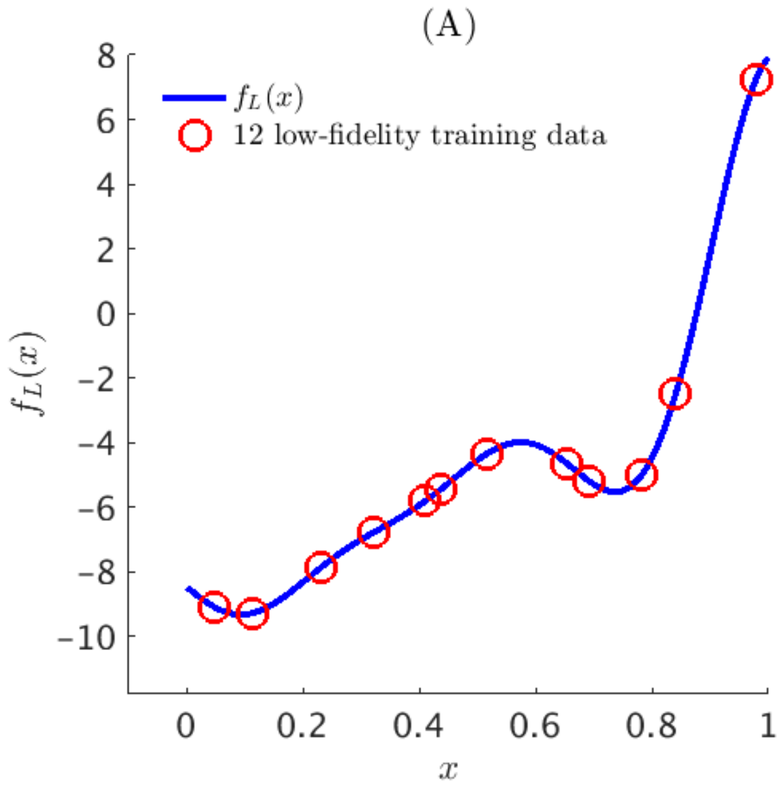
Kennedy, Marc C., and Anthony O'Hagan. “Predicting the output from a complex computer code when fast approximations are available.” *Biometrika* 87.1 (2000): 1-13.

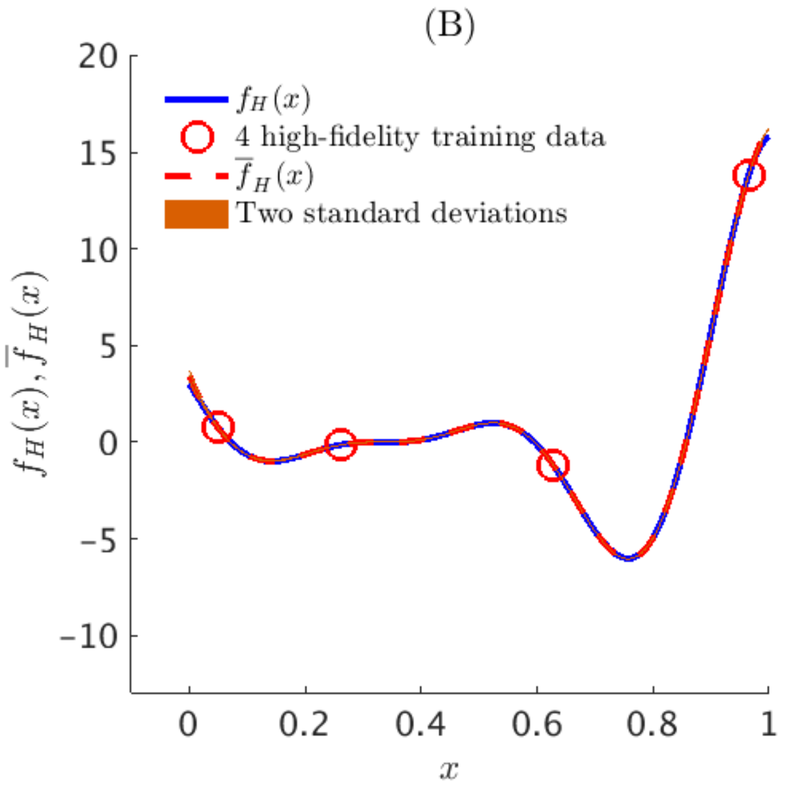
Multi-fidelity Modeling

$$f_H(x) = \rho f_L(x) + \delta(x),$$
$$f_L(x) \sim \mathcal{GP}\left(0, k_1(x, x'; \theta_1)\right),$$
$$\delta(x) \sim \mathcal{GP}\left(0, k_2(x, x'; \theta_2)\right).$$

$$\begin{bmatrix} f_H \\ f_L \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{HH} & k_{HL} \\ k_{LH} & k_{LL} \end{bmatrix}\right),$$

$$k_{HL}(x, x'; \theta_1, \rho) = \rho k_1(x, x'; \theta_1).$$





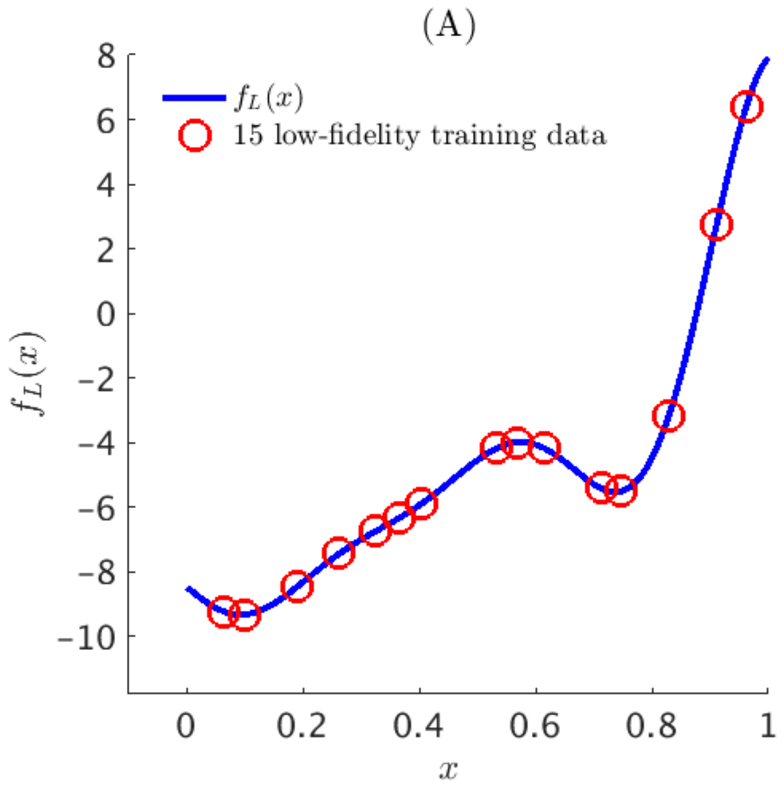
Kennedy, Marc C., and Anthony O'Hagan. “Predicting the output from a complex computer code when fast approximations are available.” *Biometrika* 87.1 (2000): 1-13.

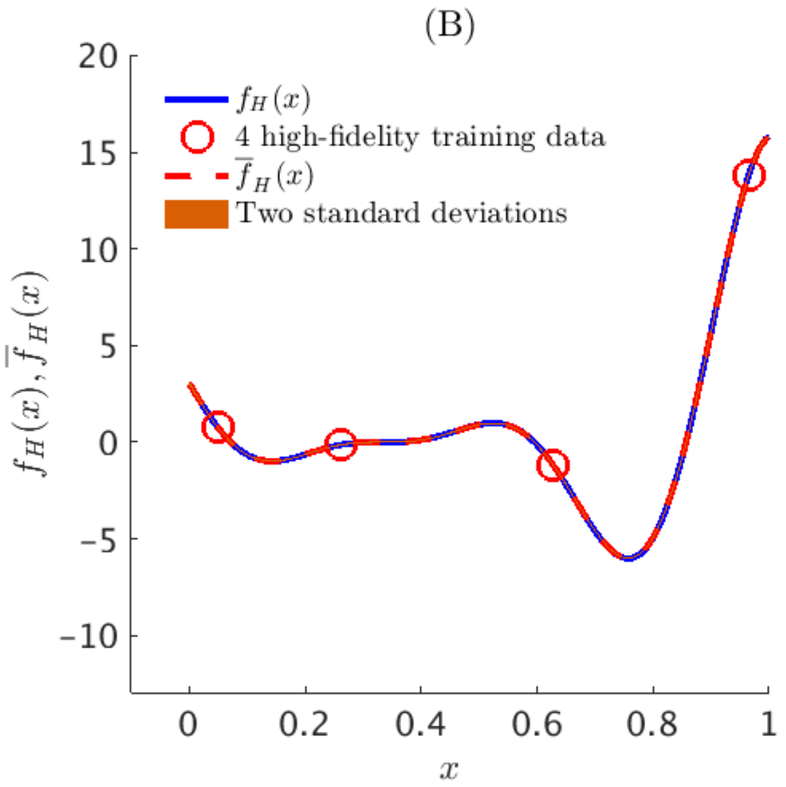
Multi-fidelity Modeling

$$f_H(x) = \rho f_L(x) + \delta(x),$$
$$f_L(x) \sim \mathcal{GP}\left(0, k_1(x, x'; \theta_1)\right),$$
$$\delta(x) \sim \mathcal{GP}\left(0, k_2(x, x'; \theta_2)\right).$$

$$\begin{bmatrix} f_H \\ f_L \end{bmatrix} \sim \mathcal{GP}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} k_{HH} & k_{HL} \\ k_{LH} & k_{LL} \end{bmatrix}\right),$$

$$k_{HL}(x, x'; \theta_1, \rho) = \rho k_1(x, x'; \theta_1).$$

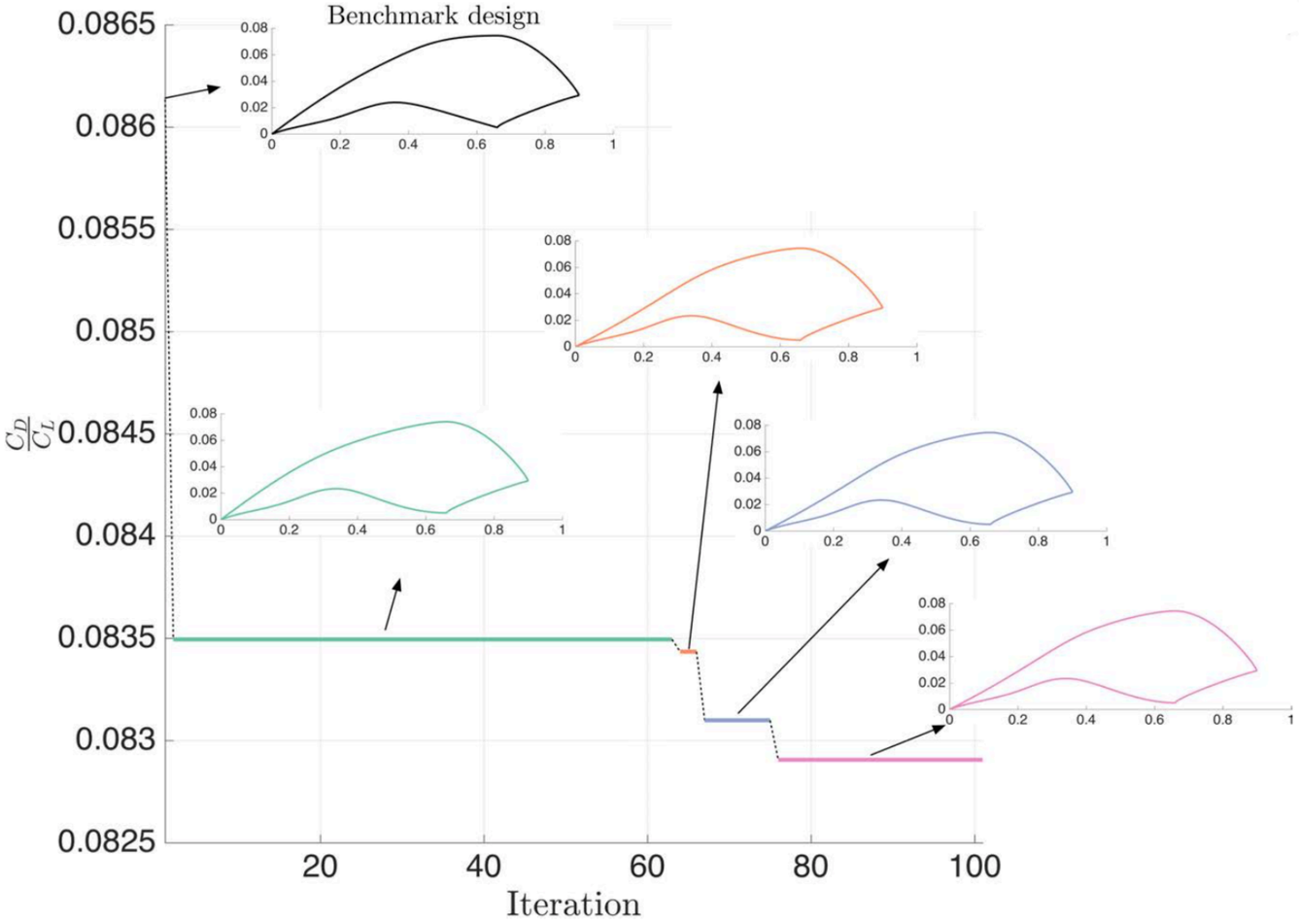




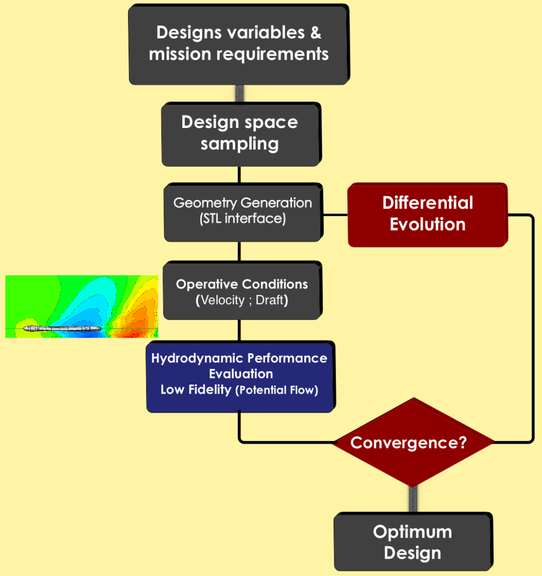
Kennedy, Marc C., and Anthony O'Hagan. “Predicting the output from a complex computer code when fast approximations are available.” *Biometrika* 87.1 (2000): 1-13.

Multi-fidelity Bayesian Optimization

The multi-fidelity framework leads to 30% improvement in performance.



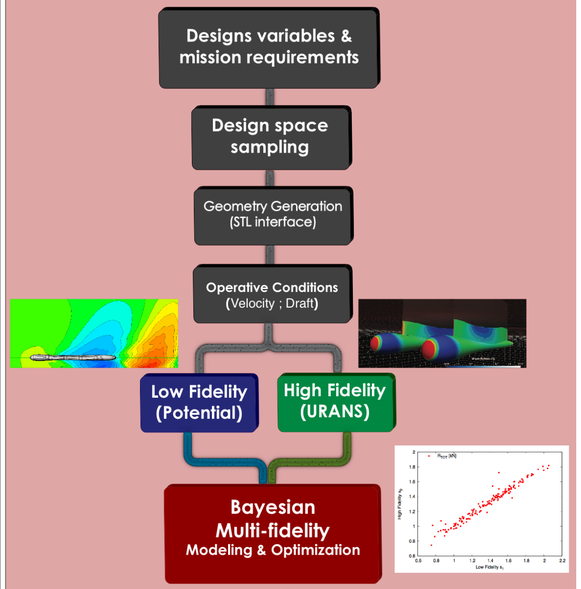
GENETIC ALGORITHM
LOW FIDELITY OPTIMIZATION



No Cases:
60,000 Low Fidelity Simulations

CPU Time (1 CPU):
1 LFsimulations = 180 s

BAYESIAN
MULTI FIDELITY OPTIMIZATION



No Cases:
1,000 Low Fidelity Simulations
100 High Fidelity Simulations

CPU Time (1 CPU):
1 LF simulation = 180 s
1 HF Simulation = 1000 h

Raissi, Maziar, and George Karniadakis. "Deep Multi-fidelity Gaussian Processes." *arXiv preprint arXiv:1604.07484* (2016).

Numerical Gaussian Processes

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0$$

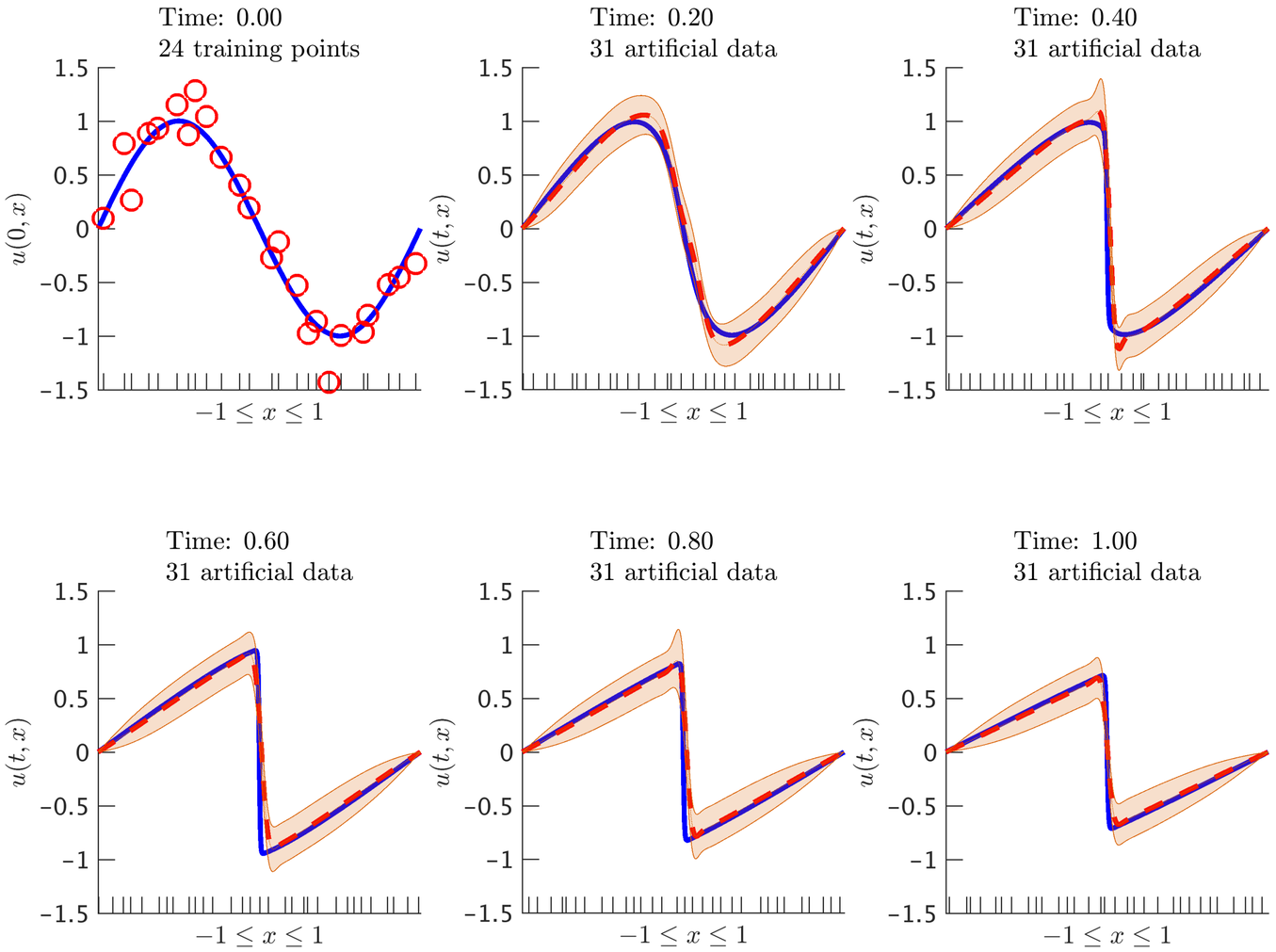
$$u^n + \Delta t\left(u^{n-1}u_x^n - (0.01/\pi)u_{xx}^n\right) = u^{n-1}$$

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

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

$$k^{n-1,n} = k + \Delta t\left(u^{n-1}k_x - (0.01/\pi)k_{xx}\right)$$

Physics Informed Prior



Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations." *SIAM Journal on Scientific Computing* 40.1 (2018): A172-A198.

Numerical Gaussian Processes

$$\begin{bmatrix} \mathbf{u}^n \\ \mathbf{u}^{n-1} \end{bmatrix} \sim \mathcal{N}(0, \mathbf{K})$$

$$\mathbf{K} = \begin{bmatrix} k^{n,n}(\mathbf{x}^n, \mathbf{x}^n; \theta) + \sigma_n^2 \mathbf{I} & k^{n,n-1}(\mathbf{x}^n, \mathbf{x}^{n-1}; \theta) \\ k^{n-1,n}(\mathbf{x}^{n-1}, \mathbf{x}^n; \theta) & k^{n,n-1}(\mathbf{x}^{n-1}, \mathbf{x}^{n-1}; \theta) + \sigma_{n-1}^2 \mathbf{I} \end{bmatrix}$$

$$\min_{\theta, \sigma_n, \sigma_{n-1}} \begin{bmatrix} \mathbf{u}^n \\ \mathbf{u}^{n-1} \end{bmatrix}^T \mathbf{K}^{-1} \begin{bmatrix} \mathbf{u}^n \\ \mathbf{u}^{n-1} \end{bmatrix} + \log |\mathbf{K}|$$

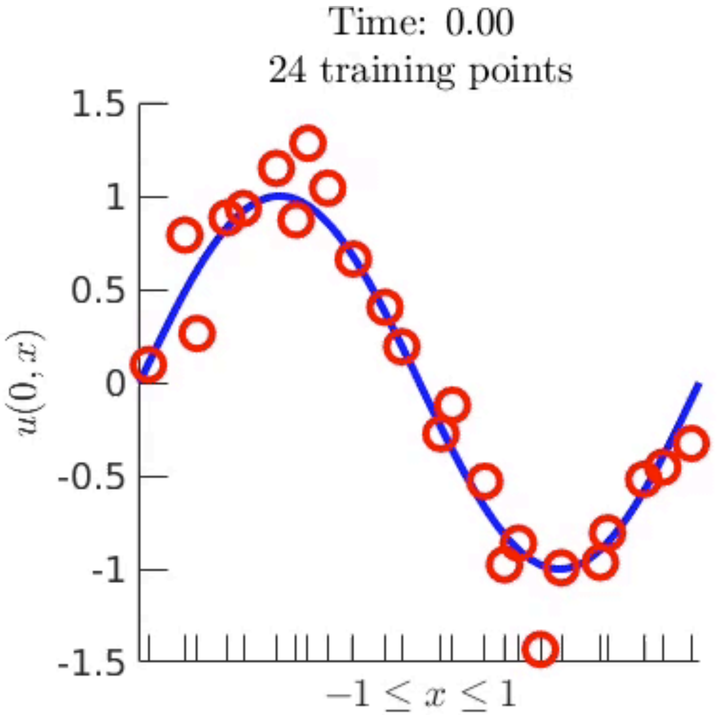
Training

$$u^n(x^*) \mid \mathbf{u}^n \sim \mathcal{N}(m^n(x^*), S^{n,n}(x^*, x^*))$$

$$m^n(x^*) = \mathbf{q}^T \mathbf{K}^{-1} \begin{bmatrix} \mathbf{u}^n \\ \mathbf{m}^{n-1} \end{bmatrix} \qquad \mathbf{q}^T = \begin{bmatrix} k^{n,n}(x^n, \mathbf{x}^n) & k^{n,n-1}(x^n, \mathbf{x}^{n-1}) \end{bmatrix}$$

$$S^{n,n}(x^*, x^*) = k^{n,n}(x^*, x^*) - \mathbf{q}^T \mathbf{K}^{-1} \mathbf{q} + \mathbf{q}^T \mathbf{K}^{-1} \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{S}^{n-1,n-1} \end{bmatrix} \mathbf{K}^{-1} \mathbf{q}$$

Prediction



$$\mathbf{u}^n \sim \mathcal{N}(\mathbf{m}^n, \mathbf{S}^{n,n})$$

$$\mathbf{m}^n = m^n(\mathbf{x}^n)$$
$$\mathbf{S}^{n,n} = S^{n,n}(\mathbf{x}^n, \mathbf{x}^n)$$

Artificial Data

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations." *SIAM Journal on Scientific Computing* 40.1 (2018): A172-A198.

Hidden Physics Models

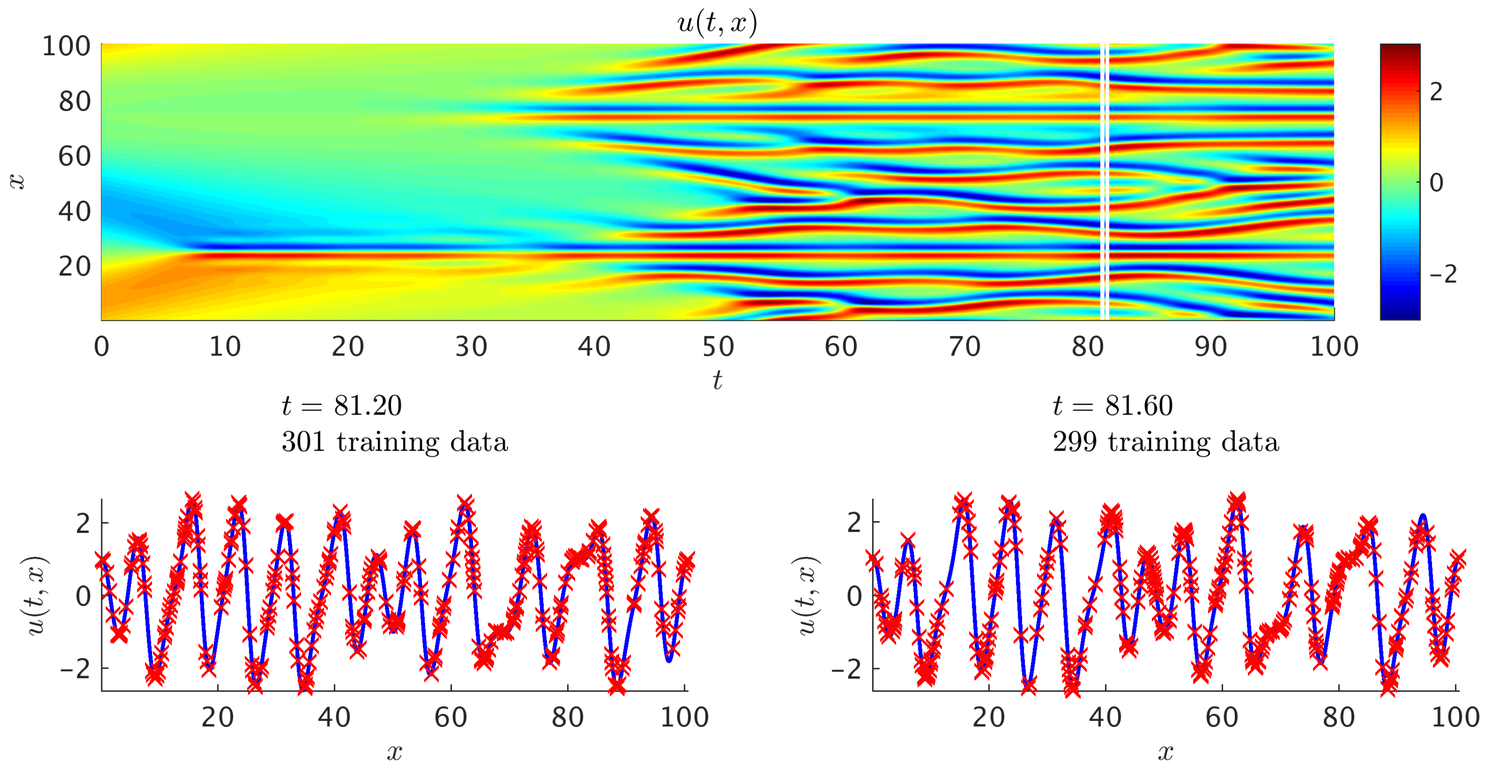
$$u_t + \lambda_1 u u_x + \lambda_2 u_{xx} + \lambda_3 u_{xxxx} = 0$$

$$u^n + \Delta t \left(\lambda_1 u^{n-1} u_x^n + \lambda_2 u_{xx}^n + \lambda_3 u_{xxxx}^n\right) = u^{n-1}$$

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

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

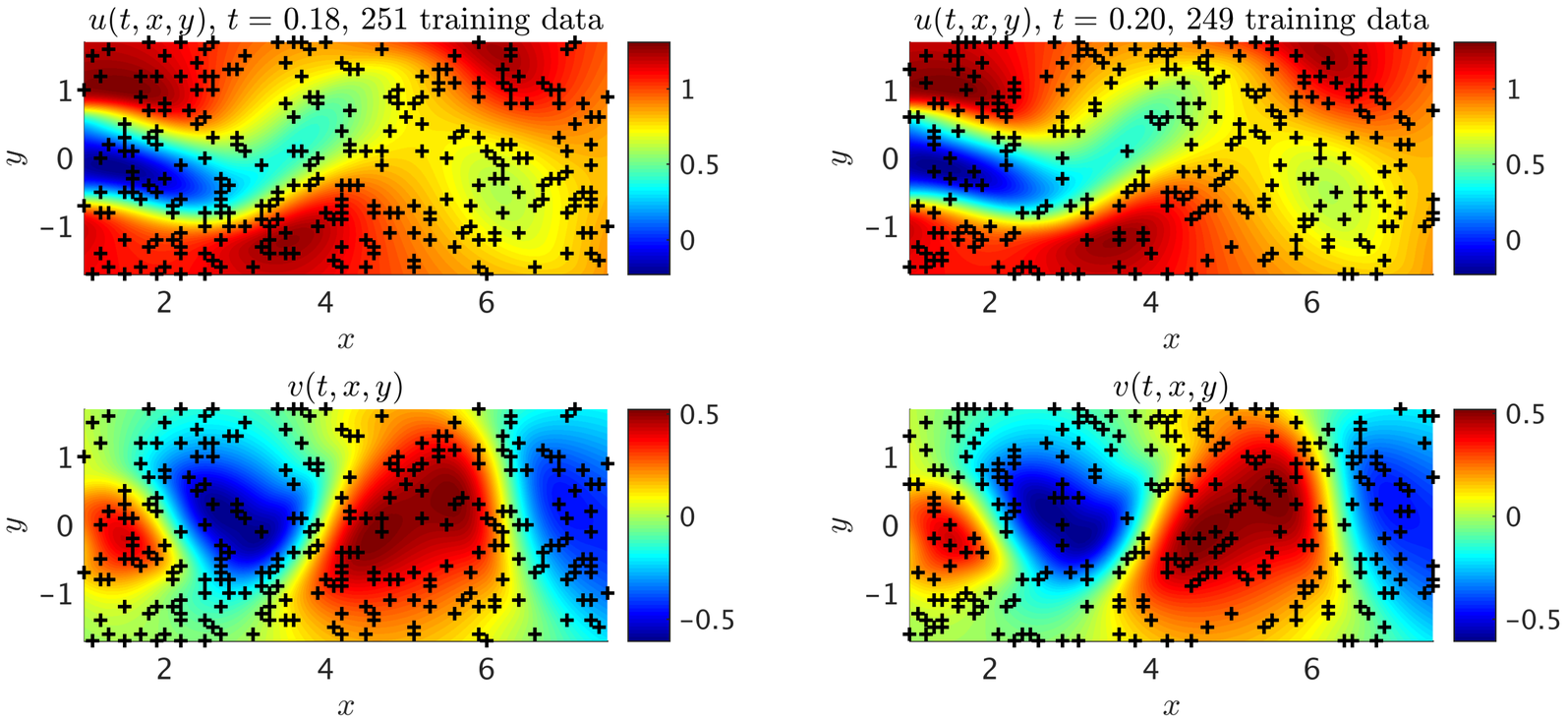
$$k^{n-1,n} = k + \Delta t \left(\lambda_1 u^{n-1} k_x + \lambda_2 k_{xx} + \lambda_3 k_{xxxx}\right)$$

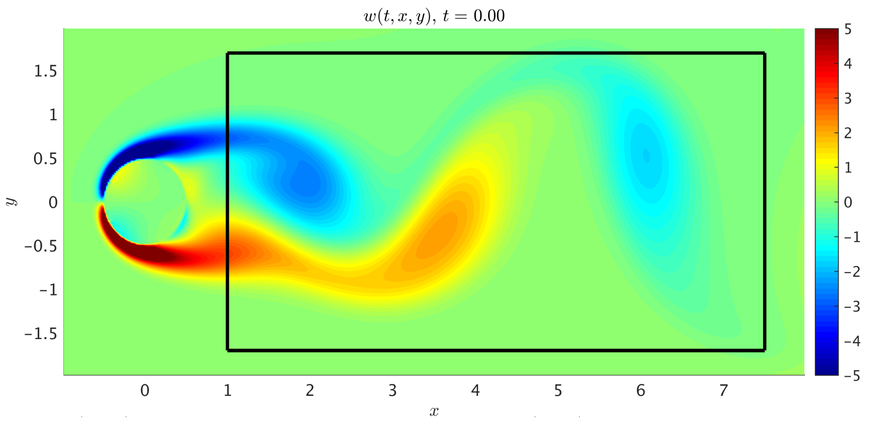


Correct PDE	$u_t + uu_x + u_{xx} + u_{xxxx} = 0$
Identified PDE (clean data)	$u_t + 0.952uu_x + 1.005u_{xx} + 0.980u_{xxxx} = 0$
Identified PDE (1% noise)	$u_t + 0.908uu_x + 0.951u_{xx} + 0.927u_{xxxx} = 0$

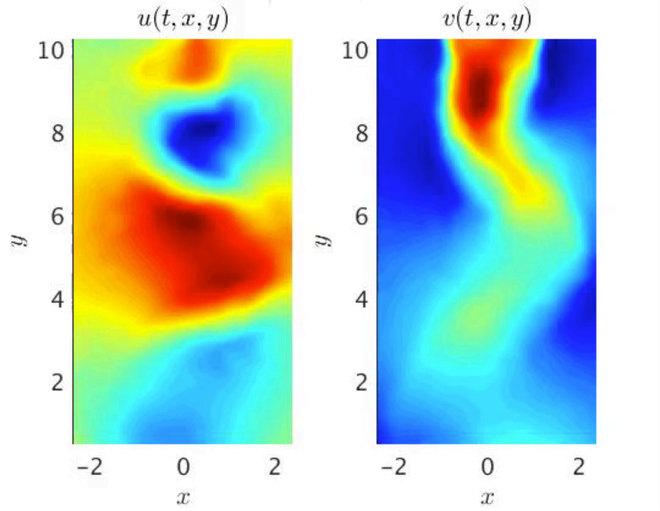
Raissi, Maziar, and George Em Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations." *Journal of Computational Physics* 357 (2018): 125-141.

Hidden Physics Models





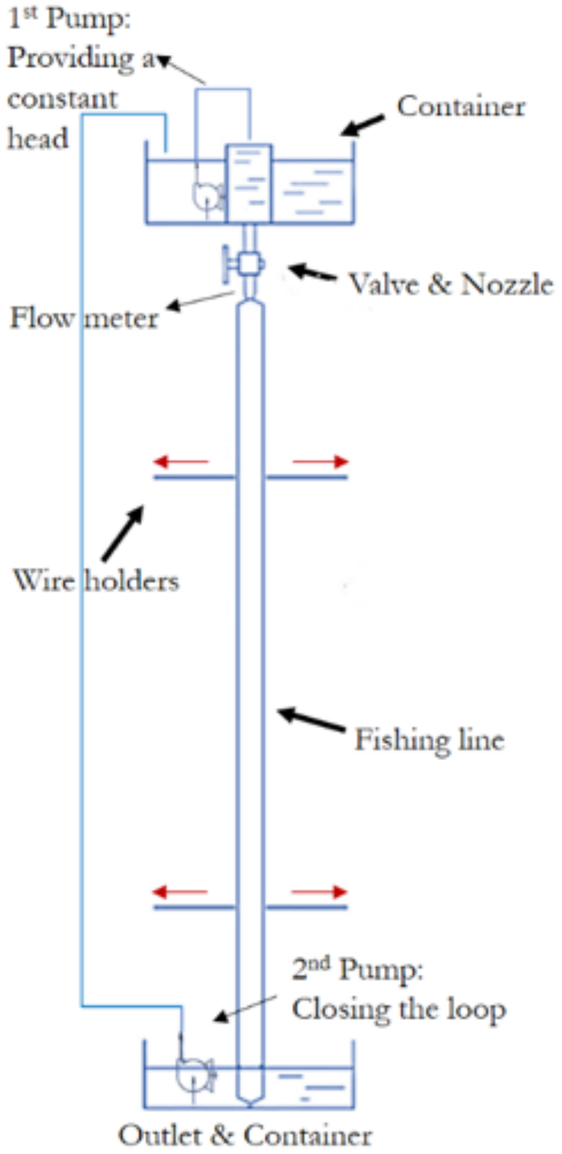
PIV Data

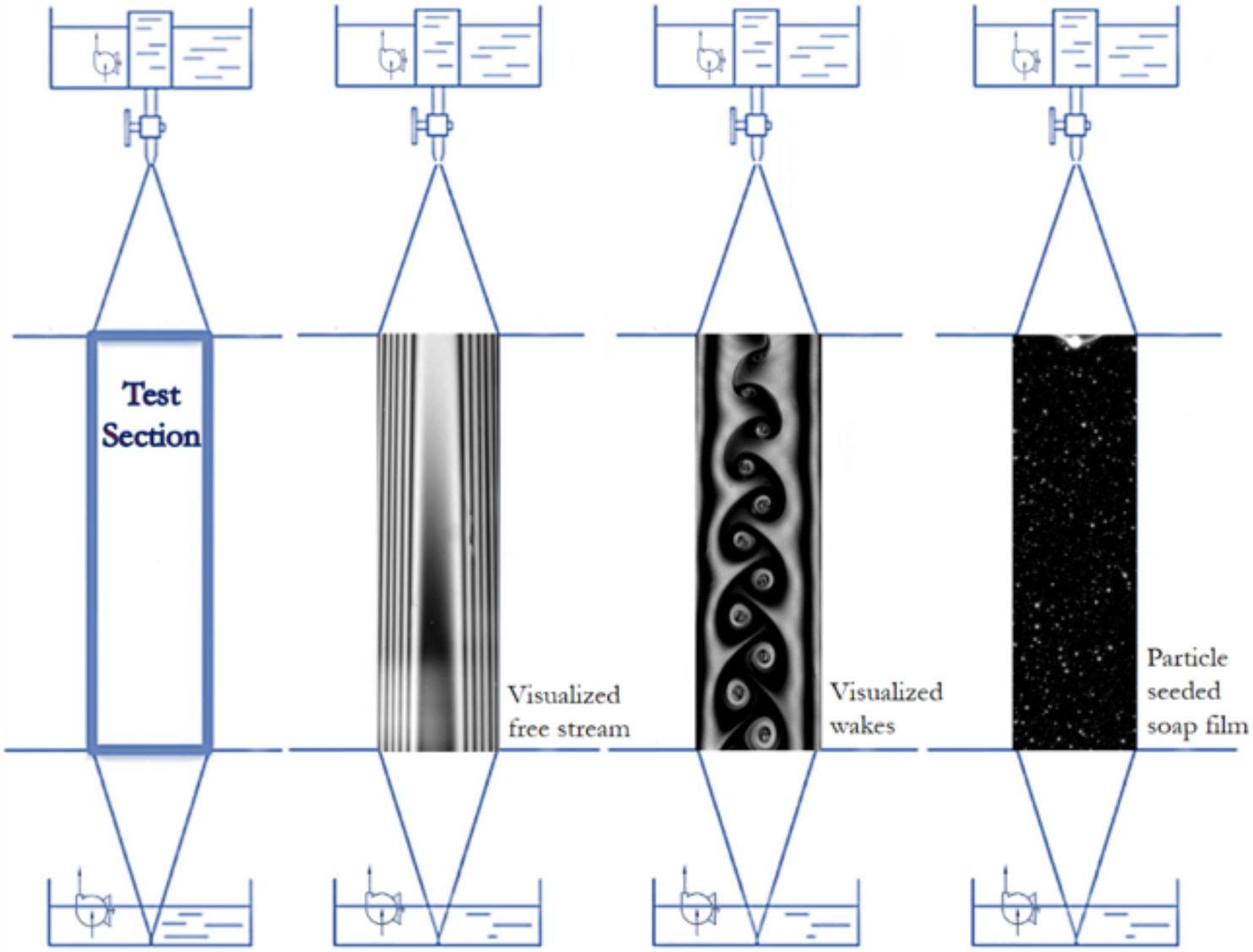


Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.983(uu_x + vu_y) = -p_x + 0.00826(u_{xx} + u_{yy})$ $v_t + 0.983(uv_x + vv_y) = -p_y + 0.00826(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.849(uu_x + vu_y) = -p_x + 0.01399(u_{xx} + u_{yy})$ $v_t + 0.849(uv_x + vv_y) = -p_y + 0.01399(v_{xx} + v_{yy})$

Raissi, Maziar, and George Em Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations." *Journal of Computational Physics* 357 (2018): 125-141.

Hidden Physics Models

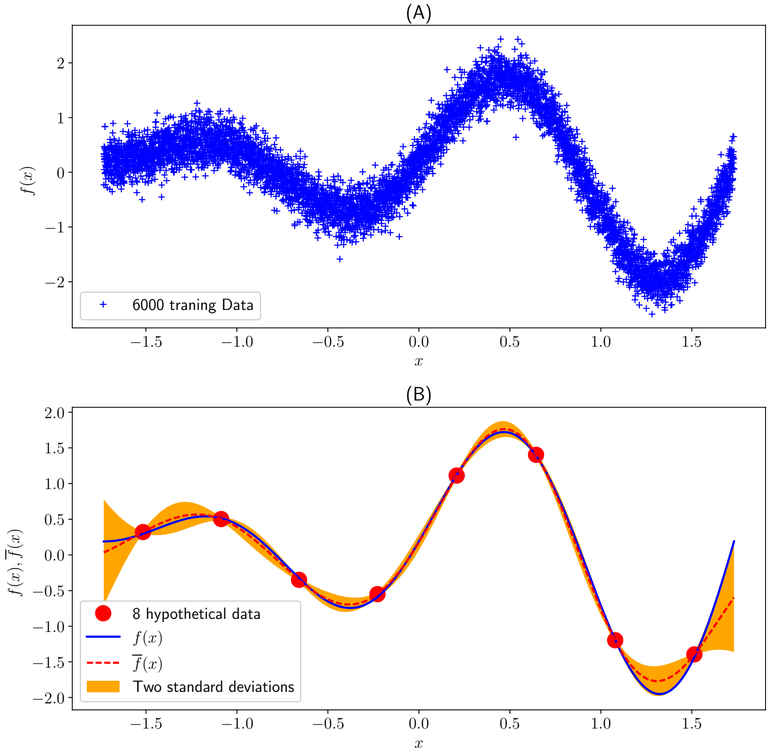


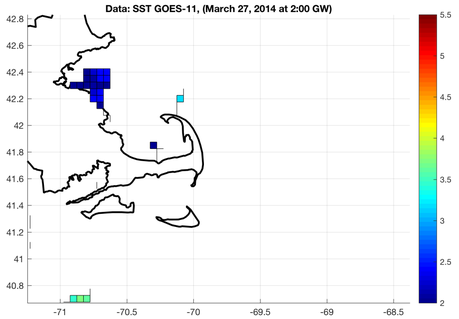
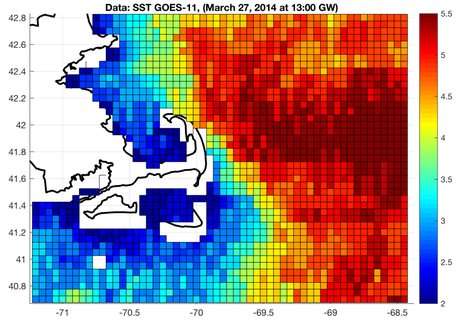
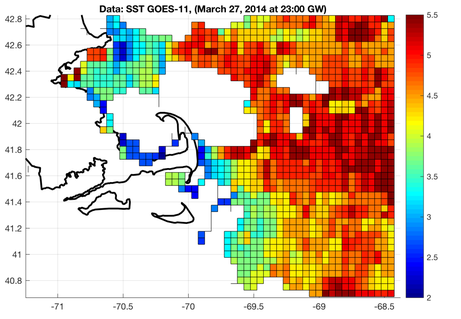
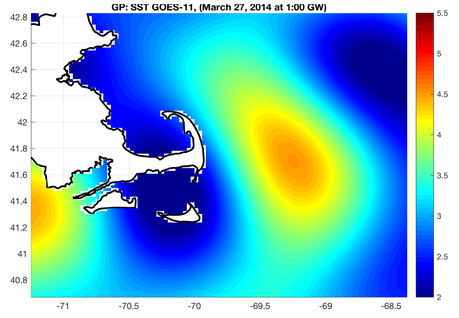
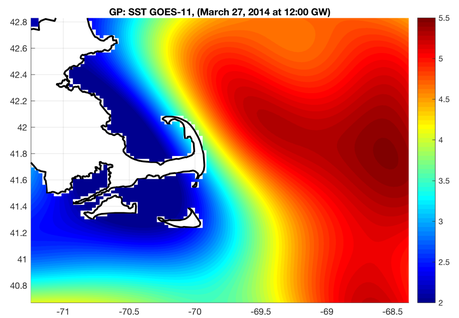
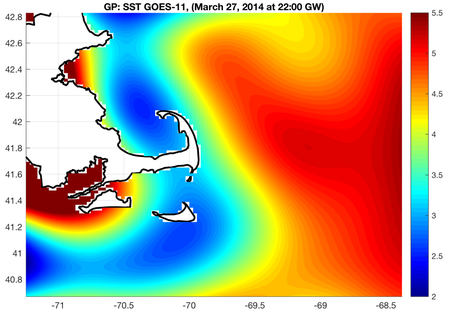
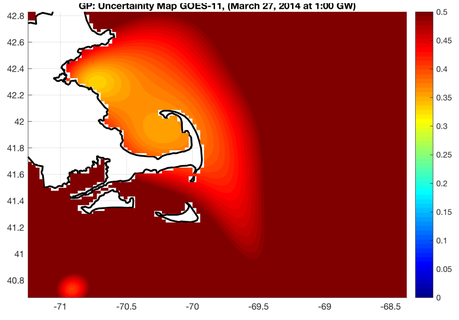
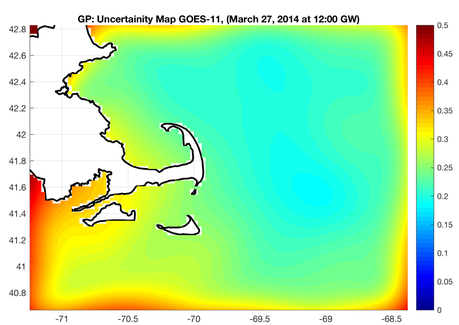
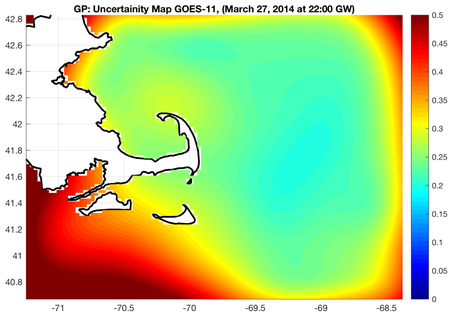


Parametric Gaussian Processes for Big Data

1) Parametric Gaussian process
(producer of the hypothetical data)

2) Classical Gaussian process
(consumer of the hypothetical data)



	March 27, 2014 - 2:00 GW	March 27, 2014 - 13:00 GW	March 27, 2014 - 22:00 GW
GOES_11 Satellite			
Prediction			
Uncertainty Map			

Raissi, Maziar. "Parametric Gaussian Process Regression for Big Data." *Computational Mechanics* (2018).

Neural Networks Regression

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right), \quad i = 1, \ldots, N$$

$$\mathbf{y} = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}\right)$$

Data

$$f(x) = W^L h^L + b^L$$

$$h^L = \tanh(W^{L-1} h^{L-1} + b^{L-1})$$

$$\vdots$$

$$h^1 = \tanh(W^0 x + b^0)$$

Prior

$$\mathbf{y} \sim \mathcal{N}\left(f(\mathbf{x}), \sigma^2 \mathbf{I}\right)$$

$$\min_{W,b} \ (\mathbf{y} - f(\mathbf{x}))'(\mathbf{y} - f(\mathbf{x}))$$

$$\min_{W,b} \ \sum_{i=1}^{N} |y_i - f(x_i)|^2$$

Training

$$f(x^*)$$

Prediction

Gaussian Processes vs Neural Networks

1) GPs are non-parametric while NNs are parametric regressors.

2) GPs are not scalable to Big dataset while NNs are scalable.

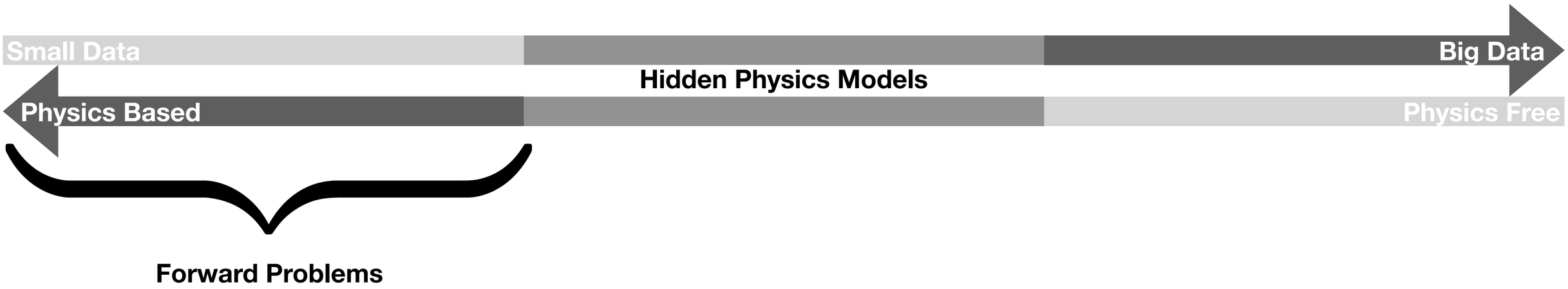
3) GPs quantify the uncertainty in their predictions while NNs do not.

4) GPs automatically balance the tradeoff between data fit and model complexity.

Goodfellow, Ian, et al. *Deep learning*. Vol. 1. Cambridge: MIT press, 2016.

Boulder

Forward Problems



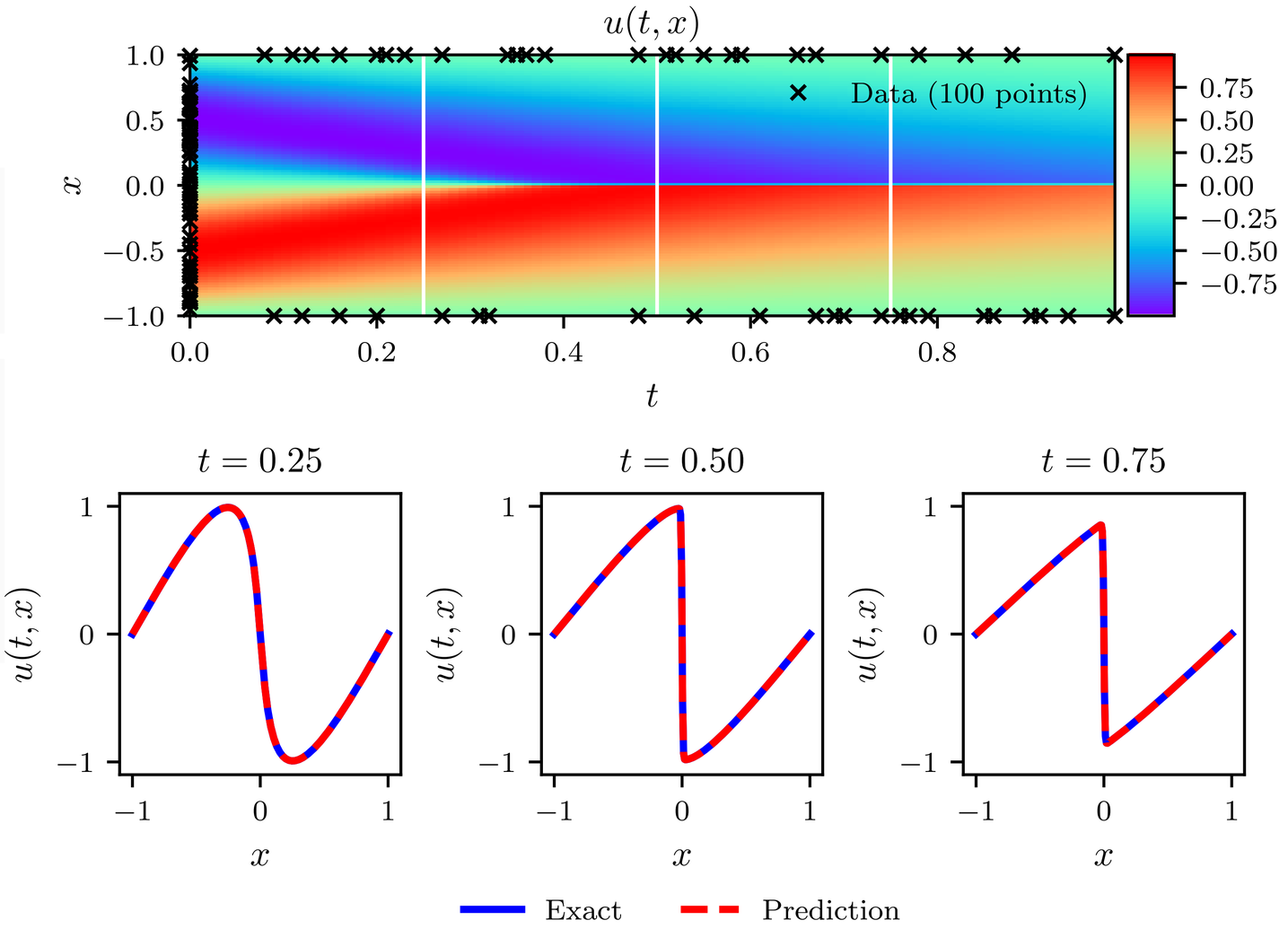
Physics Informed Neural Networks (PINNs)

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0$$

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

```
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
```

$$\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 + \sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2$$

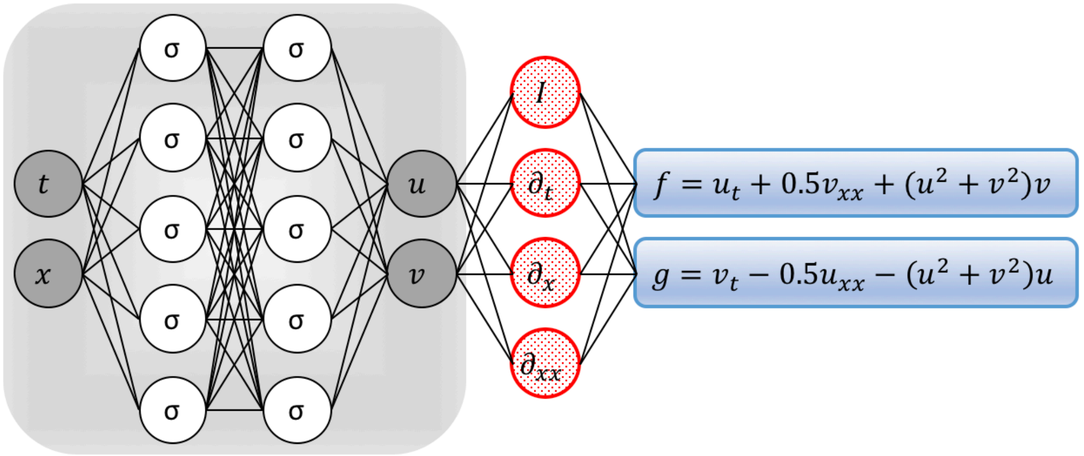


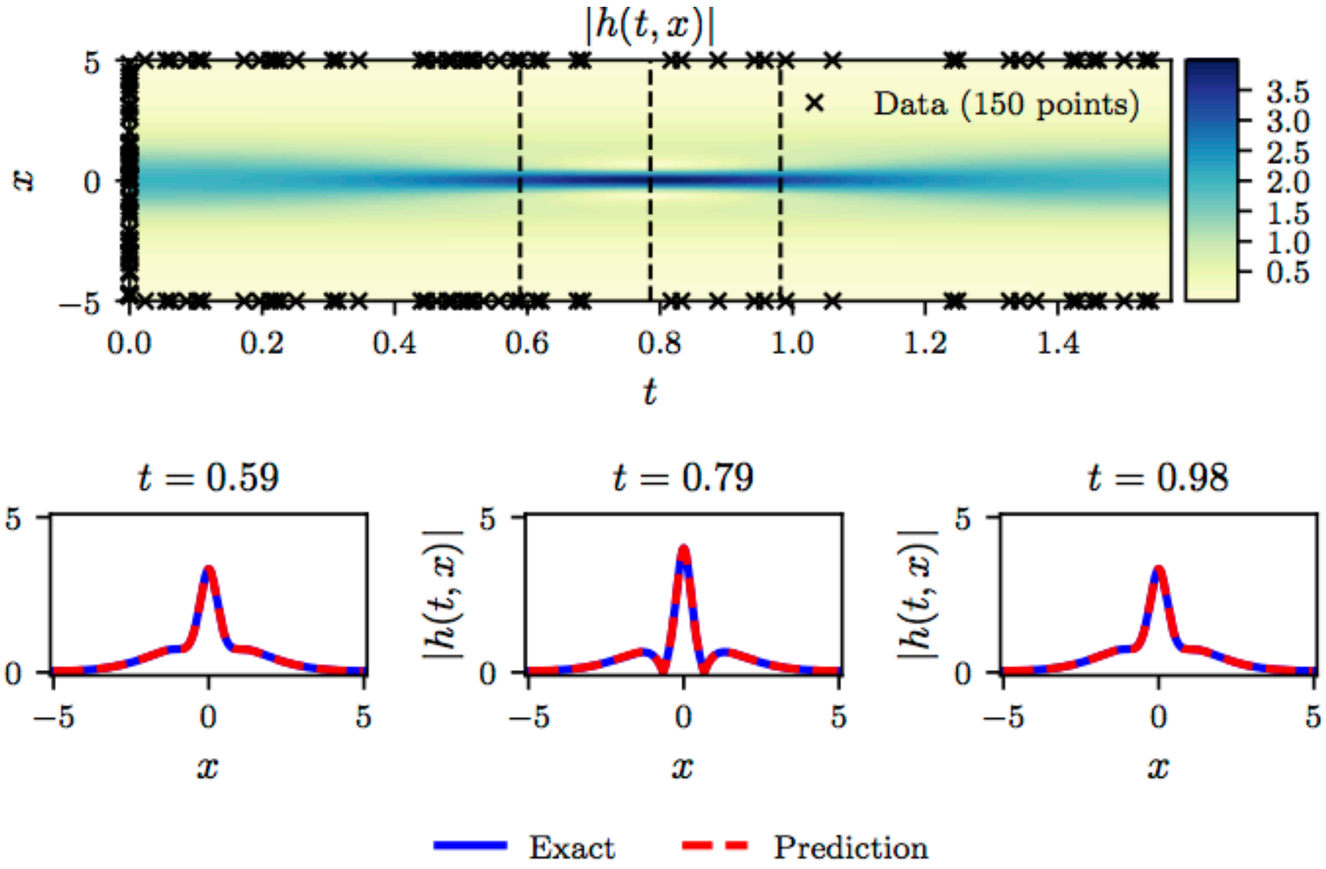
Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations." *Journal of Computational Physics* (2018).

Physics Informed Neural Networks (PINNs)

$$ih_t + 0.5h_{xx} + |h|^2 h = 0$$

$$u = \text{Real}(h), \quad v = \text{Imag}(h)$$





$$MSE = MSE_0 + MSE_b + MSE_f + MSE_g$$

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations." *Journal of Computational Physics* (2018).

Boulder

Physics Informed Neural Networks (PINNs)

	FEM/FVM/FDM	PINNs	ROMs
Solution Space	Basis Functions	Neural Networks	Smart Basis Functions
Differential Operators	Discretization/Weak-form	Automatic Differentiation	Discretization/Weak-form
Solver	Linear/non-linear/Iterative	Gradient Descent (Training)	Linear/non-linear/Iterative
Evaluate	Interpolation	Inference	Interpolation

Forward-Backward Stochastic Neural Networks

Deep Learning of High-dimensional Partial Differential Equations

$$
\begin{aligned}
dX_t &= \mu(t, X_t, Y_t, Z_t)dt + \sigma(t, X_t, Y_t)dW_t \\
X_0 &= \xi \\
dY_t &= \varphi(t, X_t, Y_t, Z_t)dt + Z_t'\sigma(t, X_t, Y_t)dW_t \\
Y_T &= g(X_T)
\end{aligned}
$$

$$
\begin{aligned}
Y_t &= u(t, X_t) \\
Z_t &= Du(t, X_t)
\end{aligned}
$$

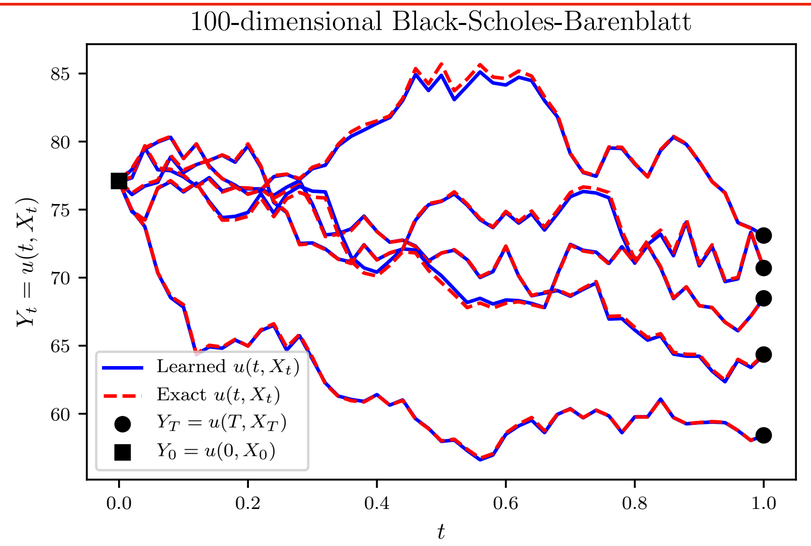
Forward-Backward Stochastic Differential Equation

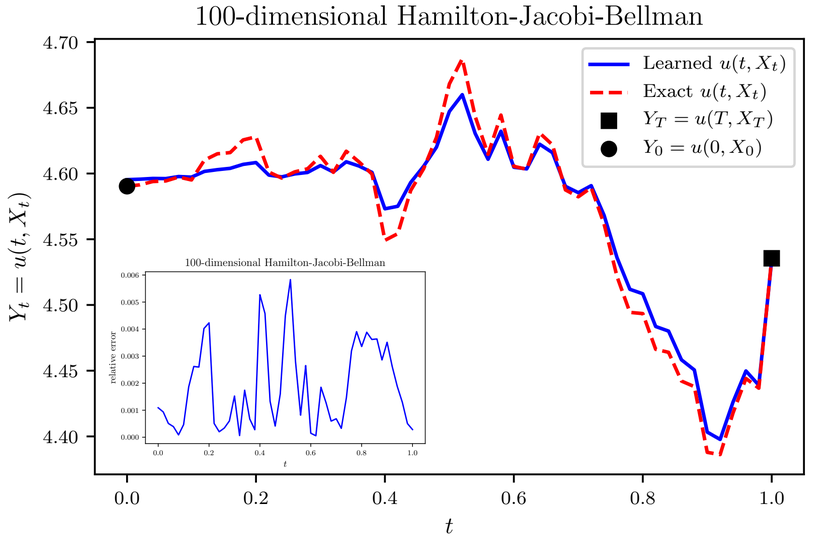
$u(t,x)$ → **Deep Neural Network**

$Du(t,x)$ → **Automatic Differentiation**

$$
\begin{aligned}
&\frac{\partial u}{\partial t} = \varphi(t, x, u, Du) - \mu(t, x, u, Du)'Du - \frac{1}{2}\mathrm{Tr}[\sigma(t, x, u)\sigma(t, x, u)'D^2u] \\
&u(T, x) = g(x)
\end{aligned}
$$

Quasi-Linear Partial Differential Equation

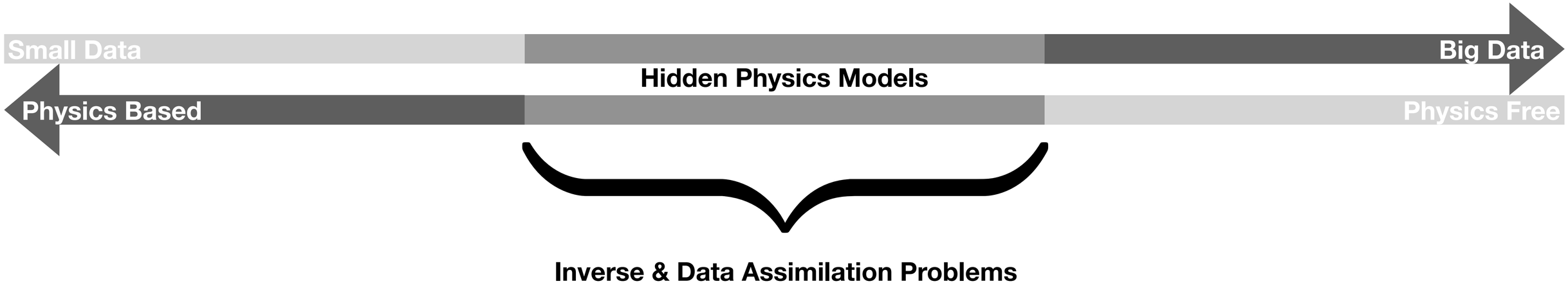




Raissi, Maziar. "Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations." *arXiv preprint arXiv:1804.07010* (2018).

Boulder

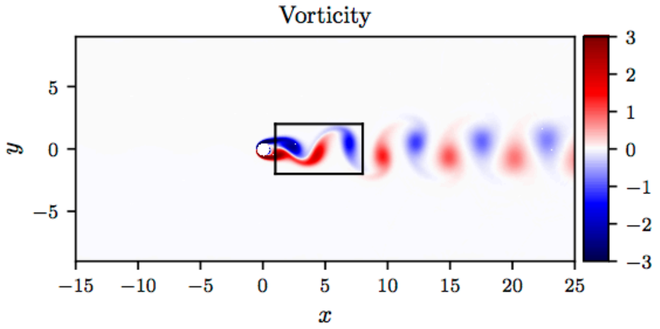
Inverse & Data Assimilation Problems

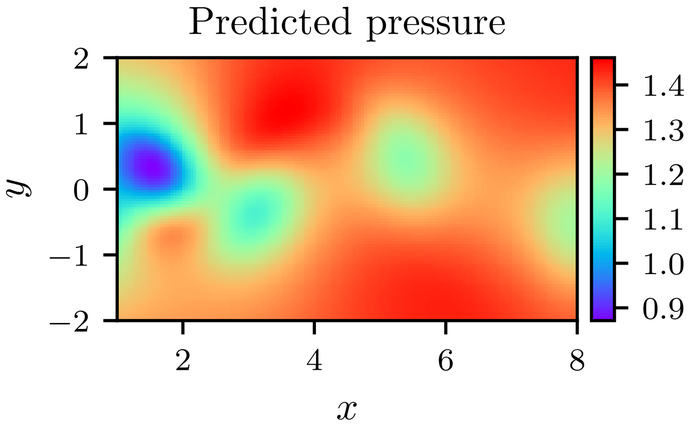


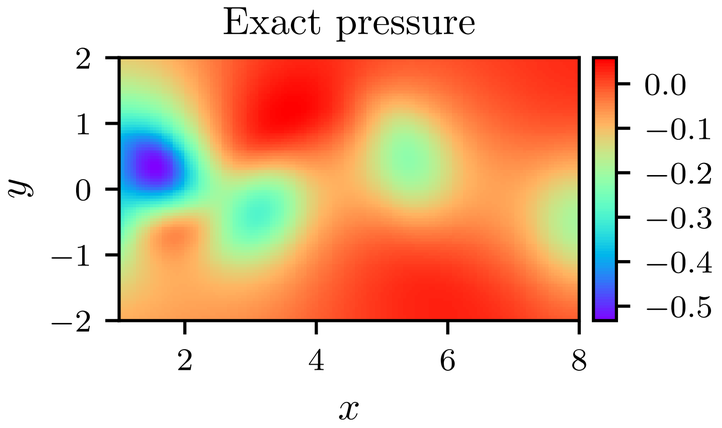
Physics Informed Neural Networks (PINNs)

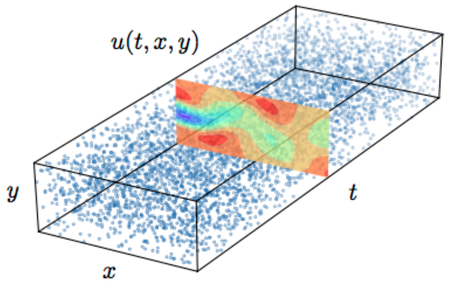
$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy})$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy})$$
$$u_x + v_y = 0$$

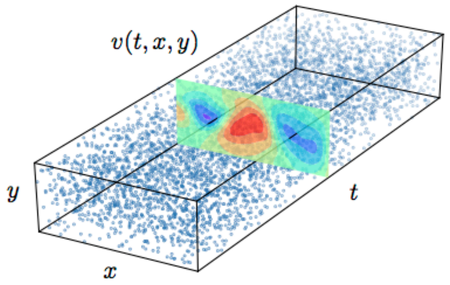
$$f := u_t + \lambda_1(uu_x + vu_y) + p_x - \lambda_2(u_{xx} + u_{yy})$$
$$g := v_t + \lambda_1(uv_x + vv_y) + p_y - \lambda_2(v_{xx} + v_{yy})$$
$$h := u_x + v_y$$







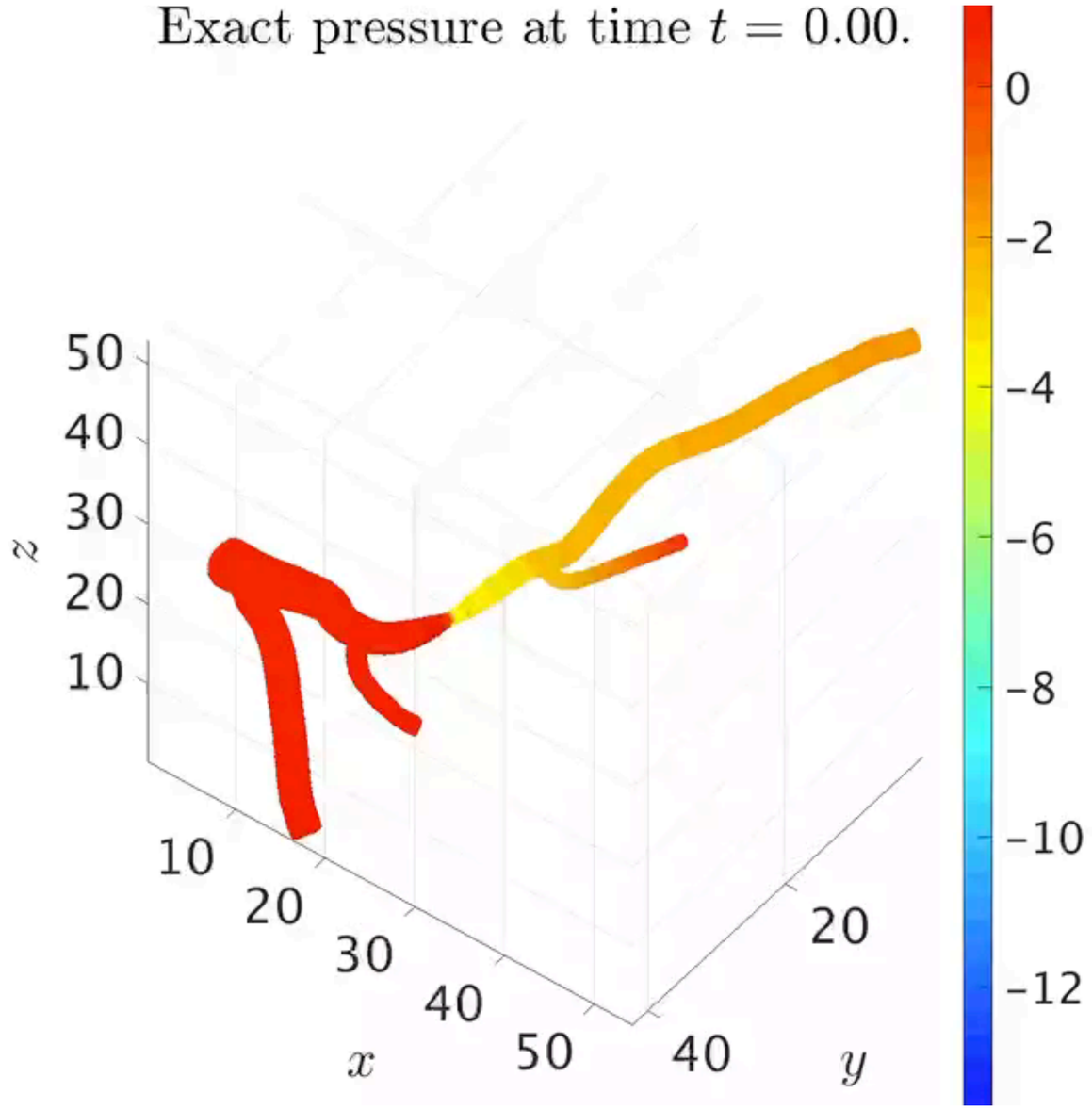


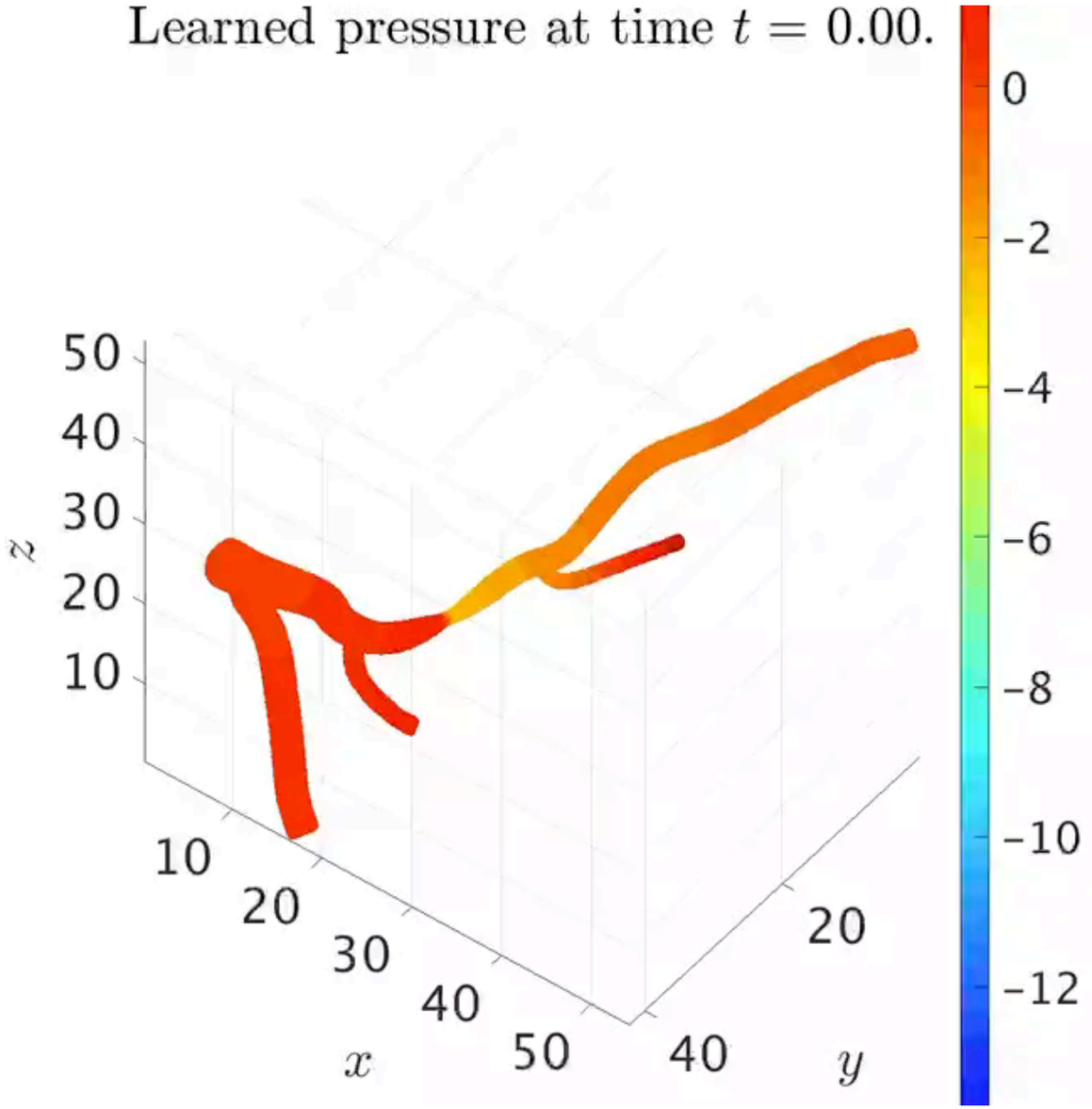


Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t +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

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations." *Journal of Computational Physics* (2018).

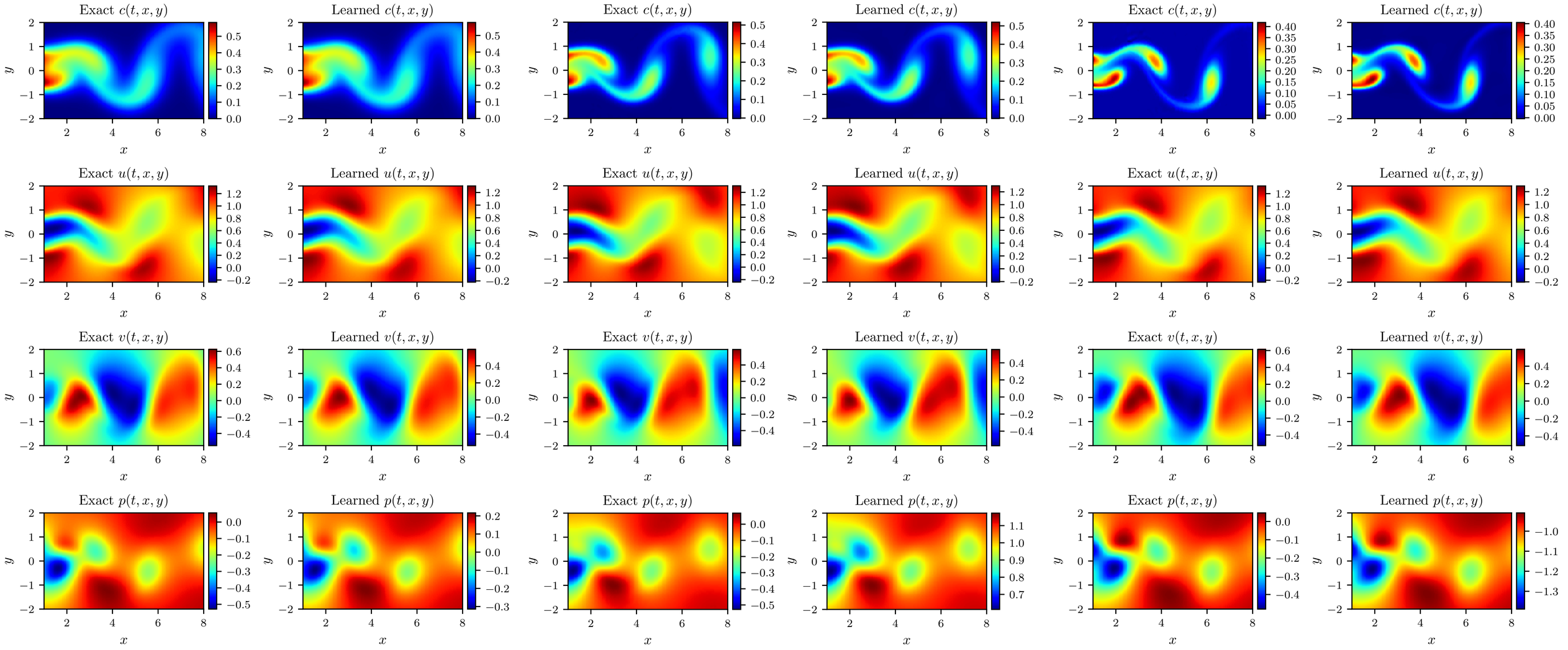
Physics Informed Neural Networks (PINNs)





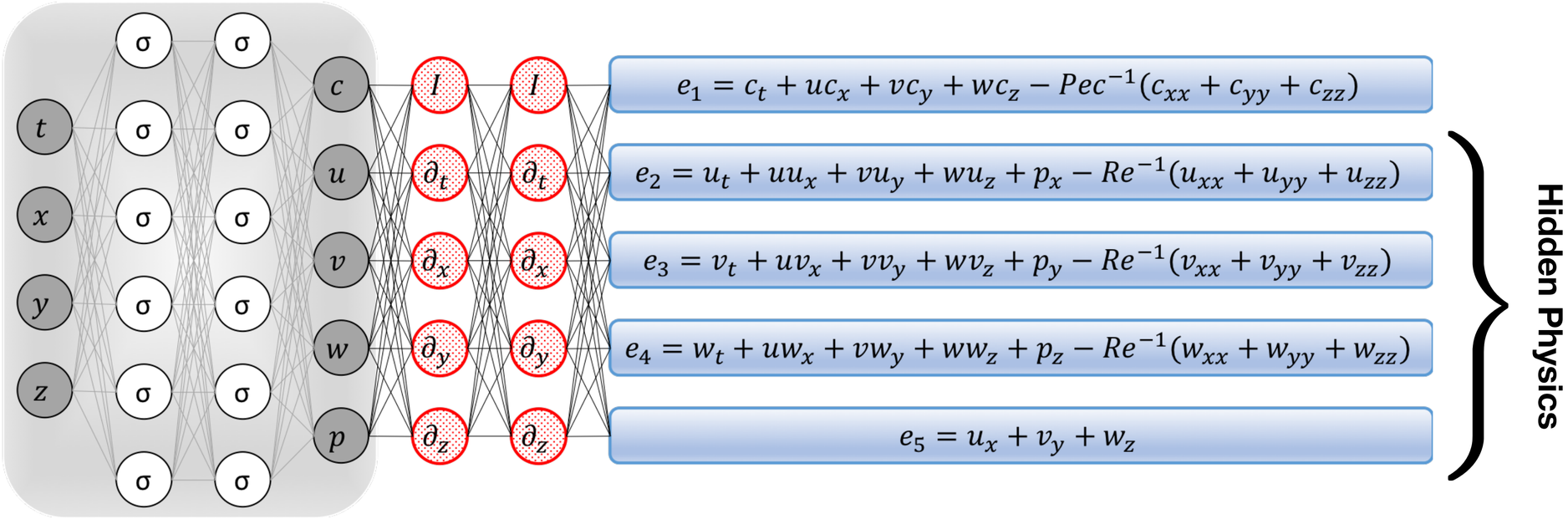
Boulder

Hidden Fluid Mechanics

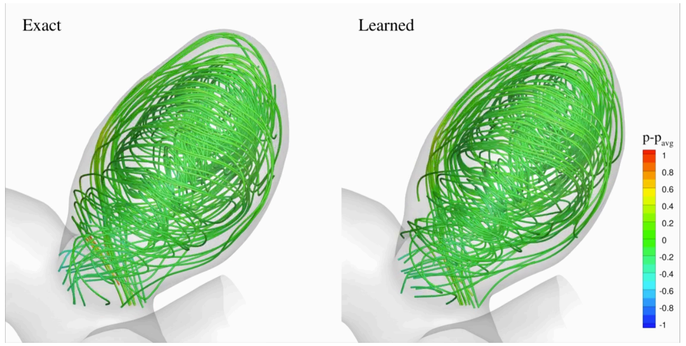


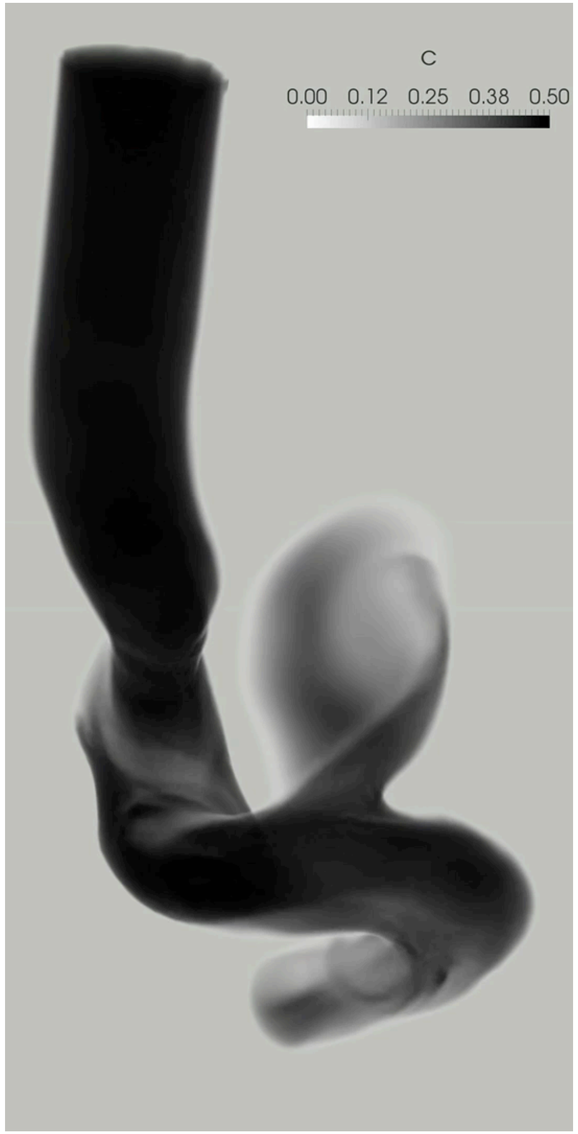
Pec = 40 **Pec = 100** **Pec = 250**

Hidden Fluid Mechanics



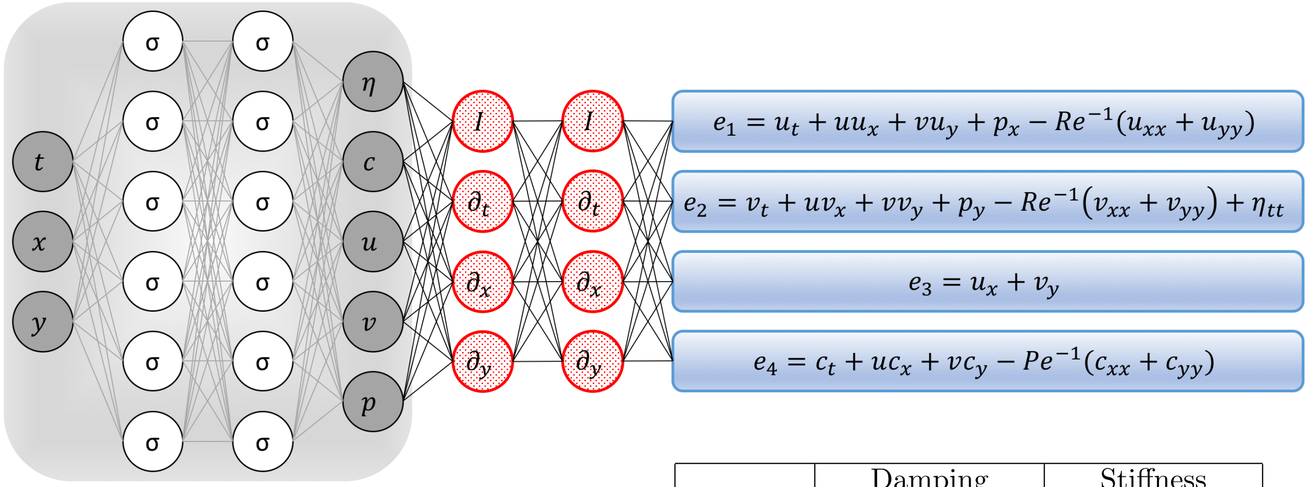
$$SSE = \sum_{n=1}^{N} |c(t^n, x^n, y^n, z^n) - c^n|^2 + \sum_{i=1}^{5} \sum_{n=1}^{N} |e_i(t^n, x^n, y^n, z^n)|^2$$



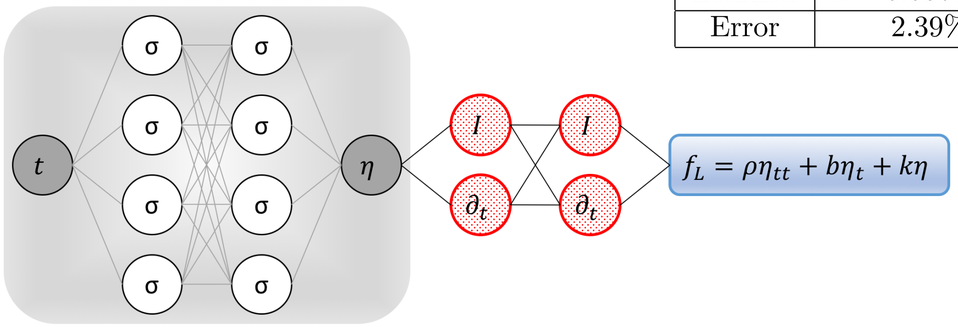


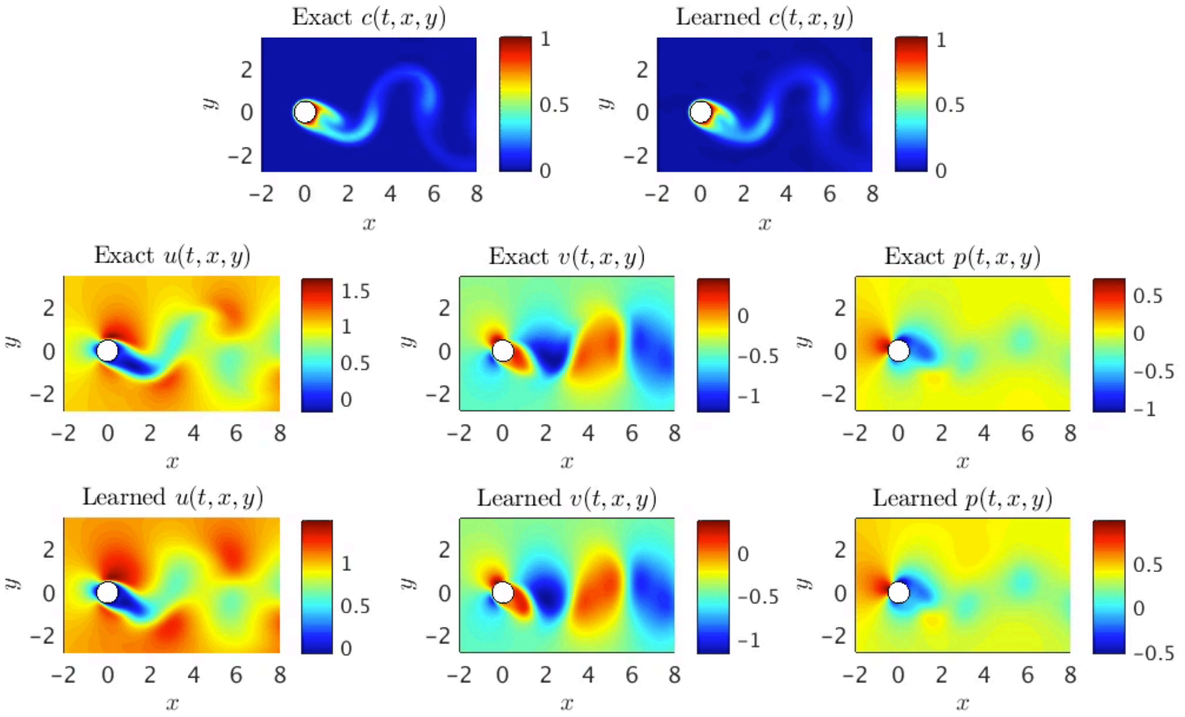
Raissi, Maziar, Alireza Yazdani, and George Em Karniadakis. "Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations." *Science* (2020).

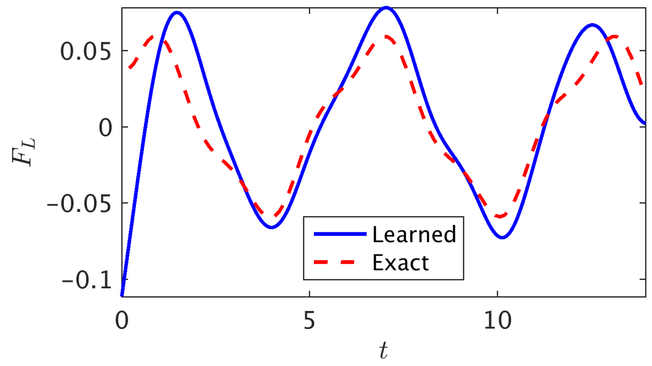
Deep Learning of Vortex Induced Vibrations

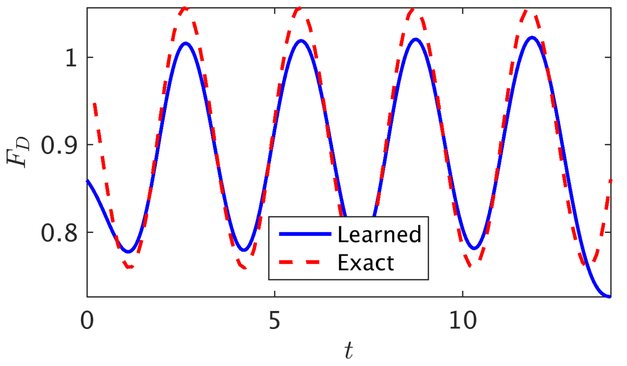


	Damping	Stiffness
Exact	$b = 0.084$	$k = 2.2020$
Learned	$b = 0.08600664$	$k = 2.2395933$
Error	2.39%	1.71%



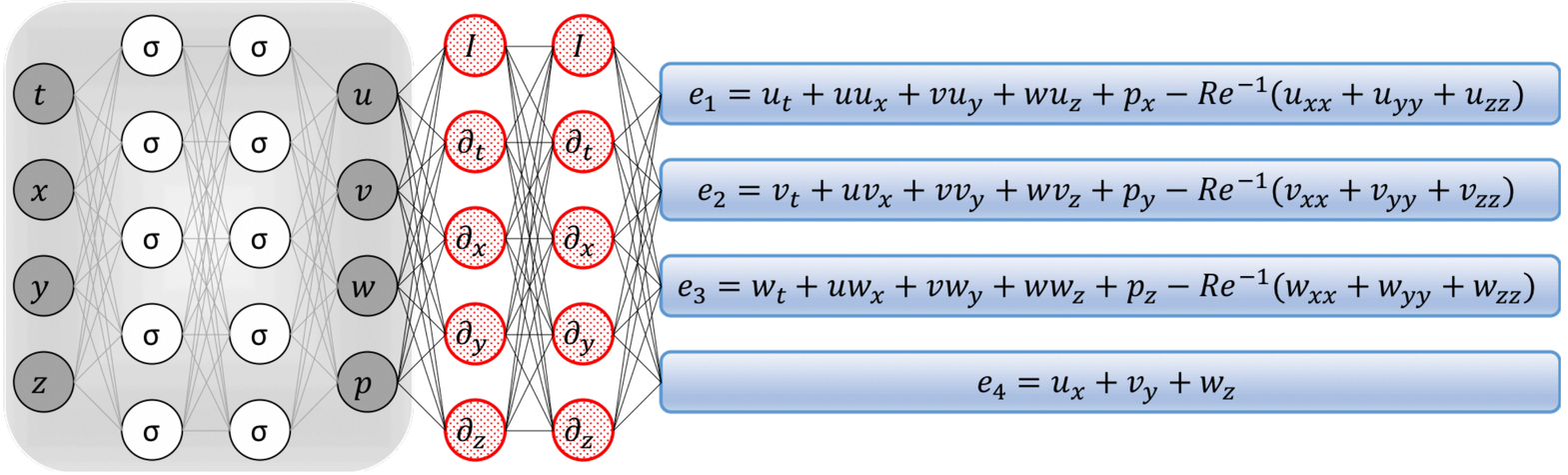






Raissi, Maziar, Zhicheng Wang, Michael S. Triantafyllou, and George Em Karniadakis. "Deep Learning of Vortex Induced Vibrations." *Journal of Fluid Mechanics* (2018).

Eulerian-Lagrangian Neural Networks

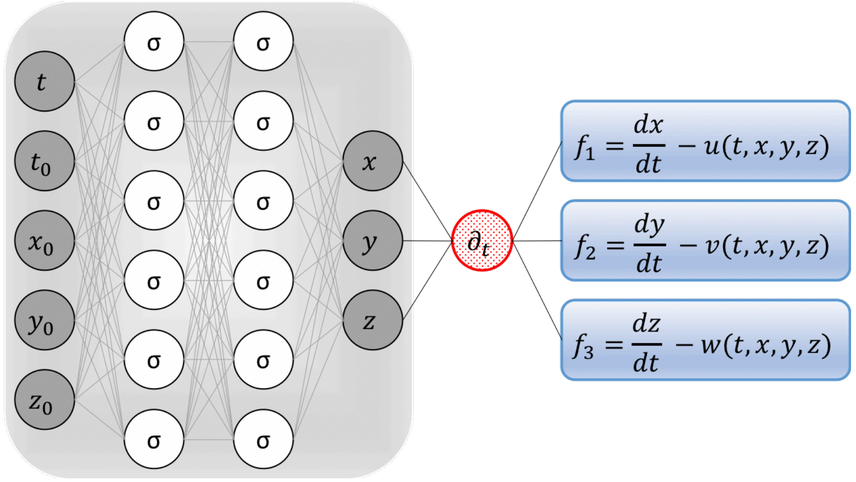


$$e_1 = u_t + uu_x + vu_y + wu_z + p_x - Re^{-1}(u_{xx} + u_{yy} + u_{zz})$$

$$e_2 = v_t + uv_x + vv_y + wv_z + p_y - Re^{-1}(v_{xx} + v_{yy} + v_{zz})$$

$$e_3 = w_t + uw_x + vw_y + ww_z + p_z - Re^{-1}(w_{xx} + w_{yy} + w_{zz})$$

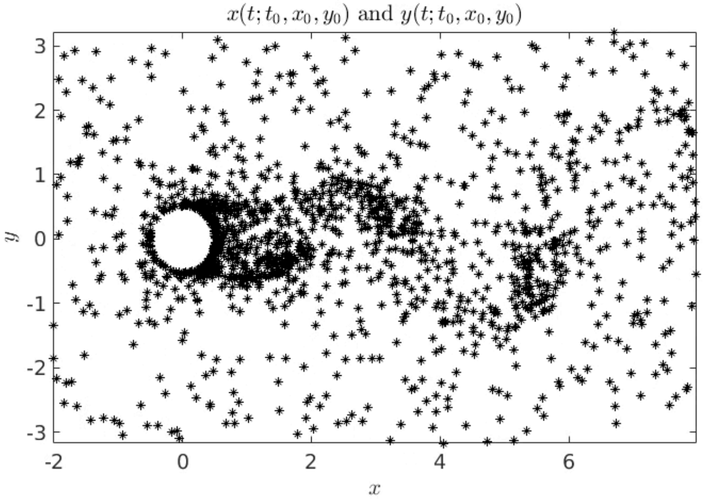
$$e_4 = u_x + v_y + w_z$$

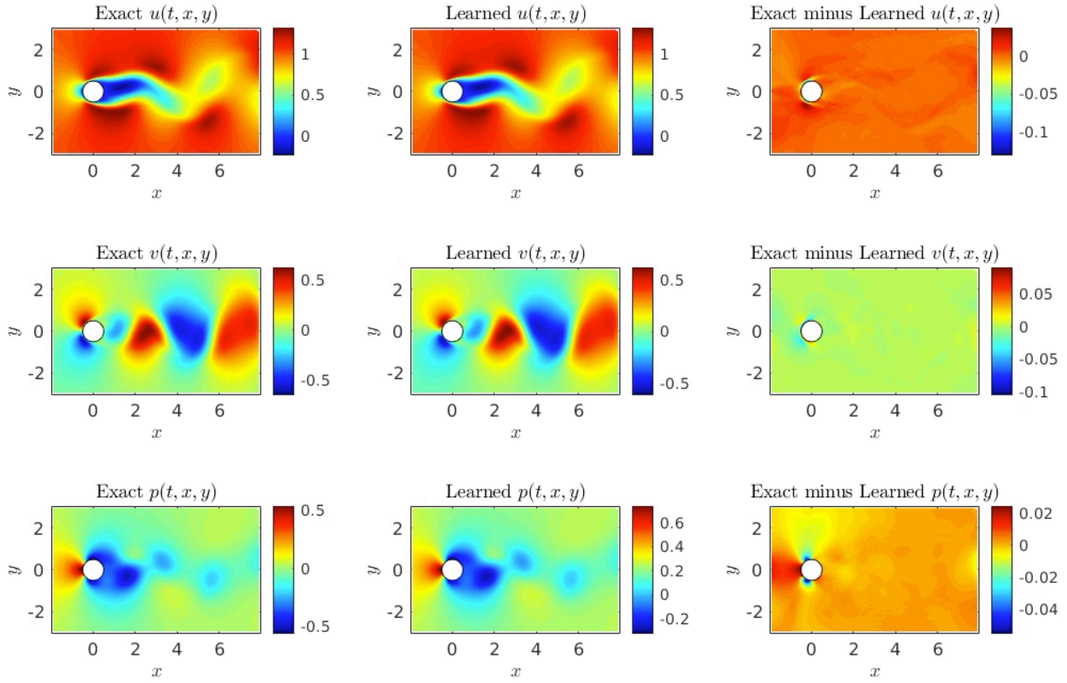


$$f_1 = \frac{dx}{dt} - u(t,x,y,z)$$

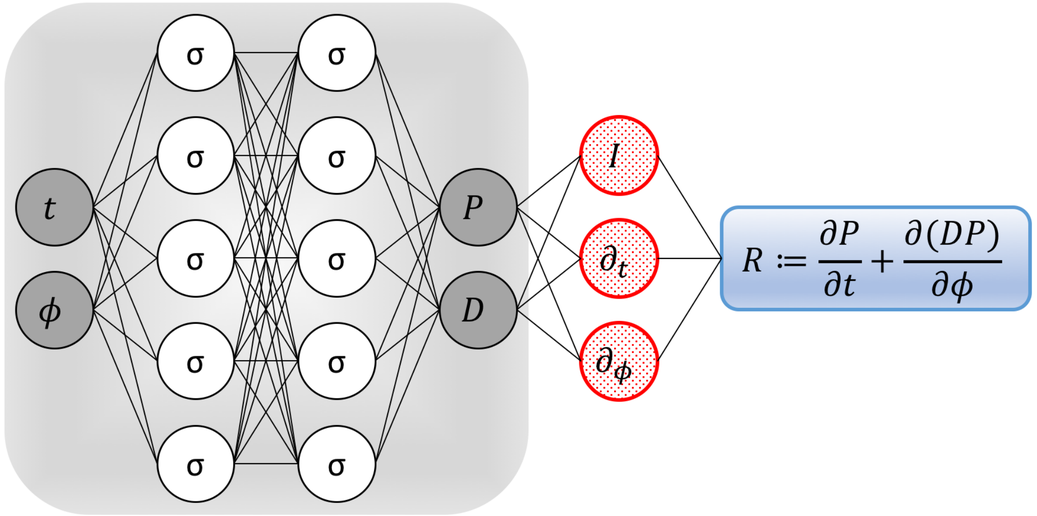
$$f_2 = \frac{dy}{dt} - v(t,x,y,z)$$

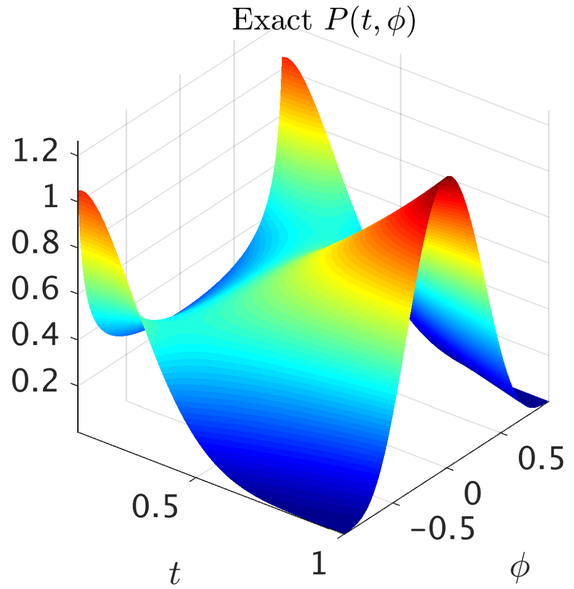
$$f_3 = \frac{dz}{dt} - w(t,x,y,z)$$

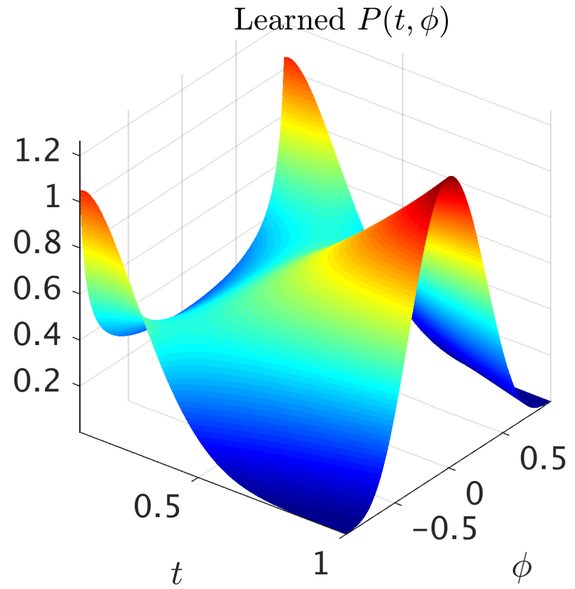




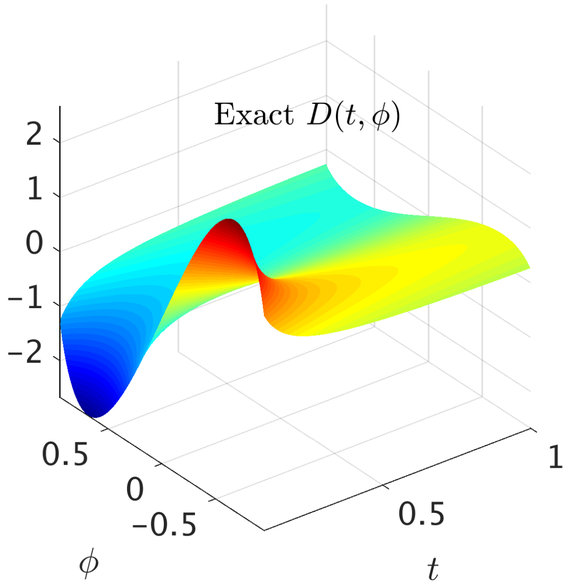
Deep Learning of Turbulent Scalar Mixing

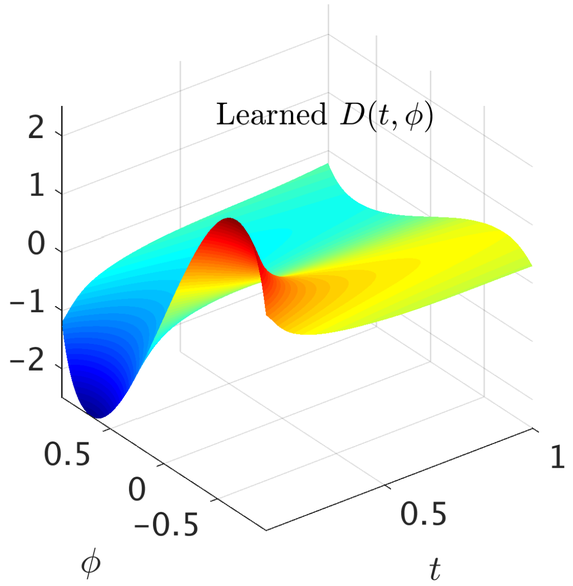






$$SSE = \sum_{n=1}^{N} |P(t^n, \phi^n) - P^n|^2 + \sum_{n=1}^{N} |R(t^n, \phi^n)|^2$$

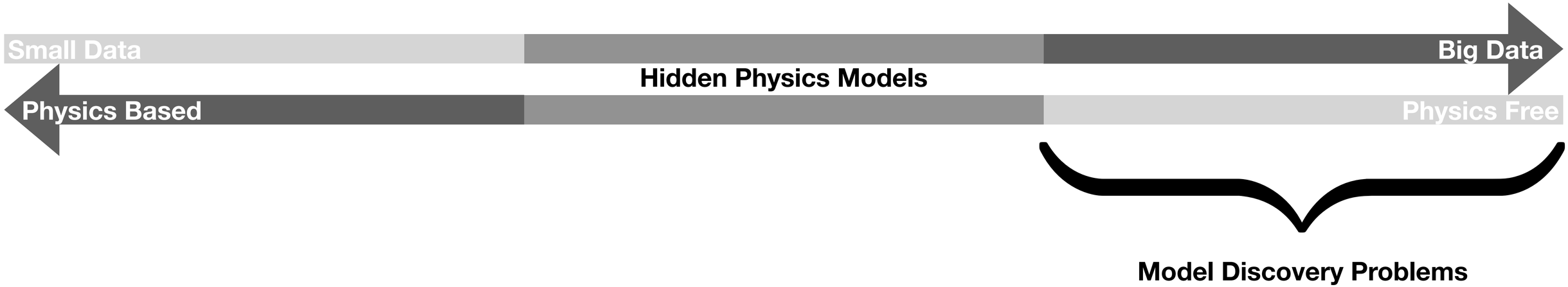




Raissi, Maziar, Hessam Babaee, and Peyman Givi. "Deep learning of turbulent scalar mixing." *Physical Review Fluids* 4.12 (2019): 124501.

Boulder

Model Discovery Problems

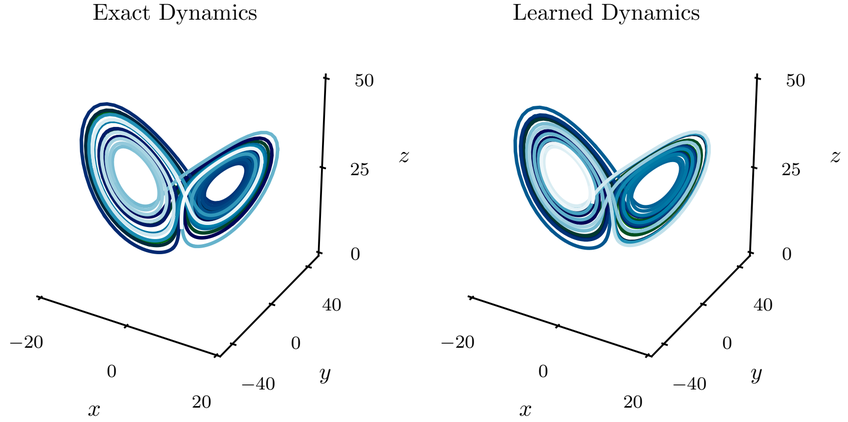


Multi-step Neural Networks

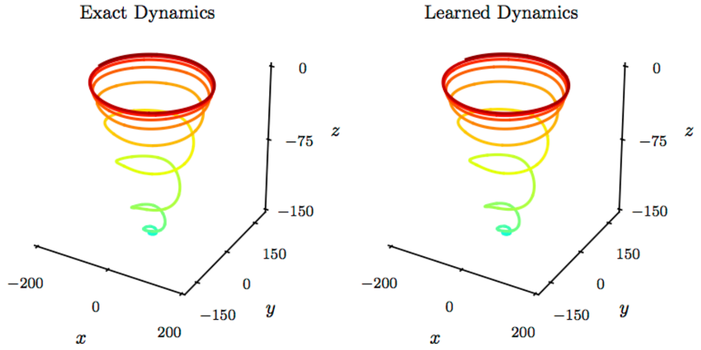
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$$

$$\mathbf{x}^{n+1} = \mathbf{x}^{n} + \frac{\Delta t}{2}\left(\mathbf{f}(\mathbf{x}^{n}) + \mathbf{f}(\mathbf{x}^{n+1})\right)$$

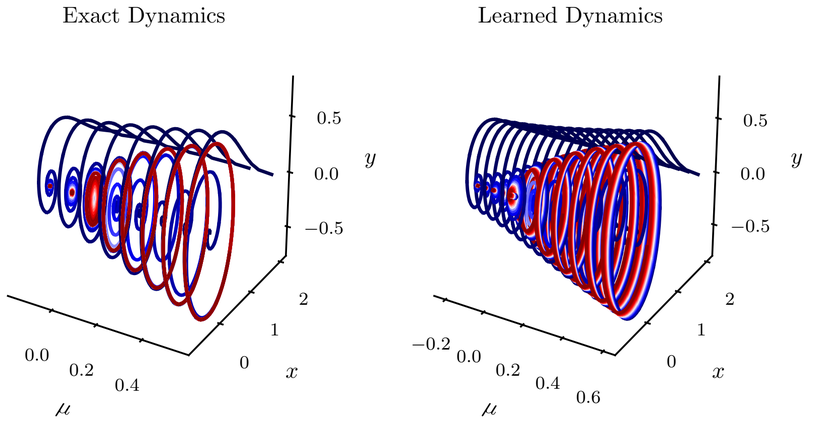
Lorenz System



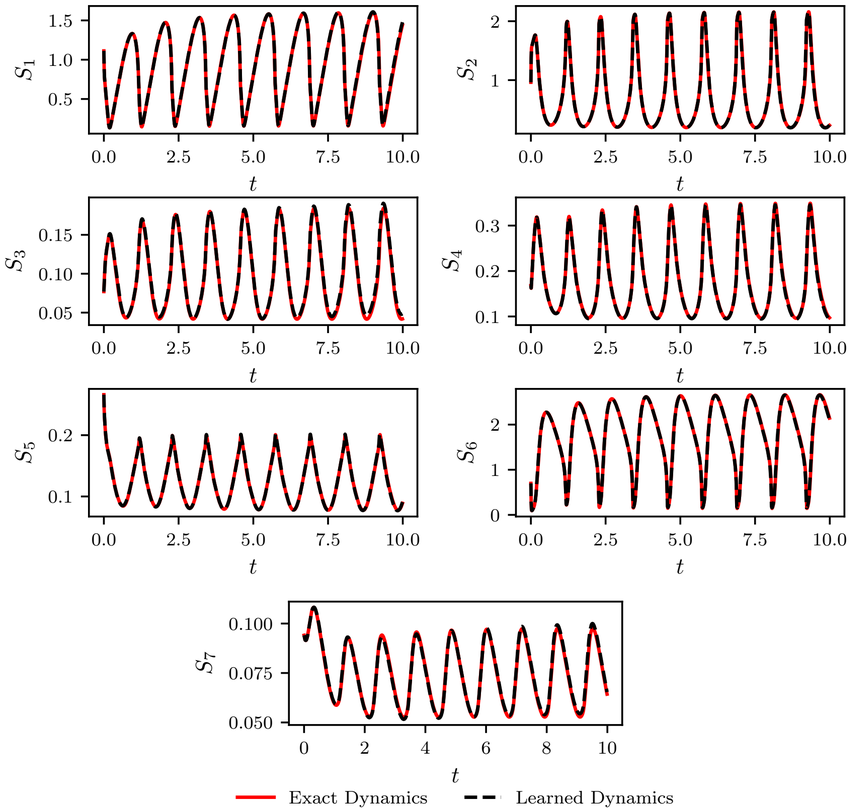
Navier-Stokes Equations



Hopf Bifurcation



Glycolytic Oscillator



Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Multistep neural networks for data-driven discovery of nonlinear dynamical systems." *arXiv preprint arXiv:1801.01236* (2018).

Deep Hidden Physics Models

$$u_t = \mathcal{N}(t, x, u, u_x, u_{xx}, \ldots) \qquad f = u_t - \mathcal{N}(t, x, u, u_x, u_{xx}, \ldots) \qquad \sum_{i=1}^{N} |u(t_i, x_i) - u_i|^2 + |f(t_i, x_i)|^2$$

Maziar Raissi (2018). Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations. *Journal of Machine Learning Research*, 19, 1-24.

Boulder

Hidden Physics Models

Thank you!

DARPA EQUiPS N66001-15-2-4055

MURI/ARO W911NF-15-1-0562

AFOSR FA9550-17-1-001

NIH U01HL116323

NSF DMS-1736088

527 followers

https://github.com/maziarraissi

NumericalGP

Numerical Gaussian Processes for Time-dependent and Non-linear Partial Differential Equations

MATLAB ☆ 35 28

HPM

Hidden physics models: Machine learning of nonlinear partial differential equations

MATLAB ☆ 66 47

ParametricGP

Parametric Gaussian Process Regression for Big Data

Python ☆ 29 14

DeepHPMs

Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations

Python ☆ 161 119

PINNs

Physics Informed Deep Learning: Data-driven Solutions and Discovery of Nonlinear Partial Differential Equations

Python ☆ 601 350

FBSNNs

Forward-Backward Stochastic Neural Networks: Deep Learning of High-dimensional Partial Differential Equations

Python ☆ 66 44