K-Splanifold Interpolation: Structured Linear-Time Parametric Manifolds via Spine–Deviation Factorization

Chase Adams
Taurion Corporation
chase.adams@gmail.com
github.com/curvedinf djangoist.com x.com/curvedinf

Abstract

We introduce *K-Splanifold* interpolation, a structured parametric manifold primitive mapping an input coordinate $r \in [0,1]^K$ to an embedding space $\mathbb{R}^N$ with $\mathcal{O}(KN)$ evaluation and $\mathcal{O}(KN)$ control storage. The construction factorizes the parameter space into a scalar *spine coordinate* t and a zero-sum *deviation* vector δ, then evaluates a cubic Hermite spine curve plus a Hermite-interpolated transverse displacement field that is linear in δ for fixed t. Because both the spine and the per-parameter basis trajectories are cubic Hermite splines, the model class contains cubic polynomial trajectories and (via chaining) standard piecewise-cubic splines along the spine direction. We use a smooth, globally defined barycentric regularization for tangent blending near $\Sigma = \mathbf{1}^\top u = 0$ and provide bounds relating the regularized weights to unregularized normalization. An optional radial deviation warp yields bounded transverse magnitude under extrapolation and admits a closed-form inverse on the warped deviation. Experiments include control-field visualizations, stability plots, runtime baselines against an RBF interpolator and a coordinate-MLP, and fixed-budget curve fitting on standard 3D parametric curves.

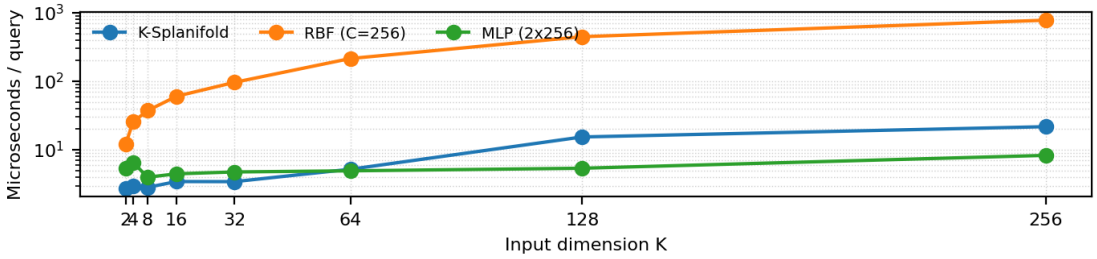
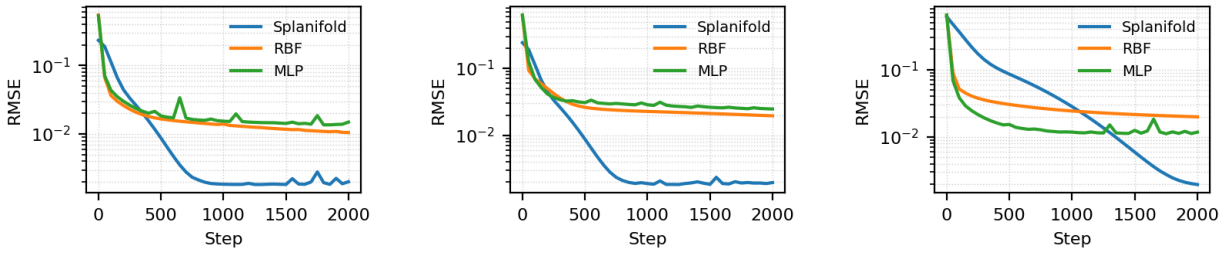


Figure 1: Runtime scaling vs. input dimension K (N=256), comparing K-Splanifold evaluation to RBF and coordinate-MLP baselines (log scale).



(a) Circle (b) Helix (c) Trefoil

Figure 2: Fixed-budget curve fitting on standard 3D parametric curves (test RMSE vs. iterations, log scale).

1 Introduction

High-dimensional parameterizations appear in geometric modeling, deformation rigs, reduced simulation controls, and learned latent spaces. A common operation is evaluating an embedding map $\Phi : [0,1]^K \to \mathbb{R}^N$ quickly and predictably, often under tight latency budgets (interactive editing, realtime playback, differentiable optimization).

Dense tensor-product spline volumes provide smooth interpolation and local support, but their control lattices scale exponentially with K [1, 2, 3]. Modern alternatives avoid dense grids in different ways: sparse grids (Smolyak constructions) prune interaction terms [8, 9]; radial basis functions (RBFs) interpolate scattered controls [10, 11]; and coordinate MLPs (neural fields) learn a compact parametric map [14, 15, 20, 19]. Neural fields can be highly expressive, but often rely on architectural choices or input encodings to represent high-frequency structure efficiently [18]. Each approach can be effective, but they introduce practical costs (basis management, global solves, or training and hyperparameters) that can be undesirable in author-driven pipelines.

This paper presents K-Splanifolds as a *structured* primitive aimed at parameter spaces that admit a dominant "progression" plus per-axis deviations, in the spirit of representing variation around a one-dimensional curve as in principal curves [13]. The central idea is to split the effective coordinate u into: (i) a scalar coordinate t that drives a global *spine* curve $\Psi(t)$, and (ii) a zero-sum deviation δ that drives a transverse displacement $\Delta(t, \delta)$. For fixed t, the displacement is linear in δ, yielding an interpretable control structure and linear-in-K evaluation.

Contributions. We contribute: (i) a concise spine–deviation factorization for $[0,1]^K$ controls; (ii) a linear-time evaluation rule combining Hermite spines with Hermite-interpolated transverse basis trajectories; (iii) a smooth tangent-weight regularization with an explicit bias bound; (iv) an optional bounded (and analytically invertible) deviation warp for extrapolation; (v) a linear least-squares formulation for fitting controls and a practical numerical projection procedure; and (vi) experiments and baselines clarifying trade-offs against RBF and neural alternatives.

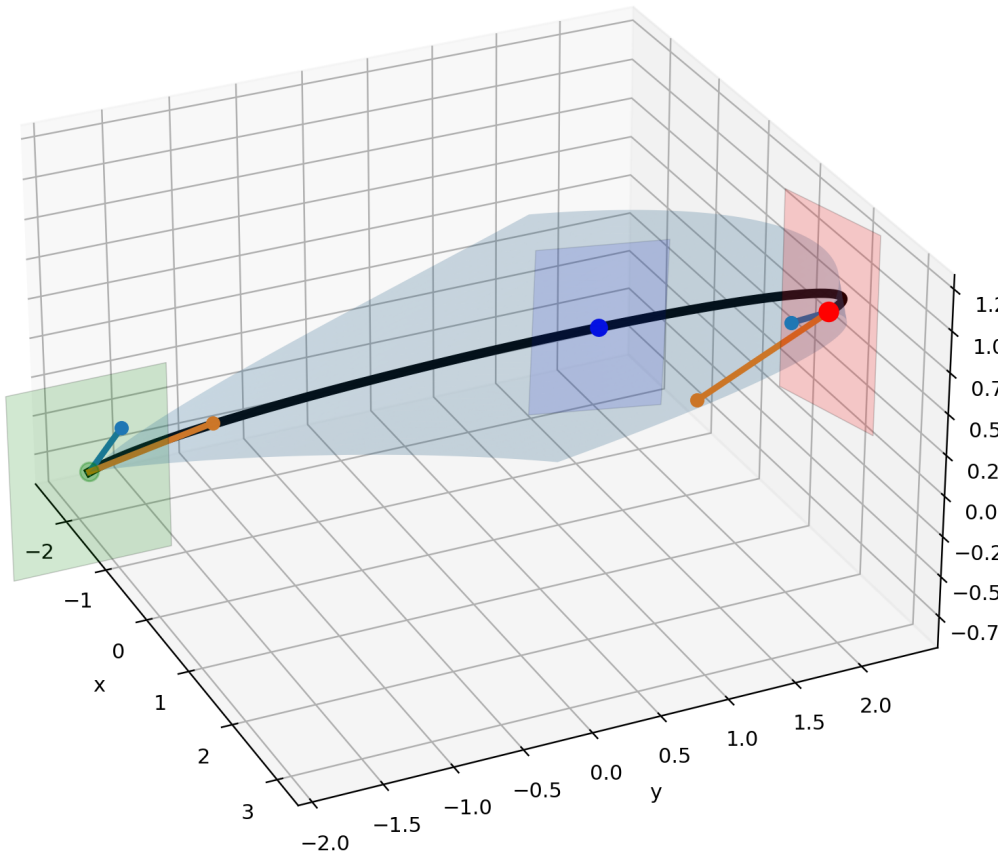


Figure 3: A K=2 splanifold in $\mathbb{R}^3$ visualized as a ribbon surface. The black curve is the spine (the δ=0 diagonal), endpoints are anchors, and the translucent planes depict transverse frames spanned by basis vectors. Colored segments show per-parameter endpoint tangent contributions that blend into the spine tangent via weights $w_k(u)$.

2 Preliminaries

Notation. We will use the following variables throughout:

- $r \in [0,1]^K$: input parameter vector.
- $u \in \mathbb{R}^K$: expanded effective coordinate (Eq. (3)).
- $\Sigma = \mathbf{1}^\top u$: summed coordinate.
- $t = \Sigma/K$: mean coordinate (spine parameter).
- $\delta = u - t\mathbf{1}$: zero-sum deviation with $\mathbf{1}^\top \delta = 0$.
- $\tilde{\delta}$: optionally warped deviation (Eq. (5)).

- $w(u) \in \mathbb{R}^K$: tangent-blending weights (Eq. (7)).

Cubic Hermite interpolation. Given endpoints $A, B \in \mathbb{R}^N$ and endpoint derivatives $A', B' \in \mathbb{R}^N$, the cubic Hermite segment $H(A, B, A', B'; t)$ for $t \in \mathbb{R}$ is [1, 4]

$$H(A, B, A', B'; t) = h_{00}(t)A + h_{01}(t)B + h_{10}(t)A' + h_{11}(t)B', \tag{1}$$

with basis functions

$$h_{00} = 2t^3 - 3t^2 + 1, \quad h_{01} = -2t^3 + 3t^2, \quad h_{10} = t(t-1)^2, \quad h_{11} = t^2(t-1). \tag{2}$$

On $t \in [0, 1]$, $\max(|h_{10}|, |h_{11}|) = 4/27$. Shape-preserving monotone piecewise-cubic Hermite variants (PCHIP) are described in [6, 7].

3 K-Splanifold Construction

3.1 Control data

A two-node K-Splanifold is specified by endpoint data at nodes $i \in \{0, 1\}$:

- an anchor point $P_i \in \mathbb{R}^N$,
- a transverse basis matrix $E_i \in \mathbb{R}^{N \times K}$ whose columns $e_{i,k}$ are basis vectors,
- a spine-derivative contribution matrix $P_i' \in \mathbb{R}^{N \times K}$ whose columns $p_{i,k}'$ contribute to the spine tangent,
- a basis-derivative matrix $E_i' \in \mathbb{R}^{N \times K}$ whose columns $e_{i,k}'$ contribute to the t-derivative of each basis trajectory.

All control storage is $\mathcal{O}(KN)$.

3.2 Spine–deviation factorization

Given $r \in [0, 1]^K$, we optionally expand the domain by a scalar $E \geq 0$:

$$u = (1 + 2E)r - E\mathbf{1} \in [-E, 1 + E]^K. \tag{3}$$

Define

$$\Sigma = \mathbf{1}^\top u, \qquad t = \frac{\Sigma}{K}, \qquad \delta = u - t\mathbf{1}. \tag{4}$$

Then $\mathbf{1}^\top \delta = 0$ and the decomposition is unique.

3.3 Deviation warping for bounded extrapolation

For extrapolated queries it is sometimes desirable to prevent transverse growth from increasing without bound. We use a smooth radial warp

$$\tilde{\delta} = \mathcal{W}(\delta) \triangleq \frac{\delta}{\sqrt{1 + (\|\delta\| / \delta_{\max})^2}}, \tag{5}$$

where $\delta_{\max} > 0$ is a chosen deviation radius. For $\|\delta\| \ll \delta_{\max}$, $\tilde{\delta} \approx \delta$; for $\|\delta\| \gg \delta_{\max}$, $\left\|\tilde{\delta}\right\| \to \delta_{\max}$.

Closed-form inverse of the warp. The map $\mathcal{W}$ is a smooth bijection from $\mathbb{R}^K$ onto the open ball $\{z : \|z\| < \delta_{\max}\}$. For any $\tilde{\delta}$ with $\left\|\tilde{\delta}\right\| < \delta_{\max}$,

$$\delta = \mathcal{W}^{-1}(\tilde{\delta}) \triangleq \frac{\tilde{\delta}}{\sqrt{1 - \left(\left\|\tilde{\delta}\right\| / \delta_{\max}\right)^2}}. \tag{6}$$

In what follows, the warp is optional; setting $\delta_{\max} \to \infty$ recovers $\tilde{\delta} = \delta$.

3.4 Smooth tangent weights

To blend per-parameter tangent contributions into a single spine tangent, define weights $w(u) \in \mathbb{R}^K$ with $\sum_k w_k = 1$. The unregularized normalization $w_k = u_k/\Sigma$ is undefined at $\Sigma = 0$. We instead use a smooth regularization parameter $\varepsilon_\Sigma > 0$:

$$w_k(u) = \frac{\Sigma\, u_k + \varepsilon_\Sigma^2/K}{\Sigma^2 + \varepsilon_\Sigma^2}. \tag{7}$$

This satisfies $\sum_k w_k(u) = 1$ for all u and is C^∞ in Σ.

Relation to unregularized weights. For $\Sigma \neq 0$, define $w_k^{(0)} = u_k/\Sigma$. A direct rearrangement of Eq. (7) yields

$$w(u) = w^{(0)}(u) - \frac{\varepsilon_\Sigma^2}{\Sigma\left(\Sigma^2 + \varepsilon_\Sigma^2\right)}\,\delta. \tag{8}$$

In particular, $w(u) = w^{(0)}(u)$ whenever $\delta = 0$ (on the diagonal). Moreover,

$$\left\|w(u) - w^{(0)}(u)\right\|_2 \leq \frac{\varepsilon_\Sigma^2}{|\Sigma|\left(\Sigma^2 + \varepsilon_\Sigma^2\right)}\,\|\delta\|_2\,. \tag{9}$$

Thus for $|\Sigma| \gg \varepsilon_\Sigma$ the bias decays as $\mathcal{O}((\varepsilon_\Sigma/\Sigma)^2)$.

3.5 Evaluation

Given $(t, \tilde{\delta}, w)$, define endpoint spine derivatives

$$V_i(u) = P_i'\, w(u) \in \mathbb{R}^N, \tag{10}$$

and evaluate the spine

$$\Psi(t) = H(P_0, P_1, V_0, V_1; t). \tag{11}$$

For the transverse field, define endpoint displacements

$$D_i(t, \tilde{\delta}) = E_i\, \tilde{\delta}, \tag{12}$$

and endpoint displacement derivatives

$$T_i(t, \tilde{\delta}) = E_i'\, \tilde{\delta}. \tag{13}$$

Then evaluate the transverse displacement

$$\Delta(t, \tilde{\delta}) = H(D_0, D_1, T_0, T_1; t), \tag{14}$$

and output

$$\Phi(r) = \Psi(t) + \Delta(t, \tilde{\delta}). \tag{15}$$

Algorithm 1 K-Splanifold evaluation

Require: $r \in [0,1]^K$, controls $(P_i, E_i, P_i', E_i')_{i\in\{0,1\}}$, expansion $E \geq 0$, $\varepsilon_\Sigma > 0$, deviation radius $\delta_{\max}$

$u \leftarrow (1+2E)r - E\mathbf{1}$
$\Sigma \leftarrow \mathbf{1}^\top u$
$t \leftarrow \Sigma/K$
$\delta \leftarrow u - t\mathbf{1}$
$\tilde{\delta} \leftarrow \delta/\sqrt{1+(\|\delta\|/\delta_{\max})^2}$ {optional; set $\delta_{\max} \to \infty$ to disable}
$w_k \leftarrow (\Sigma u_k + \varepsilon_\Sigma^2/K)/(\Sigma^2 + \varepsilon_\Sigma^2)$ for $k = 1, \ldots, K$
$V_i \leftarrow P_i' w$, $D_i \leftarrow E_i\tilde{\delta}$, $T_i \leftarrow E_i'\tilde{\delta}$ for $i \in \{0,1\}$
$\Psi \leftarrow H(P_0, P_1, V_0, V_1; t)$
$\Delta \leftarrow H(D_0, D_1, T_0, T_1; t)$
return $\Phi \leftarrow \Psi + \Delta$

Basis-trajectory view. For fixed t, $\Delta(t, \tilde{\delta})$ is linear in $\tilde{\delta}$. Defining basis trajectories

$$b_k(t) = H\big(e_{0,k}, e_{1,k}, e'_{0,k}, e'_{1,k}; t\big)\,, \tag{16}$$

the displacement is $\Delta(t, \tilde{\delta}) = \sum_{k=1}^K \tilde{\delta}_k\, b_k(t)$.

Complexity and arithmetic intensity. Evaluation uses a small number of $N \times K$ matrix–vector products and costs $\mathcal{O}(KN)$ per query. For large N, the work is often memory-bandwidth dominated: the dominant cost is streaming the control matrices E_i, E_i', P_i' from memory rather than arithmetic. Practical implementations benefit from contiguous storage, fused kernels, and caching when evaluating many queries against fixed controls.
Algorithm 1 summarizes the evaluation pipeline and makes the $\mathcal{O}(KN)$ structure explicit.

4 Properties

Diagonal collapse. If all components of r are equal, then $\delta = 0$ in Eq. (4); by Eq. (15) the transverse term vanishes and $\Phi(r) = \Psi(t)$ lies on the spine.

Affine equivariance. If an affine transform $x \mapsto Ax + b$ is applied to all control vectors (P_i, E_i, P_i', E_i'), then the evaluated manifold is transformed by the same map. This follows because Hermite interpolation is affine-linear in its endpoint data [1, 4] and all terms in Eq. (15) are sums of such interpolants.

Slice geometry. For fixed t, the displacement is linear in $\tilde{\delta}$: by Eq. (16), $\Delta(t, \tilde{\delta}) = \sum_k \tilde{\delta}_k\, b_k(t)$. Each constant-$t$ slice is therefore an affine subspace of dimension at most $K-1$ (since $\mathbf{1}^\top\tilde{\delta} = 0$).

Smoothness. For $\varepsilon_\Sigma > 0$, the weights $w(u)$ in Eq. (7) are smooth for all u, hence the blended spine derivatives $V_i(u)$ are smooth. Since Ψ and Δ are cubic Hermite polynomials in t [1, 4], Eq. (15) is smooth in t. With the deviation warp (5), the full mapping remains smooth for extrapolated queries while transverse magnitude stays bounded in $\tilde{\delta}$.

Weight-induced spine bias bound. Let $\Psi^{(0)}$ denote the spine evaluated with unregularized weights $w^{(0)}$ (defined when $\Sigma \neq 0$). For $t \in [0,1]$,

$$\left\|\Psi(t) - \Psi^{(0)}(t)\right\|_2 \leq \frac{4}{27}\left(\left\|P_0'\right\|_2 + \left\|P_1'\right\|_2\right) \frac{\varepsilon_\Sigma^2}{|\Sigma|(\Sigma^2+\varepsilon_\Sigma^2)} \left\|\delta\right\|_2, \tag{17}$$

where $\|\cdot\|_2$ is the matrix operator norm. This follows from Eq. (9) and the derivative coefficients h_{10}, h_{11} in Eq. (2).

Cubic reproduction along the spine. Cubic Hermite interpolation exactly reproduces cubic polynomials [1, 4]: if $c(t)$ is a (vector-valued) cubic polynomial, then

$$H(c(0), c(1), c'(0), c'(1); t) \equiv c(t). \tag{18}$$

Consequently, a $K{=}1$ splanifold segment represents any cubic polynomial curve exactly, and piecewise chains represent standard cubic splines. The same exactness holds for each basis trajectory $b_k(t)$ in Eq. (16).

5 Fitting and projection

The K-Splanifold map is linear in its control vectors and nonlinear in the input coordinate. This split supports two common inverse tasks: (i) fitting controls from example data, and (ii) projecting an embedding point onto the manifold.

5.1 Fitting controls by linear least squares

Given sample pairs $\{(r_j, x_j)\}_{j=1}^S$ with $x_j \in \mathbb{R}^N$, compute $(t_j, \tilde{\delta}_j, w_j)$ and Hermite scalars $\alpha_{0j} = h_{00}(t_j)$, $\alpha_{1j} = h_{01}(t_j)$, $\beta_{0j} = h_{10}(t_j)$, $\beta_{1j} = h_{11}(t_j)$. Then Eq. (15) expands to

$$\begin{aligned}\Phi(r_j) = {} & \alpha_{0j}P_0 + \alpha_{1j}P_1 + \beta_{0j}(P_0' w_j) + \beta_{1j}(P_1' w_j) \\ & + \alpha_{0j}(E_0\tilde{\delta}_j) + \alpha_{1j}(E_1\tilde{\delta}_j) + \beta_{0j}(E_0'\tilde{\delta}_j) + \beta_{1j}(E_1'\tilde{\delta}_j).\end{aligned} \tag{19}$$

Collect the unknown controls into a single matrix

$$C \triangleq [\, P_0 \; P_1 \; P_0' \; P_1' \; E_0 \; E_1 \; E_0' \; E_1' \,] \in \mathbb{R}^{N\times(2+6K)}, \tag{20}$$

and define per-sample feature vectors

$$f_j \triangleq [\, \alpha_{0j},\, \alpha_{1j},\, \beta_{0j} w_j^\top,\, \beta_{1j} w_j^\top,\, \alpha_{0j}\tilde{\delta}_j^\top,\, \alpha_{1j}\tilde{\delta}_j^\top,\, \beta_{0j}\tilde{\delta}_j^\top,\, \beta_{1j}\tilde{\delta}_j^\top \,]^\top. \tag{21}$$

Then $\Phi(r_j) = Cf_j$ and fitting controls reduces to the matrix least squares problem

$$\min_C \sum_{j=1}^S \|Cf_j - x_j\|_2^2 = \min_C \|CF - X\|_F^2, \tag{22}$$

where $F = [f_1, \ldots, f_S] \in \mathbb{R}^{(2+6K)\times S}$ and $X = [x_1, \ldots, x_S] \in \mathbb{R}^{N\times S}$. A standard ridge-stabilized (Tikhonov-regularized) solution [22, 23, 24] is

$$C = XF^\top(FF^\top + \lambda I)^{-1}. \tag{23}$$

Forming $FF^\top$ costs $\mathcal{O}(S(2+6K)^2)$; multiplying $XF^\top$ costs $\mathcal{O}(NS(2+6K))$.

Method	Storage	Per-query cost	Notes
Dense tensor-product spline	$\mathcal{O}(M^K N)$	$\mathcal{O}(4^K N)$	Smooth, local support, but grid explosion in K.
Sparse grids (Smolyak)	sub-exponential in K	varies	Retains selected interaction terms; basis/level management matters.
RBF interpolation	$\mathcal{O}(C(K+N))$	$\mathcal{O}(C(K+N))$	Scattered controls; global solve or fit; C = centers.
Coordinate MLP / neural field	params depend on width	$\mathcal{O}$(FLOPs)	Expressive; requires training and hyperparameters.
K-Splanifold	$\mathcal{O}(KN)$	$\mathcal{O}(KN)$	Structured spine–deviation geometry; explicit controls.
K-Splanifold + sparse residual	$\mathcal{O}((K+C)N)$	$\mathcal{O}(KN + C(K+N))$	Adds localized corrections when needed.

Table 1: Qualitative scaling summary. M = grid resolution per axis for tensor products; C = number of RBF centers or sparse residual centers.

5.2 Projecting an embedding point

Given $x \in \mathbb{R}^N$, a common "inverse" operation is to compute a parameter $r^* \in [0,1]^K$ that minimizes

$$r^* \in \arg\min_{r \in [0,1]^K} \|\Phi(r) - x\|_2^2 . \tag{24}$$

In general Eq. (24) is a small nonlinear least squares problem because $w(u)$ and the factorization $r \mapsto (t, \delta)$ are nonlinear. In practice, Gauss–Newton or quasi-Newton methods [25] work well because each evaluation of Φ and its Jacobian is $\mathcal{O}(KN)$, and the inner linear solve is only $K \times K$. When the deviation warp is enabled, $\tilde{\delta}$ can be unwarped analytically using Eq. (6) inside the optimization if a parameterization in terms of $\tilde{\delta}$ is preferred.

6 Baselines and positioning

K-Splanifolds are best viewed as a structured, low-rank interpolation primitive rather than a replacement for general-purpose high-dimensional methods. The canonical alternatives in Table 1 are well studied: tensor-product splines [1, 2], sparse grids [8, 9], RBF interpolation [10, 11], and coordinate MLP/neural-field models [14, 15, 20, 19]. Table 1 summarizes scaling and common trade-offs.

Both splines and neural networks have extensive approximation theory; see, e.g., spline approximation texts [2, 5] and neural approximation results [16, 17].

7 Experiments

All timings were measured in a single CPU thread using vectorized NumPy implementations with $N = 256$ output dimensions. Stochastic experiments use fixed seeds and single-threaded deterministic execution to control variability.

7.1 Qualitative visualization

Figure 3 shows the control fields and transverse frames for a $K{=}2$ example. Figure 4 shows the resulting ribbon surface without overlays.

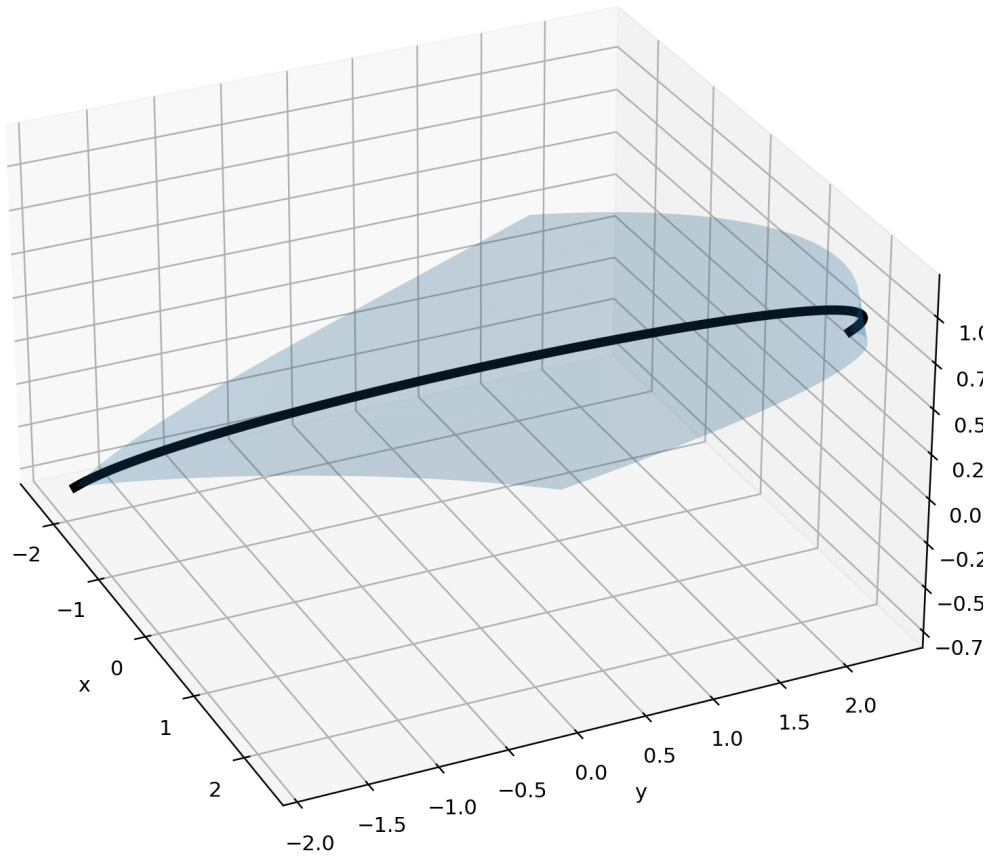


Figure 4: A $K{=}2$ example surface (spine in black). The transverse "breadth" is produced by basis trajectories $b_k(t)$ blended across the two endpoints.

7.2 Step-by-step evaluation

Figure 5 illustrates the decomposition $r \mapsto (t, \delta)$ and the embedding-space sum $\Phi = \Psi + \Delta$. Figure 6 visualizes $\Delta(t, \delta)$ as a weighted sum of basis trajectories at a fixed t.

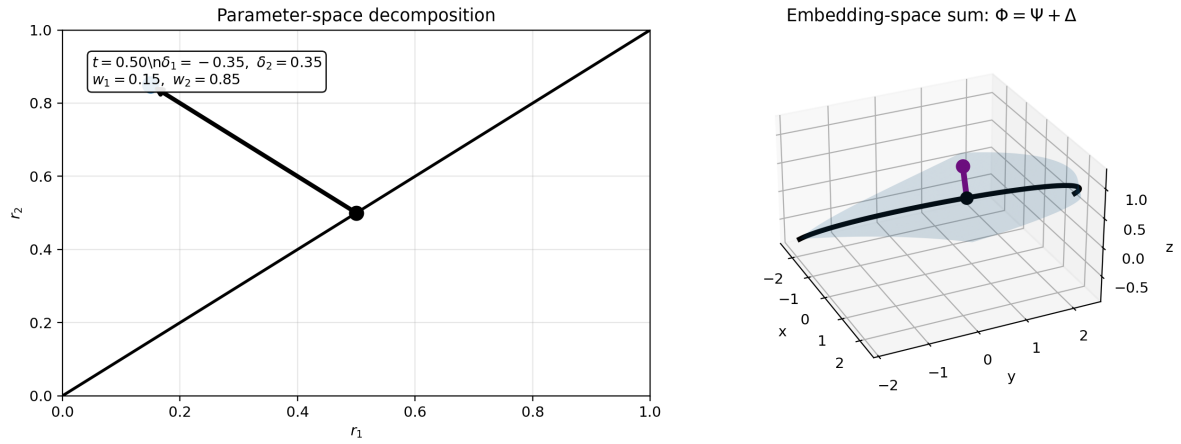


Figure 5: Decomposition and reconstruction at a sample query point. Left: r decomposes into a diagonal projection (spine coordinate t) plus a zero-sum deviation δ. Right: the output splits into a spine point plus a transverse displacement.

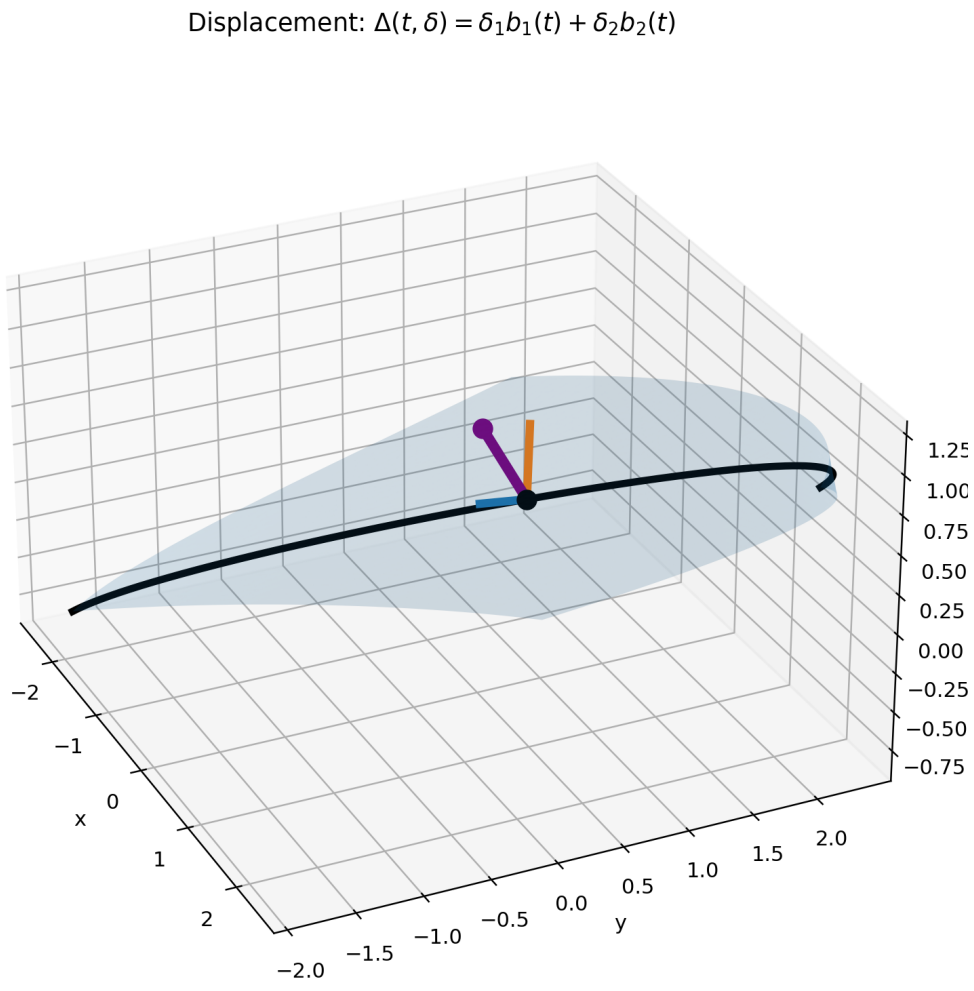


Figure 6: Displacement composition at a fixed t: $\Delta(t, \delta) = \sum_k \delta_k \, b_k(t)$. This linear structure is a key source of interpretability and speed.

7.3 Smooth weight regularization

Figure 7 plots $\max_k |w_k|$ as $\Sigma \to 0$ for the unregularized weights u_k/Σ and the smooth regularization (7). The regularized weights remain bounded and approach uniform weights at $\Sigma = 0$.

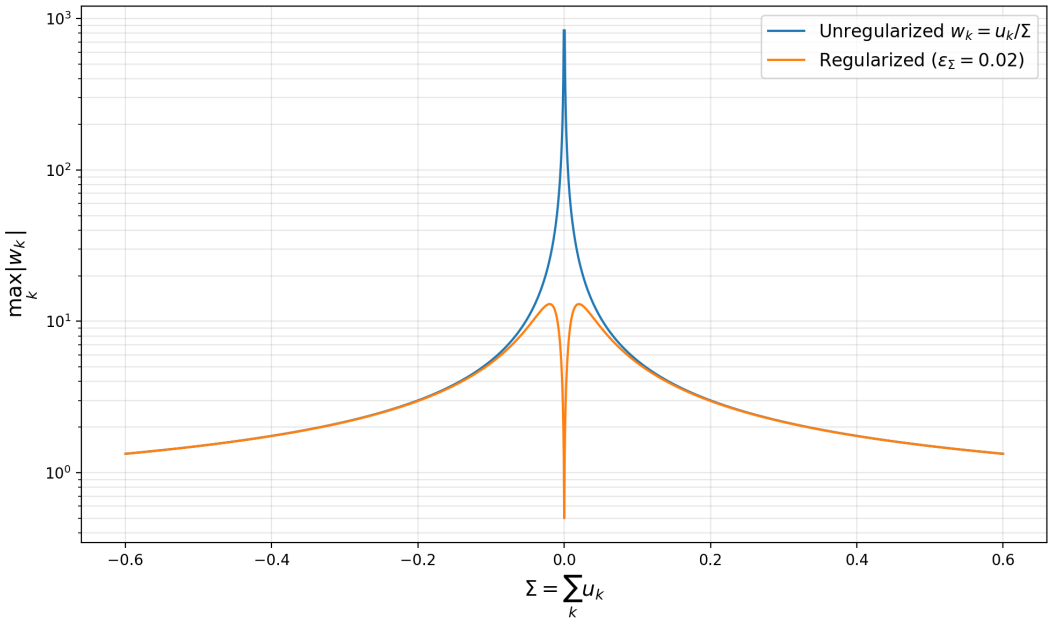


Figure 7: Tangent-weight behavior near $\Sigma = 0$. The unregularized normalization diverges for off-diagonal deviations, while the regularized weights remain finite and smooth.

7.4 Deviation warping for bounded extrapolation

Figure 8 shows how the deviation warp (5) bounds transverse magnitude as the deviation norm increases.

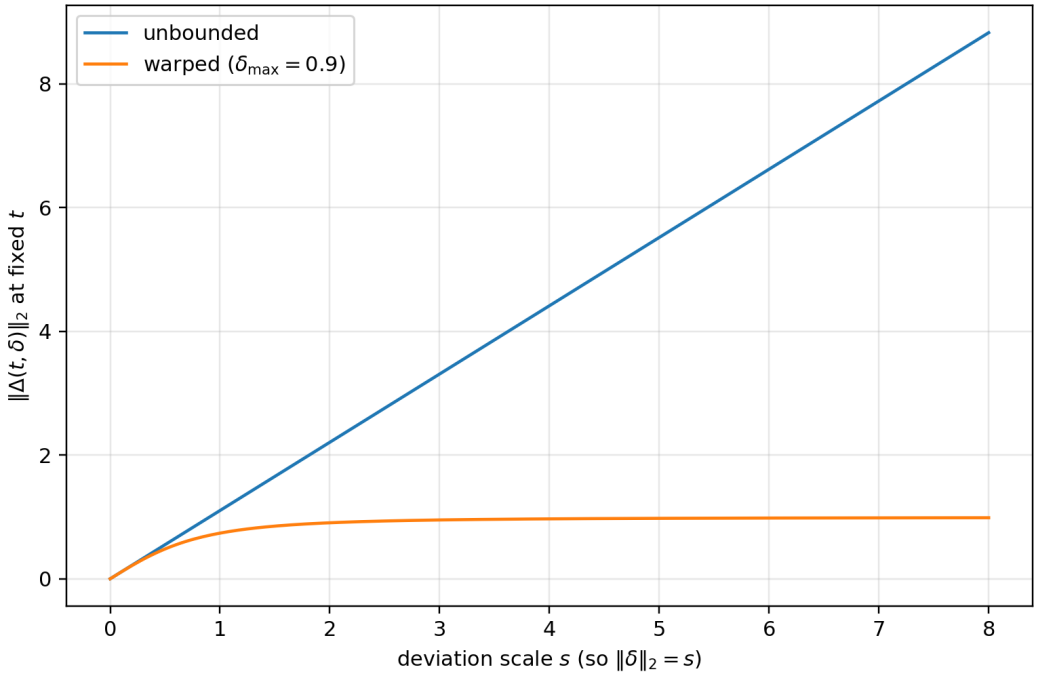


Figure 8: Deviation warping bounds transverse magnitude for large deviations while remaining approximately linear near the origin.

7.5 Runtime baselines

Figure 1 compares K-Splanifold evaluation time against an RBF interpolator (Gaussian RBF [10, 11], C=256 centers) and a small coordinate-MLP baseline (two hidden layers of width 256, ReLU). We also plot a local cubic tensor-product evaluation cost $\mathcal{O}(4^K N)$ for small K where it is still tractable to materialize the active control set. Table 2 reports representative microseconds per query.

K	K-Splanifold	RBF (C=256)	MLP (2×256)
2	2.73	12.18	5.40
4	3.00	25.51	6.58
8	2.83	37.23	3.98
16	3.44	60.37	4.46
32	3.43	96.42	4.75
64	5.20	214.85	4.93
128	15.39	447.35	5.38
256	21.74	779.06	8.29

Table 2: Representative microseconds per query (single CPU thread, N=256).

Notes on neural baselines. We use a plain ReLU coordinate-MLP as a lightweight, widely available reference (universal approximation results in [14, 15]). Specialized neural field architectures (e.g., periodic activations [20] or multiresolution hash-grid encodings [21]) can offer different accuracy/speed trade-offs.

7.6 Curve fitting under a fixed optimization budget

To probe the practical behavior of spline-based models under a fixed iteration budget, we fit several standard 3D parametric curves using three lightweight families: (i) a *splanifold curve* (the K=1 special case) implemented as a chain of 16 cubic Hermite segments [1, 4], (ii) a Gaussian RBF model with C=64 centers [10, 11], and (iii) a small coordinate-MLP with two hidden layers of width 64, tanh activations, and Fourier feature inputs [18]. We report trainable parameter counts (float32) to contextualize model memory: the splanifold curve uses 17 knots with positions and tangents in $\mathbb{R}^3$ (102 scalars, 0.40 KB), the RBF uses C=64 centers with 3D weights and bias (259 scalars, 1.01 KB; kernel width selected by validation), and the MLP has 5763 trainable parameters (22.5 KB; fixed Fourier features not counted).

All models were optimized with Adam [26] for 2000 steps on 128 training samples of $t \in [0, 1]$ and evaluated on 128 held-out samples. For the RBF baseline, the kernel width σ was selected per-seed by a log-spaced search on a small validation split of the training set. We report mean $\pm$ std test RMSE over three random seeds.

Table 3 reports test RMSE after 2000 steps, and Fig. 2 shows RMSE versus iteration checkpoints for three targets (circle, helix, trefoil knot).

Target	Splanifold curve	RBF (C=64)	MLP (2×64)
Circle	0.0014 ± 0.0009	0.0158 ± 0.0113	0.0066 ± 0.0039
Helix	0.0024 ± 0.0013	0.0146 ± 0.0068	0.0234 ± 0.0198
Trefoil	0.0023 ± 0.0004	0.0242 ± 0.0194	0.0045 ± 0.0011

Table 3: Curve-fitting test RMSE after 2000 Adam steps (mean ± std over 3 seeds).

7.7 Optional sparse residual layer

To add localized detail without changing the global spine–deviation structure, we can add a small residual field on parameter space:

$$\Phi_{\text{hybrid}}(r) = \Phi(r) + \sum_{j=1}^{C} \kappa(\|r - c_j\|)\, a_j, \tag{25}$$

where $c_j \in [0, 1]^K$ are centers, $a_j \in \mathbb{R}^N$ are residual vectors, and κ is a compactly supported kernel [11, 12]. Figure 9 shows a single localized bump added to the K=2 example. The added per-query cost is $\mathcal{O}(C(K+N))$.

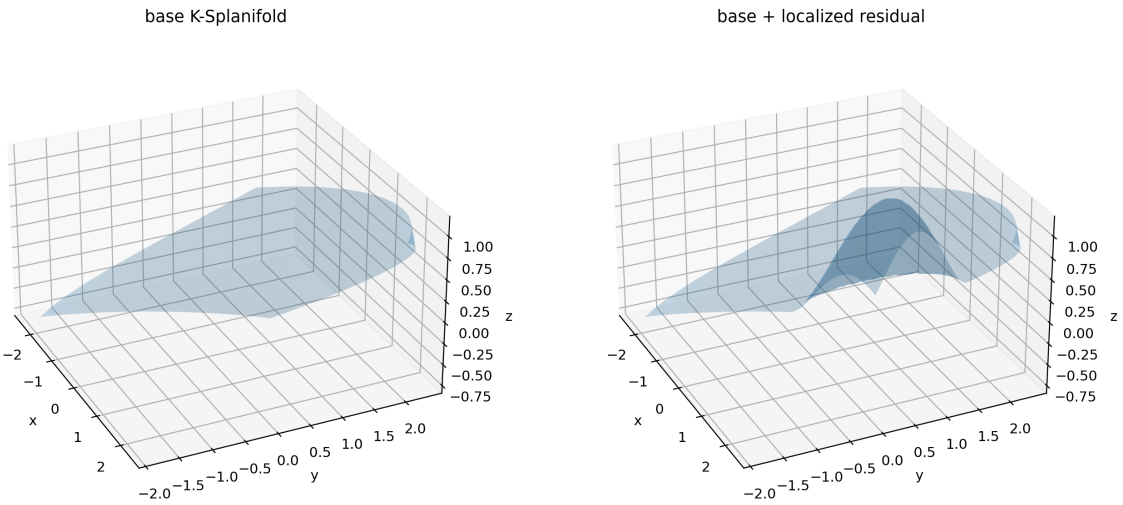


Figure 9: A localized residual adds corner detail while preserving the base K-Splanifold structure.

8 Per-parameter entropy bits and stored information in regression

Parameter count is a useful but coarse proxy for model complexity; description-length viewpoints formalize complexity directly in bits [27, 29]. For regression-style uses of K-Splanifolds, it is often helpful to quantify how many *bits* are actually needed to store a fitted model at a given predictive fidelity. This section defines a practical *entropy-coded bits per parameter* metric and illustrates it on two synthetic $K{=}4 \to N{=}4$ regression tasks, comparing a two-node K-Splanifold fit to a small coordinate-MLP whose size is adjusted so test RMSE is comparable.

8.1 Entropy-coded bits per parameter

Let the trainable parameters be partitioned into G blocks (tensors) $\{\theta^{(g)}\}_{g=1}^{G}$ with P_g scalars each and total $P = \sum_g P_g$. For a given b (bits), we apply a symmetric uniform quantizer independently per block [31, 32], $q^{(g)} = Q_b(\theta^{(g)}) \in \mathbb{Z}^{P_g}$, and form the empirical code distribution

$$p_g(k) = \frac{1}{P_g} \left|\{i : \; q_i^{(g)} = k\}\right|. \tag{26}$$

The Shannon entropy of that block's codes,

$$H\left(q^{(g)}\right) = -\sum_k p_g(k) \log_2 p_g(k), \tag{27}$$

is the ideal lossless codelength (bits/parameter) for that stream under entropy coding [28, 29, 30]. We report the *entropy-coded bits per parameter*

$$H_{\text{pp}}(b) = \frac{1}{P} \sum_{g=1}^{G} P_g \, H\left(q^{(g)}\right), \tag{28}$$

which is the ideal average codelength (bits/parameter) if each tensor is coded as its own stream (ignoring small headers such as per-tensor scales).

Fidelity-preserving precision. Because H_{pp} depends on the quantizer, we choose a task-dependent precision by sweeping b and selecting the smallest $b^\star$ such that the quantized model's test RMSE stays within 5% of its unquantized (float) RMSE.[1] We then report $H_{\text{pp}}(b^\star)$ as the *effective entropy bits per parameter* at that fidelity. This is a lightweight analogue of "trainable parameter entropy" metrics used in neural compression [34], but applied equally to spline-structured and MLP models.

8.2 Examples: smooth cubic vs. noisy irregular in 4D

We consider inputs $r \in [0,1]^4$ and outputs $y \in \mathbb{R}^4$ with the decomposition $t = \frac{1}{4}\mathbf{1}^\top r$ and $\delta = r - t\mathbf{1}$.

Models. *K-Splanifold:* a two-node model with $K{=}N{=}4$ fit by ridge least squares using Eq. (23). The parameter count is $P = N(2+6K) = 104$. *MLP:* a coordinate-MLP $r \mapsto y$ with one hidden tanh layer (universal approximation [14, 15]). We choose the hidden width (and modest training budget) so the MLP's float test RMSE matches the K-Splanifold within $\approx 5\%$ on each task (here, width 64; $P{=}580$).

Targets. *Smooth cubic (well-suited):*

$$y = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \left(B_0 + B_1 t + B_2 t^2\right)\delta + \varepsilon, \tag{29}$$

with fixed coefficients (a_i, B_i) and small noise $\varepsilon \sim \mathcal{N}(0, 0.07^2 I)$. This construction is cubic in t and linear in δ, making it well matched to the K-Splanifold factorization. *Noisy irregular (badly suited):* we add high-frequency variation in the full 4D input via random Fourier mixtures plus larger noise [33, 18],

$$y = y_{\text{cubic}} + \gamma \sum_{m=1}^{M} \sin\!\left(2\pi\omega_m k_m^\top r + \phi_m\right) v_m + \varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, 0.18^2 I), \tag{30}$$

which is poorly matched to a low-rank spine–deviation parameterization. We train on 256 random samples and report RMSE on 256 held-out samples against the *noise-free* target.

Quantization and entropy. We apply symmetric uniform b-bit quantization per parameter tensor and compute both RMSE and $H_{\text{pp}}(b)$ from Eq. (28). Figure 10 plots RMSE as a function of the quantization bits per parameter b. Table 4 summarizes the smallest fidelity-preserving $b^\star$, the corresponding $H_{\text{pp}}(b^\star)$, and the implied ideal entropy-coded sizes.

Task	Model	P	RMSE (float)	$b^\star$	RMSE ($b^\star$)	$H_{\text{pp}}(b^\star)$	Size (bytes)
cubic	K-Splanifold (K=N=4)	104	0.0232	6	0.0239	3.71	48.2
cubic	MLP (1×64)	580	0.0221	7	0.0221	5.94	430.4
irregular	K-Splanifold (K=N=4)	104	0.4449	4	0.4627	2.97	38.6
irregular	MLP (1×64)	580	0.4409	6	0.4414	4.70	341.1

Table 4: Entropy-coded bits per parameter for fitted regression models. We choose the smallest $b^\star$ such that quantization increases test RMSE by at most 5%, then report $H_{\text{pp}}(b^\star)$ from Eq. (28). "Size" is the ideal entropy-coded lower bound $P\,H_{\text{pp}}(b^\star)/8$ (header overheads ignored).

[1] Any application-specific tolerance can be substituted.

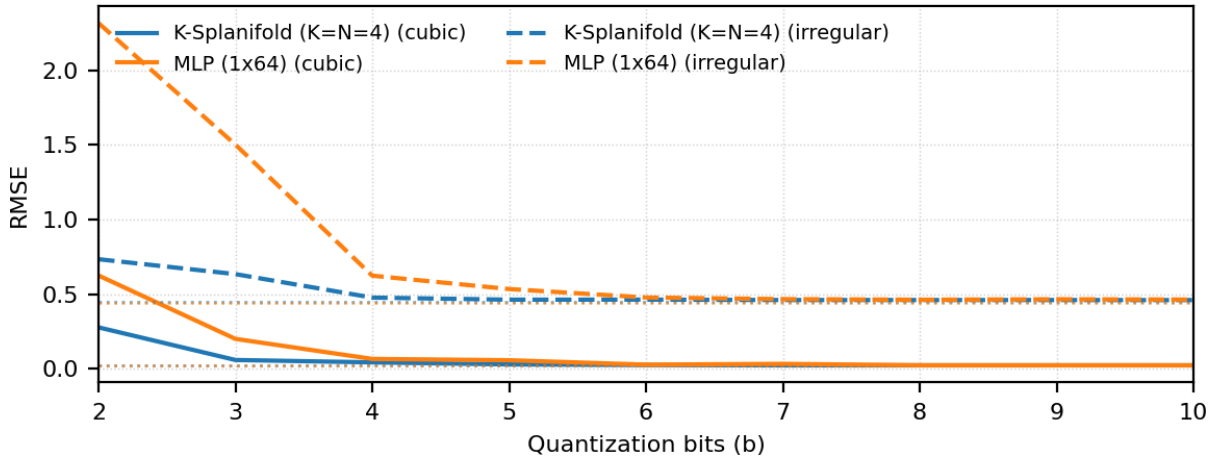


Figure 10: Quantization sensitivity expressed in quantization bits per parameter for two 4D $\to$ 4D regression targets. Each curve sweeps $b \in \{2, \ldots, 10\}$ and plots test RMSE versus b. Dashed horizontal lines show each model's unquantized RMSE baseline.

Interpretation. On the smooth cubic target, both models reach comparable RMSE (≈ 0.023), but the K-Splanifold reaches that fidelity with fewer entropy bits per parameter ($H_{\text{pp}} \approx 3.71$) than the MLP ($H_{\text{pp}} \approx 5.94$). On the noisy irregular target, both models again have comparable RMSE (≈ 0.44), and the MLP still requires higher entropy at fidelity ($H_{\text{pp}} \approx 4.70$ vs. 2.97), consistent with storing more information to represent irregular variation under this parameterization. One plausible reading is that the MLP stores more "noise-like" detail (higher entropy per parameter) to match the same test RMSE. However, higher entropy does not guarantee lower MSE: extra capacity can be redundant, less aligned with the target, or difficult to exploit under the same optimization and regularization, so the additional bits do not necessarily translate into improved error. This aligns with prior work showing substantial parameter redundancy and compressibility in neural networks (e.g., predicting most weights without accuracy loss or achieving large compression ratios via pruning/quantization/weight sharing), which suggests that distributed representations can store information in a highly redundant form [35, 34, 36]. Connectionist models emphasize distributed, superpositional representations, and holographic reduced representations provide a concrete formalism for such superposition in fixed-width vectors; in that light, higher H_{pp} for MLPs is consistent with a more redundant encoding rather than strictly more useful signal [37, 38].

9 Practical notes

Choosing transverse bases. If the columns of E_i are nearly linearly dependent, transverse variation can collapse or become poorly conditioned. In practice we recommend authoring E_i as an (approximately) orthogonal frame and monitoring the condition number of the Gram matrix $G_i = E_i^\top E_i$. When automatic conditioning is desired while approximately preserving per-axis semantics, a modified Gram–Schmidt pass with per-column renormalization is often sufficient [24].

When to use K-Splanifolds. K-Splanifolds are a good fit when (i) a meaningful "central progression" exists, (ii) deviations away from that progression can be described by per-axis basis trajectories, and (iii) explicit, low-latency, differentiable evaluation is preferred over training-based alternatives. When strong heterogeneous interactions dominate, sparse grids, RBFs, or neural fields

may be better suited [8, 9, 10, 11, 20, 19].

10 Conclusion

K-Splanifolds provide a structured spine–deviation manifold with linear scaling in the input dimension and explicit, interpretable controls. A smooth weight regularization removes singular behavior in tangent blending and admits a quantitative bound on the deviation from unregularized normalization. Deviation warping offers bounded transverse behavior for extrapolation and can be inverted analytically on the warped deviation. Experiments and baselines place the method among common fast interpolators (RBFs and coordinate MLPs) and illustrate its practical behavior under fixed iteration budgets.

References

[1] Carl de Boor. *A Practical Guide to Splines*. Springer, revised edition, 2001.

[2] Larry L. Schumaker. *Spline Functions: Basic Theory*. Cambridge University Press, 3rd edition, 2007.

[3] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh. *The Theory of Splines and Their Applications*. Academic Press, 1967.

[4] Gerald Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Morgan Kaufmann, 5th edition, 2002.

[5] Grace Wahba. *Spline Models for Observational Data*. SIAM, 1990.

[6] F. N. Fritsch and R. E. Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[7] J. M. Hyman. Accurate monotonicity preserving cubic interpolation. *SIAM Journal on Scientific and Statistical Computing*, 4(4):645–654, 1983.

[8] S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Doklady Akademii Nauk SSSR*, 1963.

[9] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[10] Martin D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003.

[11] Holger Wendland. *Scattered Data Approximation*. Cambridge University Press, 2005.

[12] Gregory E. Fasshauer. *Meshfree Approximation Methods with MATLAB*. World Scientific, 2007.

[13] Trevor Hastie and Werner Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[14] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[15] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[16] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[17] Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

[18] 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 *NeurIPS*, 2020.

[19] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[20] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.

[21] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4), 2022.

[22] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics Doklady*, 4:1035–1038, 1963.

[23] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[24] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4th edition, 2013.

[25] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[27] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[28] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948.

[29] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

[30] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[31] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[32] N. S. Jayant and Peter Noll. *Digital Coding of Waveforms*. Prentice Hall, 1984.

[33] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NeurIPS*, 2007.

[34] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In *ICLR*, 2016.

[35] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning. In *NeurIPS*, 2013.

[36] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. In *ICLR*, 2017.

[37] David E. Rumelhart and James L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1: Foundations. MIT Press, 1986.

[38] Tony A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995.