On surrogate modeling

Lecture notes with derivations

Joseph Morlier
ISAE-SUPAERO, Toulouse

November 2026

Abstract

These notes accompany the interactive article *On surrogate modeling* and its notebook `on_surrogate_modeling.ipynb`. They follow the same path — cost and break-even, design of experiments, response surfaces and honest validation, Gaussian processes, proper orthogonal decomposition, neural networks, and trust and design loops — and give the derivations the web page only states. Every number quoted is produced by the notebook. The sequence of topics follows the open course at `https://surrogatemodels.com/learn` [1]; text, examples and code are original.

Contents

1 What a surrogate is, and when it pays **2**

2 Sampling the design space **2**

3 Response surfaces and honest validation **3**
3.1 Least squares and the normal equations . . . 3
3.2 Bias–variance decomposition . . . 3
3.3 Validation and cross-validation . . . 3

4 Gaussian processes **4**
4.1 Prior, conditioning, posterior . . . 4
4.2 Hyperparameters and cost . . . 5

5 Proper orthogonal decomposition for fields **5**
5.1 POD as a truncated SVD . . . 5
5.2 POD + regression . . . 6

6 Neural-network surrogates **6**
6.1 Forward pass and backpropagation . . . 6
6.2 Optimizers . . . 7
6.3 Why normalize . . . 7

7 Trust, uncertainty and design loops **7**
7.1 Ensembles . . . 7
7.2 Split conformal prediction . . . 7
7.3 Choosing the next simulation . . . 8

8 Where this goes next **9**

1 What a surrogate is, and when it pays

Let $f : \mathcal{X} \subset \mathbb{R}^d \to \mathbb{R}^q$ be an expensive simulation code; x gathers design and operating parameters and $y = f(x)$ is a set of scalars (q small) or a discretized field ($q = M$ large). A *surrogate* is a model $\hat{y} \approx f$ fitted to a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{D}$ [2, 3].

Break-even. Let c_{sim} be the cost of one simulation, C_{train} the training cost and c_{inf} the cost of one prediction. After N queries,

$$C_{\text{solver}}(N) = N c_{\text{sim}}, \qquad C_{\text{surr}}(N) = D c_{\text{sim}} + C_{\text{train}} + N c_{\text{inf}}. \tag{1}$$

Both are affine in N; the surrogate is cheaper iff $N(c_{\text{sim}} - c_{\text{inf}}) > D c_{\text{sim}} + C_{\text{train}}$, i.e.

$$N > N^* = \frac{D c_{\text{sim}} + C_{\text{train}}}{c_{\text{sim}} - c_{\text{inf}}} \approx D + \frac{C_{\text{train}}}{c_{\text{sim}}} \quad (c_{\text{inf}} \ll c_{\text{sim}}). \tag{2}$$

With $D = 40$, $C_{\text{train}} = 5\, c_{\text{sim}}$, $c_{\text{inf}} = 10^{-4} c_{\text{sim}}$: $N^* = 45.0$ (Fig. 1). A surrogate therefore pays off only if it will be queried more often than it was trained — which is the case in optimization, uncertainty propagation and interactive design, and not for a one-off analysis.

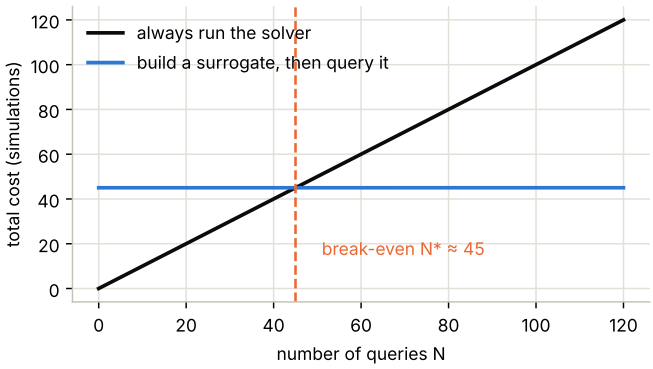


Figure 1: Cost of running the solver for every query versus building and querying a surrogate.

2 Sampling the design space

Computer experiments are (mostly) deterministic, so replication is useless and a design should *fill* the input domain [4]. We work in $[0,1]^d$ after an affine scaling of each input.

Grids. A full factorial with L levels costs L^d runs ($3^{10} = 59\,049$) and projects onto only L distinct values per input: if an input turns out inactive, most runs are duplicates.

Definition 1 (Latin hypercube [5])**.** *Let $\pi_1, \dots, \pi_d$ be independent uniform random permutations of $\{0, \dots, D-1\}$ and $U_{ij} \sim \mathcal{U}(0,1)$ i.i.d. The Latin hypercube design is $x_{ij} = (\pi_j(i) + U_{ij})/D$, $i = 1..D$, $j = 1..d$.*

Proposition 1. *(i) For every input j, each interval $[k/D, (k+1)/D)$ contains exactly one x_{ij}, so every input receives D distinct values. (ii) Each x_{ij} is marginally $\mathcal{U}(0,1)$; hence the LHS estimate $\bar{g} = \frac{1}{D}\sum_i g(x_i)$ of $\int g$ is unbiased, and for additive g its variance is never larger than with independent sampling.*

Proof. (i) π_j is a bijection, so the integer parts $\pi_j(i)$ take every value in $\{0..D-1\}$ once. (ii) $\pi_j(i)$ is uniform on $\{0..D-1\}$ and U_{ij} uniform on $[0,1)$, so x_{ij} is uniform on $[0,1)$; linearity of expectation gives unbiasedness. For $g(x) = \sum_j g_j(x_j)$ the estimator decomposes into d one-dimensional stratified-sampling estimators, each with variance at most that of plain Monte Carlo [5]. □

The pairing between inputs is random, so an LHS can still contain close points. The *maximin* criterion [6] chooses, among candidate designs, the one maximizing $d_{\min}(X) = \min_{i \neq k} \|x_i - x_k\|$. In the notebook (16 points in 2D, Fig. 2): grid $d_{\min} = 0.250$ with 4 levels per input; random 0.092; plain LHS 0.062; maximin LHS (best of 500) 0.157 with 16 levels. The centred L_2 discrepancy, a measure of departure from uniformity, is 0.0114, 0.0105, 0.0027 and 0.0030 respectively.

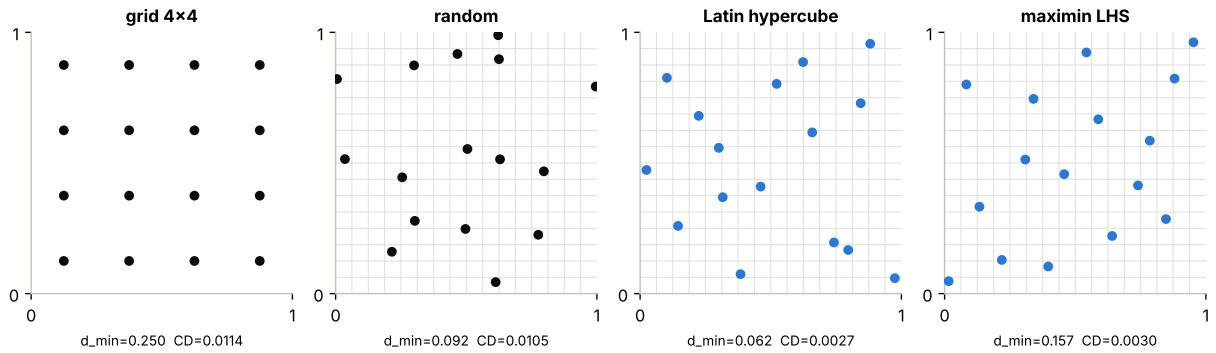


Figure 2: Four 16-point designs. Grey lines: LHS strata.

3 Response surfaces and honest validation

3.1 Least squares and the normal equations

With basis functions $\phi_0, \dots, \phi_p$ and $\Phi \in \mathbb{R}^{D \times (p+1)}$, $\Phi_{ik} = \phi_k(x_i)$, the least-squares coefficients minimize $J(\beta) = \|\Phi\beta - y\|^2$. Since $J(\beta) = \beta^\top \Phi^\top \Phi \beta - 2\beta^\top \Phi^\top y + y^\top y$,

$$\nabla J = 2\Phi^\top \Phi \beta - 2\Phi^\top y = 0 \iff \Phi^\top \Phi\, \hat{\beta} = \Phi^\top y. \tag{3}$$

J is convex; if Φ has full column rank, $\Phi^\top \Phi$ is positive definite and $\hat{\beta} = (\Phi^\top \Phi)^{-1} \Phi^\top y$ is the unique minimizer. Numerically one never forms $\Phi^\top \Phi$ (its condition number is the square of that of Φ): with a thin QR factorization $\Phi = QR$, $\hat{\beta} = R^{-1} Q^\top y$. Orthogonal bases (Legendre polynomials on $[-1, 1]$) keep Φ well conditioned.

The fitted values are $\hat{y} = Hy$ with the *hat matrix* $H = \Phi(\Phi^\top \Phi)^{-1} \Phi^\top$, an orthogonal projector of rank $p + 1$.

3.2 Bias–variance decomposition

Let $y = f(x) + \varepsilon$ with $\mathbb{E}\varepsilon = 0$, $\operatorname{Var} \varepsilon = \sigma_\varepsilon^2$, and let $\hat{y}$ depend on a random training set $\mathcal{D}$ independent of the new noise ε. At a fixed x, write $\bar{y}(x) = \mathbb{E}_{\mathcal{D}} \hat{y}(x)$.

Proposition 2. $\mathbb{E}\big[(y - \hat{y}(x))^2\big] = \big(f(x) - \bar{y}(x)\big)^2 + \operatorname{Var}_{\mathcal{D}} \hat{y}(x) + \sigma_\varepsilon^2$.

Proof. $y - \hat{y} = \varepsilon + (f - \bar{y}) + (\bar{y} - \hat{y})$. The three terms are respectively zero-mean noise, a constant, and a zero-mean function of $\mathcal{D}$; ε is independent of $\mathcal{D}$. Squaring and taking expectations, all cross terms vanish: $\mathbb{E}[\varepsilon(f - \bar{y})] = 0$, $\mathbb{E}[\varepsilon(\bar{y} - \hat{y})] = \mathbb{E}\varepsilon\, \mathbb{E}(\bar{y} - \hat{y}) = 0$, $\mathbb{E}[(f - \bar{y})(\bar{y} - \hat{y})] = (f - \bar{y})\, \mathbb{E}(\bar{y} - \hat{y}) = 0$. □

Low-degree fits have large bias; high-degree fits large variance. The training error cannot reveal the trade-off: it is non-increasing in p for nested bases, and zero once $p + 1 \geq D$.

3.3 Validation and cross-validation

Use disjoint *training* (fit), *validation* (choices) and *test* (final report) sets. With few runs, k-fold cross-validation partitions $\{1..D\}$ into folds $F_1..F_k$ and estimates the prediction error by

$$\mathrm{CV}_k = \frac{1}{D} \sum_{m=1}^{k} \sum_{i \in F_m} \big(y_i - \hat{y}^{(-m)}(x_i)\big)^2, \tag{4}$$

where $\hat{y}^{(-m)}$ is fitted without fold m [7].

Proposition 3 (Leave-one-out for least squares)**.** *For a linear least-squares fit, the leave-one-out residual is* $y_i - \hat{y}^{(-i)}(x_i) = \dfrac{y_i - \hat{y}_i}{1 - H_{ii}}$*, so* CV_D *costs a single fit.*

Proof. Let $A = \Phi^\top \Phi$ and $\phi_i = \Phi_{i,:}^\top$. Removing row i gives $A_{-i} = A - \phi_i \phi_i^\top$ and right-hand side $\Phi^\top y - \phi_i y_i$. By Sherman–Morrison, $A_{-i}^{-1} = A^{-1} + \frac{A^{-1}\phi_i \phi_i^\top A^{-1}}{1 - H_{ii}}$ with $H_{ii} = \phi_i^\top A^{-1} \phi_i$. Then $\hat{y}^{(-i)}(x_i) = \phi_i^\top A_{-i}^{-1}(\Phi^\top y - \phi_i y_i) = \hat{y}_i + \frac{H_{ii}\hat{y}_i}{1-H_{ii}} - H_{ii} y_i - \frac{H_{ii}^2 y_i}{1-H_{ii}} = \frac{\hat{y}_i - H_{ii} y_i}{1 - H_{ii}}$, and $y_i - \hat{y}^{(-i)}(x_i) = \frac{y_i - \hat{y}_i}{1 - H_{ii}}$. □

Split by run. When a run yields a field, its nodal values are strongly correlated; folds must contain whole runs, otherwise CV measures interpolation between already-seen neighbours.

Metrics. RMSE $= \sqrt{\frac{1}{n}\sum (y_i - \hat{y}_i)^2}$; relative L_2 error $\|y - \hat{y}\| / \|y\|$; $R^2 = 1 - \sum (y_i - \hat{y}_i)^2 / \sum (y_i - \bar{y})^2$; maximum absolute error.

Example. Forrester function $f(x) = (6x - 2)^2 \sin(12x - 4)$, 15 LHS points, noise $\sigma_\varepsilon = 1$ (Fig. 3). Degree 12 has training RMSE 0.22 and test RMSE 43.8. 5-fold CV selects degree 4 (test RMSE 1.40, relative L_2 31%, $R^2 = 0.90$, max. error 8.3); the best degree in hindsight is 7.

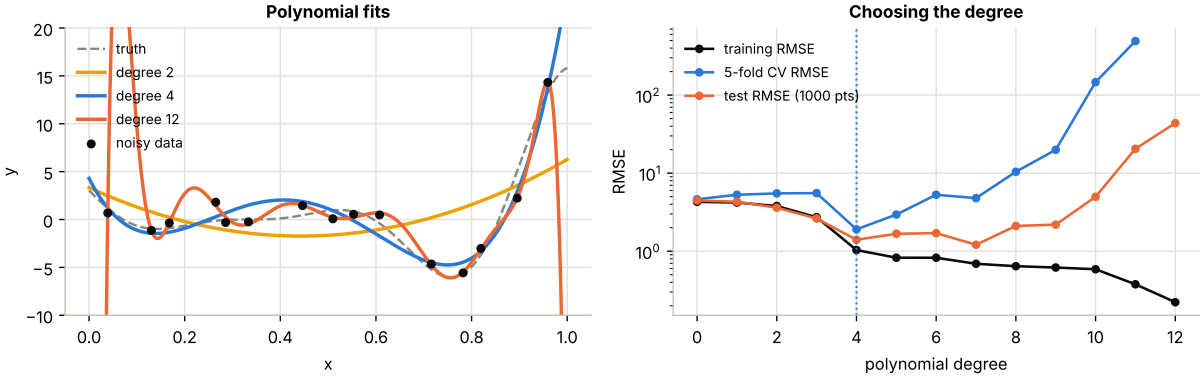


Figure 3: Polynomial fits and error versus degree: training, 5-fold CV and test (1000 points).

4 Gaussian processes

4.1 Prior, conditioning, posterior

A GP prior $f \sim \mathcal{GP}(0, k)$ states that any finite vector of function values is jointly Gaussian with covariance given by the kernel, e.g. the squared exponential $k(x, x') = \sigma_f^2 \exp(-\sum_j (x_j - x'_j)^2 / 2\ell_j^2)$. With observations $y = f(X) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma_n^2 I)$, and a test input x_*,

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K + \sigma_n^2 I & k_* \\ k_*^\top & k_{**} \end{bmatrix}\right), \quad K_{ij} = k(x_i, x_j),\ (k_*)_i = k(x_i, x_*),\ k_{**} = k(x_*, x_*). \tag{5}$$

Proposition 4 (Gaussian conditioning)**.** *If* $\begin{bmatrix} a \\ b \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} A & C^\top \\ C & B \end{bmatrix}\right)$ *with* A *invertible, then* $b \mid a \sim \mathcal{N}(CA^{-1}a,\ B - CA^{-1}C^\top)$.

Proof. Let $e = b - CA^{-1}a$. Then $\operatorname{Cov}(e, a) = C - CA^{-1}A = 0$; since (a, e) is a linear image of a Gaussian vector it is jointly Gaussian, so e and a are independent. $\operatorname{Cov}(e) = B - 2CA^{-1}C^\top + CA^{-1}AA^{-1}C^\top = B - CA^{-1}C^\top$. Conditioning on a leaves e unchanged and $b = CA^{-1}a + e$. □

With $a = y$, $b = f_*$, $A = K + \sigma_n^2 I$, $C = k_*^\top$, $B = k_{**}$:

$$\bar{f}_* = k_*^\top (K + \sigma_n^2 I)^{-1} y, \qquad \operatorname{Var} f_* = k_{**} - k_*^\top (K + \sigma_n^2 I)^{-1} k_*. \tag{6}$$

4.2 Hyperparameters and cost

$\theta = (\sigma_f, \ell, \sigma_n)$ maximizes the log marginal likelihood

$$\log p(y \mid X, \theta) = -\tfrac{1}{2} y^\top K_y^{-1} y - \tfrac{1}{2} \log\det K_y - \tfrac{D}{2} \log 2\pi, \qquad K_y = K + \sigma_n^2 I, \tag{7}$$

a data-fit term plus a complexity penalty [8]. All quantities follow from one Cholesky factorization $K_y = LL^\top$: $O(D^3)$ time, $O(D^2)$ memory.

On the 15 noisy Forrester points the GP reaches a test RMSE of 0.62 (cross-validated polynomial: 1.40) and its $\pm 2\sigma$ band contains the true function on all of $[0, 1]$ (Fig. 4). The companion article [9] develops kernels and Bayesian optimization interactively.

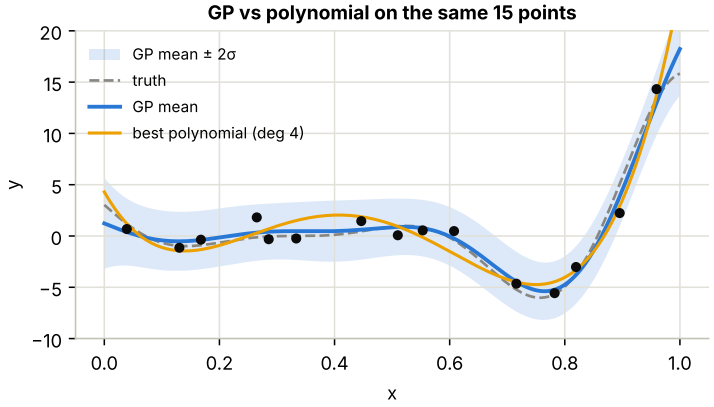


Figure 4: GP posterior mean and $\pm 2\sigma$ band versus the cross-validated polynomial.

5 Proper orthogonal decomposition for fields

5.1 POD as a truncated SVD

Let $u_1, \ldots, u_N \in \mathbb{R}^M$ be snapshots, $\bar{u}$ their mean and $Y = [u_1 - \bar{u}, \ldots, u_N - \bar{u}] \in \mathbb{R}^{M \times N}$. With the SVD $Y = U\Sigma V^\top = \sum_k \sigma_k \phi_k v_k^\top$ ($\sigma_1 \geq \sigma_2 \geq \ldots$), the POD modes are the columns ϕ_k of U [10–12].

Proposition 5 (Optimality — Eckart–Young)**.** *Among all orthonormal $\Psi \in \mathbb{R}^{M \times K}$, the projection error on the snapshots $E(\Psi) = \sum_{j=1}^{N} \|(I - \Psi\Psi^\top)(u_j - \bar{u})\|^2$ is minimized by $\Psi = [\phi_1 .. \phi_K]$, and* $\min E = \sum_{k>K} \sigma_k^2$.

Proof. $E(\Psi) = \|Y\|_F^2 - \|\Psi^\top Y\|_F^2 = \sum_k \sigma_k^2 - \operatorname{tr}(\Psi^\top Y Y^\top \Psi)$, so we must maximize $\operatorname{tr}(\Psi^\top Y Y^\top \Psi)$ over orthonormal Ψ. Writing $\Psi^\top U = W$, $\operatorname{tr}(\Psi^\top Y Y^\top \Psi) = \sum_k \sigma_k^2 w_k$ with $w_k = \|W_{:,k}\|^2 \in [0, 1]$ and $\sum_k w_k \leq K$ (the columns of Ψ are orthonormal). A linear function of w over this set is maximized by putting weight 1 on the K largest σ_k^2, which $\Psi = [\phi_1 .. \phi_K]$ achieves (Ky Fan's principle). □

The *energy* retained by K modes is $\sum_{k \leq K} \sigma_k^2 / \sum_k \sigma_k^2$.

Method of snapshots. When $M \gg N$, solve the $N \times N$ eigenproblem $Y^\top Y v_k = \sigma_k^2 v_k$ and set $\phi_k = Y v_k / \sigma_k$; indeed $YY^\top \phi_k = Y(Y^\top Y v_k)/\sigma_k = \sigma_k^2 \phi_k$ and $\|\phi_k\|^2 = v_k^\top Y^\top Y v_k / \sigma_k^2 = 1$. On non-uniform meshes, use the weighted inner product $\langle u, w \rangle = \sum_i a_i u_i w_i$ (a_i the cell areas): apply the SVD to $\mathrm{diag}(\sqrt{a})Y$ and divide the modes by $\sqrt{a}$.

5.2 POD + regression

Approximate $u(\mu) \approx \bar{u} + \sum_{k \le K} a_k(\mu)\phi_k$ and learn each map $\mu \mapsto a_k(\mu)$ (here one GP per coefficient) from the training coefficients $a_k(\mu_j) = \phi_k^\top (u_j - \bar{u})$. The error splits into a projection part (fixed by K) and a regression part.

Example. Two families on $[0, 1]$, $M = 200$, 40 snapshots (Fig. 5): a square pulse on $[0.3, 0.6]$ under the heat equation, $u(x,t) = \sum_{k \ge 1} b_k \sin(k\pi x) e^{-k^2\pi^2 t}$, $t \in [10^{-3}, 3 \cdot 10^{-2}]$; and a Gaussian bump of width 0.03 centred at $c \in [0.15, 0.85]$. Energy 99 % needs 2 modes for diffusion and 14 for transport. POD(6)+GP predicts unseen diffusion fields to 0.01% relative error (mean field alone: 24%); POD(10)+GP gives 22.4% for transport, equal to the projection error: the linear basis, not the regression, is the bottleneck. Transported features have a slowly decaying Kolmogorov n-width.

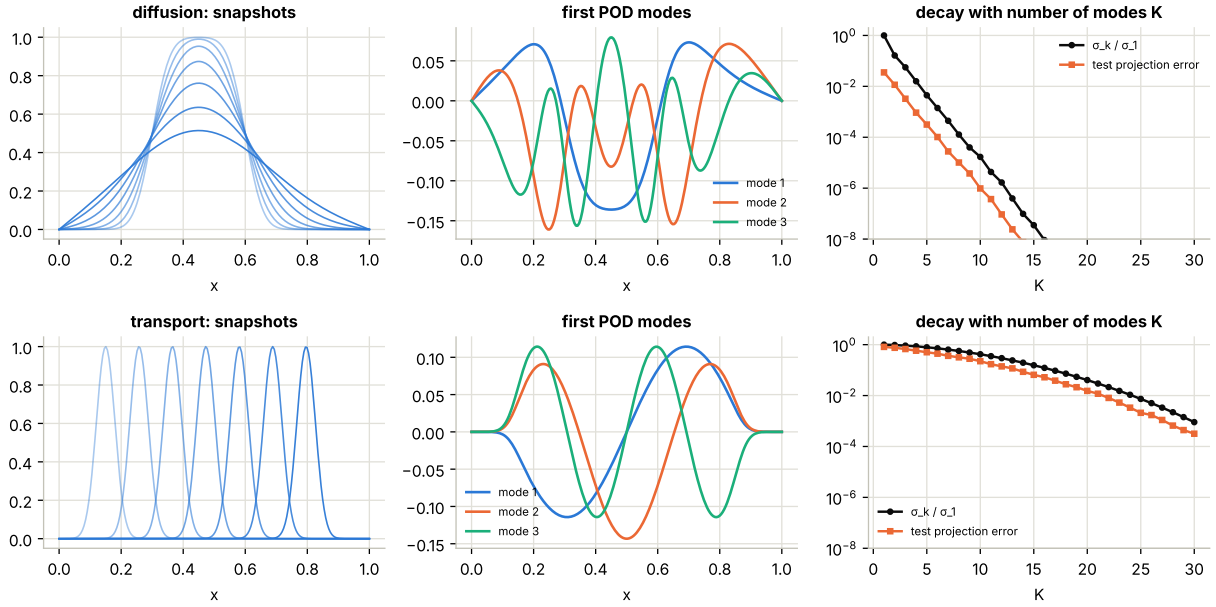


Figure 5: Snapshots, first modes, singular values and test projection error for diffusion (top) and transport (bottom).

6 Neural-network surrogates

6.1 Forward pass and backpropagation

An MLP with L layers computes $h^{(0)} = x$, $z^{(l)} = W^{(l)} h^{(l-1)} + b^{(l)}$, $h^{(l)} = \tanh z^{(l)}$ for $l < L$ and $\hat{y} = z^{(L)}$. For the loss $\mathcal{L} = \frac{1}{n}\sum_i (\hat{y}_i - y_i)^2$, define $\delta^{(l)} = \partial \mathcal{L} / \partial z^{(l)}$ (per sample, summed over the batch). The chain rule [13] gives

$$\delta^{(L)} = \tfrac{2}{n}(\hat{y} - y), \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)} h^{(l-1)\top}, \qquad \frac{\partial \mathcal{L}}{\partial b^{(l)}} = \delta^{(l)}, \tag{9}$$

$$\delta^{(l-1)} = (W^{(l)\top} \delta^{(l)}) \odot (1 - (h^{(l-1)})^2), \tag{10}$$

since $z^{(l)}$ depends on $W^{(l)}$ linearly through $h^{(l-1)}$, on $h^{(l-1)}$ through $W^{(l)}$, and $\tanh' = 1 - \tanh^2$. The notebook's implementation agrees with central finite differences to a relative error of 2.7×10^{-9}.

6.2 Optimizers

Gradient descent: $w \leftarrow w - \eta \nabla \mathcal{L}$. Adam [14]: with $g_t = \nabla \mathcal{L}$, $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$, $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$, $w \leftarrow w - \eta\, \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ with bias corrections $\hat{m}_t = m_t/(1-\beta_1^t)$, $\hat{v}_t = v_t/(1-\beta_2^t)$. The step is roughly $\eta\, \mathrm{sign}(g)$: invariant to a rescaling of each gradient component.

6.3 Why normalize

From the backpropagation formula, $\partial \mathcal{L}/\partial W^{(1)} = \delta^{(1)} x^\top$ scales linearly with the input. Feeding $x_{\text{phys}} = 100x$ multiplies first-layer gradients by ~ 100 relative to the others, so one learning rate cannot suit all layers, and pre-activations $W^{(1)} x_{\text{phys}}$ of order 100 saturate the tanh units ($1 - \tanh^2 \approx 0$, so almost no gradient flows back through them). Over 8 seeds (1-32-32-1 network, 11 training points), gradient descent gives a median test RMSE of 4.48 on raw data against 0.82 standardized; Adam, being scale invariant per parameter, gives 0.97 and 1.23 (Fig. 6). Standardize inputs and outputs with training statistics only.

Capacity and early stopping. The network has 1153 weights for 11 points: it can interpolate the noise. Keep the weights at the minimum of the validation error (early stopping) or add weight decay. On this small smooth problem the GP (test RMSE 0.62) beats the network (0.75). Networks learn low frequencies first (spectral bias); Fourier features of the inputs help with sharp fields [15].

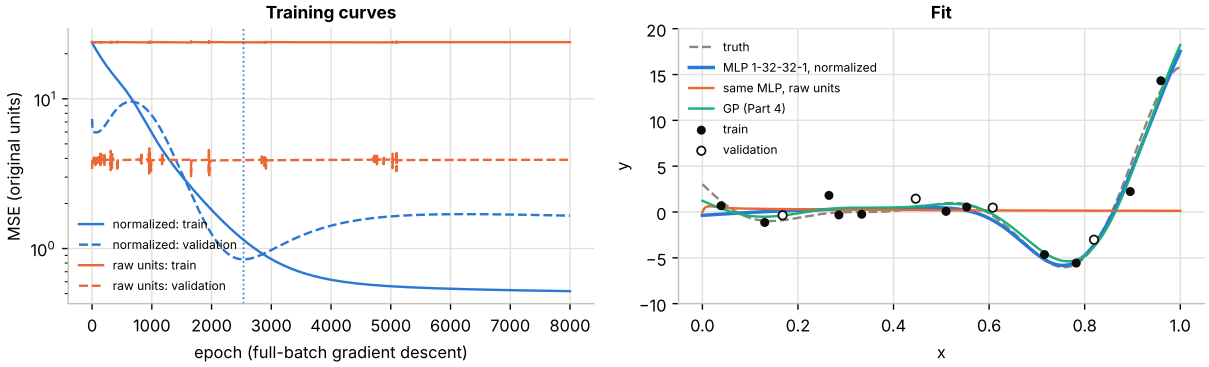


Figure 6: Gradient-descent learning curves on raw versus standardized data, and the early-stopped fit.

7 Trust, uncertainty and design loops

7.1 Ensembles

Train E networks from independent initializations; use the mean as prediction and the spread as an uncertainty signal [16]. In the notebook the members' standard deviation averages 0.38 inside $[0, 1]$ and 1.10 just outside, yet all members and the GP band miss the true downturn beyond $x = 1$ (Fig. 7): agreement is not correctness, and extrapolation is not certified by any of these tools.

7.2 Split conformal prediction

Fit $\hat{y}$ on a training set. On a calibration set $\{(x_i, y_i)\}_{i=1}^n$ disjoint from it, compute scores $s_i = |y_i - \hat{y}(x_i)|$ and let $\hat{q} = s_{(\lceil (n+1)(1-\alpha) \rceil)}$ (the order statistic; $\hat{q} = \infty$ if the rank exceeds n) [17].

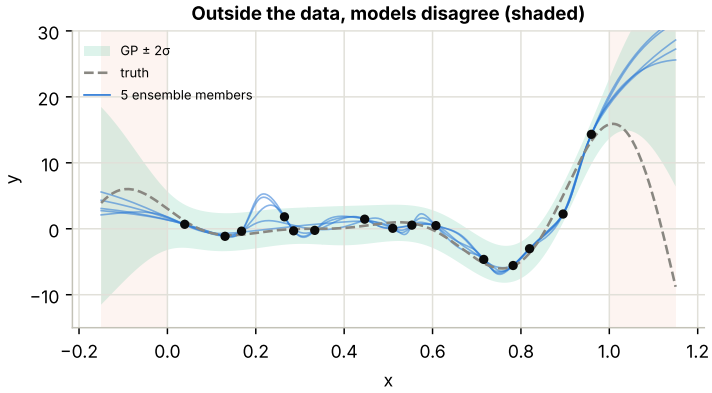


Figure 7: Five networks and a GP band; shaded regions lie outside the training data.

Proposition 6 (Marginal coverage)**.** *If* $(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1})$ *are exchangeable, then* $\mathbb{P}(|y_{n+1} - \hat{y}(x_{n+1})| \le \hat{q}) \ge 1 - \alpha$.

Proof. $\hat{y}$ is fixed given the training set, so the scores $s_1, \ldots, s_{n+1}$ are exchangeable. Assuming no ties, the rank of s_{n+1} among them is uniform on $\{1, \ldots, n+1\}$. The event $s_{n+1} \le \hat{q}$ holds whenever s_{n+1} is among the $\lceil (n+1)(1-\alpha) \rceil$ smallest of the $n+1$ scores, which has probability $\lceil (n+1)(1-\alpha) \rceil / (n+1) \ge 1 - \alpha$. (With ties the inequality still holds.) □

Using $s_i = |y_i - \hat{y}(x_i)| / \hat{\sigma}(x_i)$ gives adaptive intervals $\hat{y} \pm \hat{q}\,\hat{\sigma}(x)$ with the same guarantee. Example (Fig. 8): a GP on 30 noisy points, $n = 200$, $\alpha = 0.1$: coverage 89.6% on 5000 test points and 89.9% on average over 300 recalibrations, whereas the GP's own Gaussian 90 % intervals cover 82.6%. The guarantee is marginal and requires calibration and use to come from the same distribution.

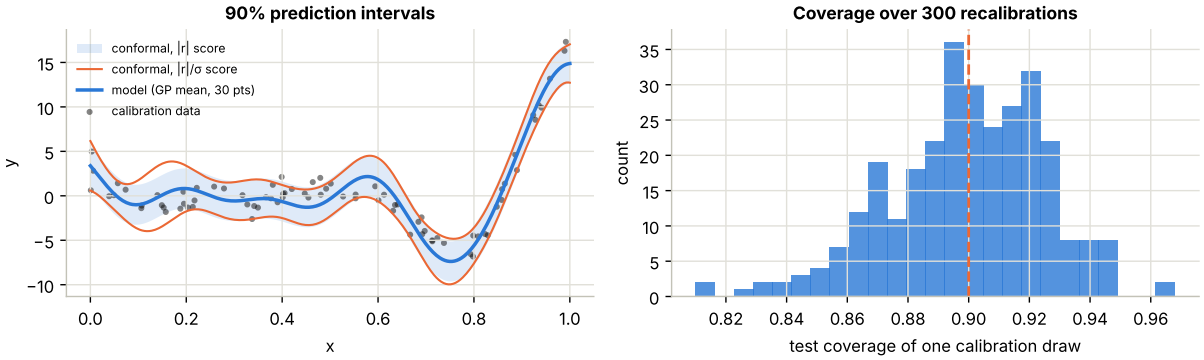


Figure 8: Split-conformal bands and coverage over 300 calibration draws.

7.3 Choosing the next simulation

Uncertainty sampling adds $x^+ = \arg\max \sigma(x)$. *Expected improvement* [18] adds $x^+ = \arg\max EI(x)$ with, for $y(x) \sim \mathcal{N}(\mu, \sigma^2)$ and $I = \max(y_{\min} - y, 0)$,

$$EI = \int_{-\infty}^{y_{\min}} (y_{\min} - y)\, \frac{1}{\sigma} \phi\Big(\frac{y - \mu}{\sigma}\Big) dy = (y_{\min} - \mu)\Phi(z) + \sigma\phi(z), \quad z = \frac{y_{\min} - \mu}{\sigma}, \tag{11}$$

obtained by substituting $y = \mu + \sigma t$, splitting $(y_{\min} - \mu - \sigma t)$, and using $\int_{-\infty}^{z} t\phi(t)dt = -\phi(z)$. Over 20 initial designs (3 points, 8 additions; Fig. 9), uncertainty sampling reaches a median global RMSE of 0.10 (EI 0.38, random 0.46), while EI finds the true minimum and uncertainty sampling stays 0.36 above it. See [19] for Bayesian optimization.

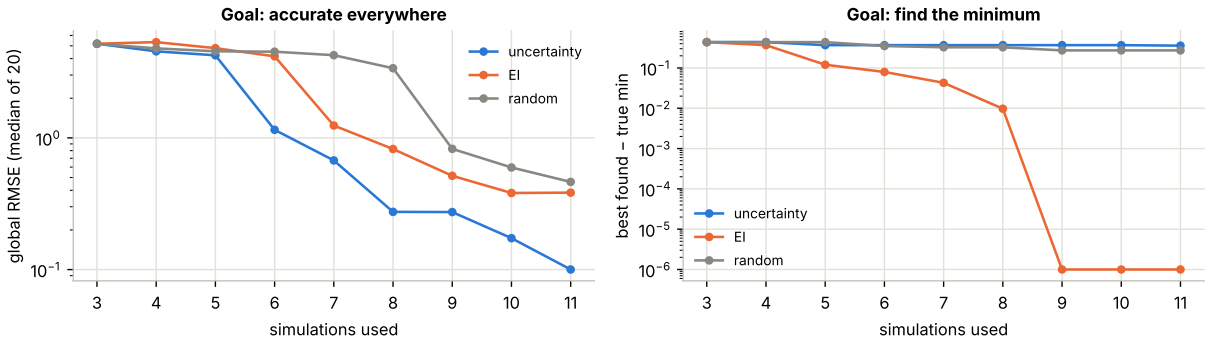


Figure 9: Active learning: global accuracy (left) versus finding the minimum (right).

Also in the design loop. Global sensitivity analysis by Sobol indices [20] computed on the cheap surrogate; multi-fidelity models that learn the discrepancy between cheap and expensive codes, e.g. $f_{\text{hi}}(x) = \rho f_{\text{lo}}(x) + \delta(x)$ with GP priors [21, 22]; and, always, a re-simulation of the design the surrogate proposes.

8 Where this goes next

For large meshes and varying geometry: U-Nets on grids [23]; graph networks on meshes (MeshGraphNets [24]); neural operators — DeepONet [25], $\mathcal{G}(u)(y) \approx \sum_{k=1}^{p} b_k(u)t_k(y)$, a nonlinear analogue of POD's coefficients times modes, and the Fourier neural operator [26], $v_{l+1} = \sigma(Wv_l + \mathcal{F}^{-1}(R \cdot \mathcal{F}v_l))$; transformers on mesh tokens [27] with linear-cost attention (Transolver [28]); physics-informed losses [29]; public benchmarks such as AirfRANS [30]; and pretrained physics models fine-tuned on a few runs. Tools: SMT [31].

Checklist for reading a surrogate paper. (1) Is the test split by run or geometry, and does it include extrapolation? (2) Which metrics, in which units, on which regions? (3) Is there a strong, equally tuned classical baseline (GP, POD + regression)? (4) Are there ablations? (5) What did data generation and training cost, compared with the break-even of Section 1? (6) Were predictions checked against physics (conservation, integrated loads, re-simulation)?

References

[1] surrogatemodels.com. Surrogate models: Physics AI for engineers — learn. Online course, 2026. URL https://surrogatemodels.com/learn.

[2] Alexander I. J. Forrester, András Sóbester, and Andy J. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley & Sons, 2008. doi: 10.1002/9780470770801.

[3] Nestor V. Queipo, Raphael T. Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P. Kevin Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41(1):1–28, 2005. doi: 10.1016/j.paerosci.2005.02.001.

[4] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989. doi: 10.1214/ss/1177012413.

[5] Michael D. McKay, Richard J. Beckman, and William J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979. doi: 10.2307/1268522.

[6] Max D. Morris and Toby J. Mitchell. Exploratory designs for computational experiments. *Journal of Statistical Planning and Inference*, 43(3):381–402, 1995. doi: 10.1016/0378-3758(94)00035-T.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009. doi: 10.1007/978-0-387-84858-7.

[8] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL http://gaussianprocess.org/gpml/.

[9] Joseph Morlier. A numerical tour of Gaussian processes and Bayesian optimization. Interactive article, MID2 group, ISAE-SUPAERO, 2026. URL https://mid2supaero.github.io/gp-bo/.

[10] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. part i: Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, 1987. doi: 10.1090/qam/910462.

[11] Gal Berkooz, Philip Holmes, and John L. Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25:539–575, 1993. doi: 10.1146/annurev.fl.25.010193.002543.

[12] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015. doi: 10.1137/130932715.

[13] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. doi: 10.1038/323533a0.

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

[15] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. URL https://arxiv.org/abs/2006.10739.

[16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. URL https://arxiv.org/abs/1612.01474.

[17] Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023. doi: 10.1561/2200000101.

[18] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998. doi: 10.1023/A:1008306431147.

[19] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2023. URL https://bayesoptbook.com.

[20] Ilya M. Sobol'. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55(1–3):271–280, 2001. doi: 10.1016/S0378-4754(00)00270-6.

[21] Marc C. Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000. doi: 10.1093/biomet/87.1.1.

[22] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018. doi: 10.1137/16M1082469.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015. doi: 10.1007/978-3-319-24574-4_28.

[24] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03409.

[25] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021. doi: 10.1038/s42256-021-00302-5.

[26] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.08895.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

[28] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2402.02366.

[29] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.

[30] Florent Bonnet, Jocelyn Ahmed Mazari, Paola Cinnella, and Patrick Gallinari. AirfRANS: High fidelity computational fluid dynamics dataset for approximating Reynolds-Averaged Navier-Stokes solutions. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022. URL https://arxiv.org/abs/2212.07564.

[31] Mohamed Amine Bouhlel, John T. Hwang, Nathalie Bartoli, Rémi Lafage, Joseph Morlier, and Joaquim R. R. A. Martins. A python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 135:102662, 2019. doi: 10.1016/j.advengsoft.2019.03.005.