LEARNED FUNCTIONAL PROJECTION: LEARNING FOR RECOVERY IN IMPLIED VOLATILITY SMOOTHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Implied volatility smoothing requires recovering continuous surfaces from sparse, irregular market quotes, yet approximation quality alone does not determine recovery accuracy. We prove that representations with zero approximation error can have arbitrarily large recovery-risk ratios under the same observations and coefficient-fitting rule. This motivates selecting representations by their recovery risk under the available observations. We introduce Learned Functional Projection (LFP), which learns a low-dimensional continuous function space by optimizing recovery from partial quotes. Training differentiates through a weighted ridge solve and evaluates predictions on held-out quotes; each new surface requires only one explicit coefficient solve. Financial regularization penalizes shape violations. The explicit recovery rule admits exact error identities that separate approximation, regularization and input error, together with a deterministic pointwise upper bound and a representation-error lower bound on recovery risk. Experiments use 2012–2021 data from four equity-index option markets. On SPX under standard inputs, LFP reduces held-out vega-weighted relative RMSE by 61.8% against the graph neural operator (GNO). With the same nine reference quotes, LFP-9 reduces this error by 58.4% against HyperIV and achieves 3.9 times its bid–ask coverage. The SPX-trained representation also leads all evaluated baselines in quote accuracy on NDX and RUT without retraining. Surface construction takes milliseconds, with approximately $42\times$ CPU and $113\times$ GPU speedups over the public GNO implementation. Controlled ablations show that learning for recovery outperforms learning for reconstruction, supporting the method's design and the practical relevance of the theoretical distinction.

1 INTRODUCTION

Implied volatility smoothing constructs a continuous surface over strike and maturity from discrete option quotes. The surface must agree with observed prices, support evaluation at unquoted contracts and respect financial shape conditions, while remaining inexpensive to update for pricing and risk management (Wiedemann et al., 2025). The difficulty is that market quotes provide only finite, irregular and potentially noisy observations of the surface.

A useful structural prior is that volatility surfaces admit compact descriptions. SVI and SSVI capture smiles and term structure with few parameters (Gatheral & Jacquier, 2014), while factor models describe common historical variation (Fengler et al., 2007). Parametric methods exploit this structure through a prescribed functional family: calibration adapts its parameters, but does not learn the family itself. Neural function approximators offer greater flexibility (Ackerer et al., 2020; Wiedemann et al., 2025), although a general neural parameterization does not by itself impose a compact shared function space. This motivates combining a representation learned from data with a small set of coefficients for each surface.

Principal component analysis (PCA) is a natural starting point. It learns a low-dimensional linear representation of dominant historical variation (Jolliffe & Cadima, 2016); coefficients can then be fitted to partial observations, as in gappy reconstruction (Everson & Sirovich, 1995). This combines an adaptive representation with inexpensive fitting for new surfaces. Standard PCA, however, selects its space to minimize projection error on the training samples. That criterion does not directly optimize the predictions obtained after fitting coefficients to the quotes available for a new surface.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The distinction matters because market observations impose two constraints. First, complete surfaces are unavailable during both training and deployment. In image reconstruction settings such as masked autoencoding, removed pixels remain available as training targets (He et al., 2022); historical option data instead contain partial observations of different surfaces. A conventional grid-based PCA implementation must first complete these observations, so its training targets also reflect the interpolation procedure. Second, observation locations cannot be freely designed as in sensor placement (Manohar et al., 2018). Quoted strikes and maturities depend on listed contracts, trading activity and liquidity. Selecting among available quotes cannot create reliable observations at arbitrary coordinates. The learned representation must therefore suit the observation patterns supplied by the market.

Under these conditions, a good approximation space need not give good recovery. Approximation concerns the best surface a representation can express; recovery concerns the surface selected from the available quotes. For example, a historically important component may vary mainly in a maturity region with few current quotes. Retaining it can improve approximation while leaving its coefficient poorly determined during recovery. A representation that links variation in that region to well-observed maturities can instead use current quotes to inform both. The resulting design question is how to learn a compact function space by the recovery achieved after quote fitting.

We introduce Learned Functional Projection to address this question, building on Function Encoders (Ingebrand et al., 2025) and differentiable ridge solvers (Bertinetto et al., 2019). LFP retains low-dimensional coefficient fitting while learning the representation through recovery. A coordinate network supplies shared continuous basis functions. Training fits coefficients to part of a historical surface's quotes and uses prediction on the remaining quotes to learn the basis and ridge parameter. Supervision uses actual quotes and the fitting rule used at deployment. For each new surface, the shared parameters remain fixed and one weighted linear solve determines its coefficients. Continuous basis functions accommodate changing input and query locations without a common grid. Quote weighting and financial regularization address accuracy and shape requirements. Code is available at `https://anonymous.4open.science/r/lfp`.

Our contributions are threefold:

- **Theory: representation selection for recovery.** We prove that equal-dimensional representations with zero approximation error can have arbitrarily large recovery-risk ratios under the same observations and ridge parameter. For weighted ridge recovery, we derive exact error identities separating representation and coefficient recovery error and exposing approximation, ridge shrinkage and input-error propagation. The analysis yields a representation-error lower bound on recovery risk and a deterministic pointwise upper bound.
- **Method: learning compact continuous spaces through recovery.** LFP learns a shared, low-dimensional continuous basis through held-out prediction after coefficient fitting. It combines neural representation learning with a single weighted ridge solve per surface and financial regularization for shape control.
- **Evidence: predictive gains and their sources.** Experiments on four equity-index markets show substantial accuracy gains over GNO and HyperIV on SPX, low measured shape violations and millisecond construction. Cross-asset tests assess transfer, while controlled ablations isolate the benefits of recovery training and support the practical relevance of the theoretical distinction.

2 Related Work

Classical smoothing imposes structure through arbitrage-free surface families and calibration rules (Gatheral & Jacquier, 2014; Guo et al., 2016; Corbetta et al., 2019) or constrained splines (Fengler, 2009). Factor models exploit low dimensional volatility dynamics (Fengler et al., 2007), while sparse functional PCA predicts scores from irregular observations (Yao et al., 2005).

Neural approaches incorporate financial priors and shape penalties (Ackerer et al., 2020; Zheng et al., 2021), fit latent representations with variational autoencoders (Bergeron et al., 2022), or accelerate stochastic volatility calibration using random training grids (Baschetti et al., 2024). Operator Deep Smoothing (Wiedemann et al., 2025) maps irregular quote sets to surface predictions with a graph neural operator; HyperIV (Yang et al., 2025) generates surface network parameters from reference

quotes. Neural processes and multitask Gaussian processes also combine sparse quote conditioning with SABR priors (Zhuang & Wu, 2025; 2026). These methods already learn to infer surfaces from observations; LFP focuses on learning a compact linear function space for explicit coefficient recovery.

This formulation connects to gappy POD, which fits modal coefficients from incomplete samples (Everson & Sirovich, 1995), and Function Encoders, which learn continuous neural bases with least-squares encoding (Ingebrand et al., 2025; Low et al., 2025). Differentiable ridge solvers enable representation learning through fitted predictions (Bertinetto et al., 2019).

3 METHODOLOGY

3.1 SMOOTHING AS RECOVERY

We now formalize the recovery criterion introduced in Section 1. We compare the best approximation within a function space with the surface recovered from market quotes, using a common evaluation measure and weighted ridge coefficient fitting, the rule used by LFP.

Representation and recovery under market quotes. Let $\mathcal{X} \subset \mathbb{R}^2$ be a compact domain with nonempty interior, expressed in quote coordinates. A target surface $f : \mathcal{X} \to \mathbb{R}$ is continuous, and its quotes satisfy $y_i = f(x_i)+\xi_i$, where ξ_i denotes observation error. Write S for the indices of the quotes used in coefficient fitting and $n = |S| \geq 1$. We measure function error by $\|g\|_\nu^2 = \int_{\mathcal{X}} |g(x)|^2 \, \mathrm{d}\nu(x)$, where ν is a finite positive Borel measure common to the representations being compared. It specifies an evaluation region or a finite set of evaluation points.

Consider continuous functions $\phi = (\phi_1, \ldots, \phi_d)^\top$ and their span $\mathcal{S}_\phi = \{\phi(\cdot)^\top a : a \in \mathbb{R}^d\}$. Write $X_S = (x_i)_{i\in S}$. For weighted ridge fitting, let Φ_S have rows $\phi(x_i)^\top$, let $y_S = (y_i)_{i\in S}$, and let $W_S = \mathrm{diag}(w_i)_{i\in S}$ with $w_i \geq 0$. For $\lambda > 0$,

$$\begin{aligned} \widehat{a}_S &= (\Phi_S^\top W_S \Phi_S + \lambda I)^{-1} \Phi_S^\top W_S y_S, \\ \widehat{f}_{\mathcal{A},S}(x) &= \phi(x)^\top \widehat{a}_S. \end{aligned} \tag{3.1}$$

The recovery rule $\mathcal{A} = (\phi, \lambda, \mathcal{W})$ includes the weight rule $W_S = \mathcal{W}(X_S, y_S)$. Learned parameters and preprocessing are fixed when evaluating recovery. The joint task distribution $\mathbb{P}$ describes the surface, quote locations and observation errors, allowing the quote design to depend on the surface.

Definition 3.1 (Representation error and recovery risk)**.** Let $\Pi_{\mathcal{S}_\phi}$ be the orthogonal projection onto $\mathcal{S}_\phi$ in $L^2(\nu)$. For a fixed evaluation measure ν and recovery rule $\mathcal{A}$, define

$$\begin{aligned} \mathcal{E}_{\mathrm{rep}}(\mathcal{S}_\phi) &= \mathbb{E}_f \big\| f - \Pi_{\mathcal{S}_\phi} f \big\|_\nu^2, \\ \mathcal{R}_{\mathrm{rec}}(\mathcal{A}) &= \mathbb{E}_{\mathbb{P}} \big\| \widehat{f}_{\mathcal{A},S} - f \big\|_\nu^2, \end{aligned} \tag{3.2}$$

assuming these expectations are finite. The expectation over f uses the surface marginal of $\mathbb{P}$.

Representation error measures approximation with the whole function available; recovery risk measures prediction after fitting to quotes. The latter depends on the basis, weights, regularization and quote design. Since $\widehat{f}_{\mathcal{A},S} \in \mathcal{S}_\phi$, orthogonality gives

$$\big\| \widehat{f}_{\mathcal{A},S} - f \big\|_\nu^2 = \underbrace{\big\| f - \Pi_{\mathcal{S}_\phi} f \big\|_\nu^2}_{\text{representation error}} + \underbrace{\big\| \widehat{f}_{\mathcal{A},S} - \Pi_{\mathcal{S}_\phi} f \big\|_\nu^2}_{\text{coefficient recovery error}}. \tag{3.3}$$

Taking expectations yields $\mathcal{R}_{\mathrm{rec}}(\mathcal{A}) \geq \mathcal{E}_{\mathrm{rep}}(\mathcal{S}_\phi)$. The decomposition also holds conditionally for an empirical evaluation measure; its coordinates and weights then enter the joint expectation.

Separation of representation and recovery. Two representations can express the same surface but encode different links between observed and unobserved regions. Consider orthonormal smooth functions g_A, g_B, g_R supported on disjoint regions A, B, R, and set $u = (g_A + g_B)/\sqrt{2}$. Suppose the target has the form $f = cu$. The noiseless input includes quotes in B and R where the respective functions are nonzero, but none in A. Both

$$\mathcal{S}_1 = \mathrm{span}\{u, g_R\}, \qquad \mathcal{S}_2 = \mathrm{span}\{g_A, g_B\} \tag{3.4}$$

have dimension two and contain every target surface. In $\mathcal{S}_1$, the quote in B informs the coefficient governing both A and B. In $\mathcal{S}_2$, the coefficient of g_A is unconstrained by the input quotes and ridge fitting sets it to zero.

Proposition 3.2 (Separation of representation and recovery)**.** *For every integer $d \geq 2$ and every $M > 0$, there exist a distribution of smooth surfaces, a finite noiseless quote design, a common evaluation measure ν, two bases ϕ_1, ϕ_2 of dimension d orthonormal under ν, and a shared $\lambda > 0$ such that the corresponding rules in Eq.* (3.1)*, using the same quotes and unit weights, satisfy*

$$\begin{aligned} &\mathcal{E}_{\mathrm{rep}}(\mathcal{S}_{\phi_1}) = \mathcal{E}_{\mathrm{rep}}(\mathcal{S}_{\phi_2}) = 0, \\ &\mathcal{R}_{\mathrm{rec}}(\mathcal{A}_2) \geq M\, \mathcal{R}_{\mathrm{rec}}(\mathcal{A}_1) > 0. \end{aligned} \tag{3.5}$$

Appendix A.1 proves the result and computes the error at the omitted quote. Encoding joint variation over A and B lets a quote in one region inform the other. Historical quotes covering both regions can provide these relationships even when one region is absent from the current input.

We therefore select representations by prediction after quote fitting. The application also requires continuous evaluation at changing coordinates and inexpensive construction of new surfaces. Section 3.2 implements these criteria with shared continuous basis functions.

3.2 Learned Functional Projection

LFP learns basis functions across historical surfaces and fits coefficients to each new quote set. Training evaluates the representation through the resulting predictions.

Continuous representation and coefficient fitting. We parameterize $\phi_\theta : \mathcal{X} \to \mathbb{R}^d$ with a coordinate network, building on learned functional representations (Ingebrand et al., 2025). Following Wiedemann et al. (2025), we use $x = (r, z)$ with $r = \sqrt{\tau}$ and $z = \log(K/F)/\sqrt{\tau}$, where τ is time to maturity, K is strike and F is the forward price. A multilayer perceptron takes Fourier features of the normalized coordinates (Tancik et al., 2020) and outputs $d-1$ learned functions. We include a constant function, giving

$$\phi_\theta(x) = \big(1, \psi_{\theta,1}(x), \ldots, \psi_{\theta,d-1}(x)\big)^\top, \qquad \widehat{f}_S(x) = \phi_\theta(x)^\top \widehat{a}_S. \tag{3.6}$$

Here d counts all coefficients; Appendix B.1 specifies the network.

For each quote set, Eq. (3.1) computes $\widehat{a}_S$ using $\phi = \phi_\theta$ and a learned positive parameter $\lambda = \lambda(\alpha)$. All coefficients, including the constant, receive the same ridge penalty. One linear solve provides coefficients for all query locations. Its cost for n inputs and m queries is $O(nd^2 + d^3 + md)$, excluding basis evaluation; Appendix B.4 derives the full cost. Appendix A.3 relates this rule to kernel ridge regression with $k_\theta(x, x') = \phi_\theta(x)^\top \phi_\theta(x')$ and its function norm.

Learning the representation through recovery. Each episode partitions a historical quote pool D_t into nonempty input and query sets S and Q. We compute $W_{t,S}$ from the input quotes, solve for the coefficients on S, and predict on Q.

Following Wiedemann et al. (2025), we use vega-weighted relative root mean squared error (RWRMSE, abbreviated RW). For positive targets y_j, let $\delta_{t,Q} = \big((\widehat{f}_{t,S}(x_j) - y_j)/y_j\big)_{j \in Q}$ and $\Omega_{t,Q} = \mathrm{diag}(\omega_{t,j})_{j \in Q}$ collect relative errors and scoring weights. The loss is

$$\ell_Q(\widehat{f}_{t,S}, y_Q) = \frac{\|\Omega_{t,Q}^{1/2} \delta_{t,Q}\|_2}{\sqrt{|Q|}}. \tag{3.7}$$

Averaging over training surfaces and random splits gives the recovery objective

$$J_{\mathrm{rec}}(\theta, \alpha) = \mathbb{E}_{t,S,Q}\big[\ell_Q(\widehat{f}_{t,S}, y_Q)\big]. \tag{3.8}$$

The square root is taken per surface; scoring weights enter only the loss. For new surfaces, we fix θ and α and fit coefficients. Appendix B.2 specifies the weights and evaluation protocol.

Gradients pass through the solve to θ and α (Bertinetto et al., 2019), selecting the basis by prediction after quote fitting as motivated by Proposition 3.2. The financial penalties below augment this recovery objective.

Recovery error analysis. For a fixed representation, the following result separates approximation, regularization and input error in the predictor of Eq. (3.1).

Theorem 3.3 (Recovery error of weighted ridge)**.** *Fix ϕ, input coordinates, nonnegative diagonal weights W_S, and $\lambda > 0$. Let $y_S = f_S + \xi_S$, and define*

$$G = \Phi_S^\top W_S \Phi_S, \qquad H = G + \lambda I, \qquad \gamma = \lambda_{\min}(G). \tag{3.9}$$

For any $a^\circ \in \mathbb{R}^d$, put $h = f - \phi(\cdot)^\top a^\circ$ and $e_S = h_S + \xi_S$. Then, for every $x \in \mathcal{X}$,

$$\widehat{f}_S(x) - f(x) = -h(x) - \lambda \phi(x)^\top H^{-1} a^\circ + \phi(x)^\top H^{-1} \Phi_S^\top W_S e_S. \tag{3.10}$$

Consequently,

$$|\widehat{f}_S(x) - f(x)| \le |h(x)| + \|\phi(x)\|_2 \left(\frac{\lambda \|a^\circ\|_2}{\gamma + \lambda} + \frac{\|W_S^{1/2} e_S\|_2}{\sqrt{\gamma + \lambda}} \right). \tag{3.11}$$

Appendix A.2 gives the proof. The three terms describe the representation residual, ridge shrinkage and propagation of input errors. Choosing a° to represent the $L^2(\nu)$ projection links h to Eq. (3.3). Its input values enter e_S, so approximation affects both fitting and evaluation.

The matrix G couples the basis to quote design. Its interaction with residuals and regularization motivates selecting θ and λ jointly under the task's observation patterns.

Financial regularization. We apply financial penalties to the fitted surface $\widehat{f}_{t,S} = \phi_\theta(\cdot)^\top \widehat{a}_{t,S}$. Following Wiedemann et al. (2025), we combine butterfly and calendar hinge penalties with second derivative penalties in the z and r directions. With positive coefficients β_k, the full objective is

$$J_{\text{total}}(\theta, \alpha) = J_{\text{rec}}(\theta, \alpha) + \mathbb{E}_{t,S,\mathcal{G}} \left[\sum_{k \in \mathcal{K}} \beta_k \mathcal{L}_k(\widehat{f}_{t,S}; \mathcal{G}) \right]. \tag{3.12}$$

Here $\mathcal{K} = \{\text{but}, \text{cal}, z, r\}$ indexes the four terms on collocation grids $\mathcal{G}$. Appendix B.3 defines the penalties; Appendix C.2 gives their coefficients.

The full method, LFP-Arb, updates the shared representation through the fitted coefficients. Its predictor retains Eq. (3.1), so Theorem 3.3 applies.

4 Experiments

We evaluate LFP on held-out quote recovery, financial shape, cross-asset transfer and surface construction time. Objective ablations assess the benefits of recovery training and the practical relevance of the theoretical results. Accuracy and shape results for LFP variants and HyperIV are means over five training seeds, using the aggregation in Section 4.2. Tables mark the **best** and second-best results within each ranked column. Ranks use values displayed to four decimal places; ties share a rank.

4.1 Experimental setup

Data and preprocessing. We use daily closing option quotes from WRDS OptionMetrics for SPX, DJX, NDX and RUT over 2012–2021. Each asset–date pair defines one surface. The 2012–2020 observations form the development set, with 750 dates sampled for validation using a fixed split seed; 2021 is reserved for testing. The common test set contains 252 dates per asset. We retain AM-settled contracts, construct midpoint IVs from out-of-the-money quotes, and use the coordinates of Section 3.2 on $r \in [0.01, 1]$ and $z \in [-1.5, 0.5]$. Supervision and quote-error evaluation use actual quotes. Grids used for PCA preprocessing or financial checks do not supply ground-truth surfaces. Appendix C.1 specifies preprocessing and sample counts.

Baselines. Baselines cover parametric calibration (SSVI), fixed linear representations (PCA ridge), and neural surface prediction (GNO and HyperIV). SSVI is fitted per surface and PCA on training history. We use GNO's released SPX weights without retraining; HyperIV is retrained on the same daily data, splits and computing environment as LFP, using its authors' implementation. LFP denotes the full LFP-Arb model; LFP-9 is trained with HyperIV's reference-nine input rule.

Training and implementation. LFP is implemented in PyTorch (Paszke et al., 2019) and uses $d = 33$ coefficients, including the constant. Its general training episodes sample input fractions uniformly from $[0.3, 0.7]$, with at least 37 input quotes and one query quote. Training uses AdamW (Loshchilov & Hutter, 2019), batch size 64, an initial learning rate of 2×10^{-3}, and 300 epochs with 10 warm-up epochs followed by cosine decay. The experimental design uses five training seeds for LFP and HyperIV, with model selection based on validation data as detailed in Appendix C.2. Experiments run with NVIDIA GH200 GPUs (120 GB) and NVIDIA Grace CPUs (72 Arm Neoverse-V2 cores per CPU). Each model is trained on one GPU. Offline training cost is reported separately from the time to construct a new surface. For transfer from SPX, the frozen model is conditioned on target-asset quotes; target-trained LFP provides a separate historical-training reference.

Controlled variants. We compare recovery (C), the objective in Eq. (3.8), with full-pool reconstruction (A) and input reconstruction (B), which fit and score on D_t and S, respectively. The comparison fixes the architecture, dimension, training dates, optimization budget and penalty weights, with a common validation criterion based on held-out recovery. All three fit coefficients to the same input set at test time. A versus C compares the full representation-selection procedure, while B versus C also controls exposure to partial inputs.

4.2 Evaluation protocol

Available quotes and held-out evaluation. The split across dates is distinct from the input/query split within a surface. Under standard inputs (P1), five common random masks assign approximately half of each day's quotes to S and the rest to Q. All methods receive the same available pool; HyperIV selects its nine references from S only. Under reference-nine inputs (P2), all methods receive the same nine distinct contracts, selected near 7, 30 and 90 days using HyperIV's ATM, 25-delta call and 25-delta put reference rule. The remaining quotes form the common Q within P2. Since P1 and P2 have different query sets, their difference is not a pure input-count effect. The objective comparison uses 64 inputs, sampled after fixing 20% of each day's quotes as a common query set, with five input masks. Fitting weights depend only on S; scoring weights are common to all methods. Appendix C.3 gives the sampling rules and additional observation conditions.

Metrics. We report RW from Eq. (3.7), with scoring weights from Eq. (B.4), and MAPE:

$$\mathrm{MAPE} = |Q|^{-1} \sum\nolimits_{j \in Q} |\delta_j|, \qquad \delta_j = (\widehat{f}_S(x_j) - y_j)/y_j. \tag{4.1}$$

Bid–ask (BA) coverage is the fraction of predicted option prices within the corresponding observed bid–ask intervals, using the same pricing convention as IV inversion. Butterfly (Bfly) and calendar (Cal) violation rates are evaluated on an independent 25×81 grid over the modeling domain, at tolerance 10^{-3}. Calendar pairs hold $k = \log(K/F)$ fixed. All five metrics are reported as decimals. The shape metrics describe finite-grid violations.

For the representation comparison, let $\mathcal{E}_Q(a) = \ell_Q\big(\phi(\cdot)^\top a, y_Q\big)^2$ and define

$$E_{\mathrm{app}} = \min_a \mathcal{E}_Q(a), \qquad E_{\mathrm{rec}} = \mathcal{E}_Q(\widehat{a}_S), \qquad E_{\mathrm{gap}} = E_{\mathrm{rec}} - E_{\mathrm{app}}. \tag{4.2}$$

The unpenalized projection defining E_{app} uses query labels only for this diagnostic; deployed predictions use $\widehat{a}_S$. The decomposition corresponds to Eq. (3.3) on finite, noisy quotes, rather than directly measuring the unknown continuous target. Numerical details are in Appendix C.4.

Aggregation and uncertainty. Accuracy, shape and decomposition metrics are computed for each successful surface–mask evaluation, averaged over masks within each date, and then averaged with equal weight across dates. For independently trained models, the reported mean averages these seed-level performances, not their predictions. The 95% paired date-block bootstrap intervals describe uncertainty across test dates conditional on the trained models. Complete statistics and failure coverage accompany the corresponding results in Appendix D.

4.3 Results

Recovery on market quotes. LFP leads all baselines in held-out RW, MAPE and bid–ask coverage under both input protocols (Table 4.1). With standard inputs, it reduces RW by 39.3% relative

Table 4.1: Recovery on SPX from standard inputs and reference-nine quotes.

	a Standard inputs					**b** Reference nine				
Method	RW↓	MAPE↓	BA↑	Bfly↓	Cal↓	RW↓	MAPE↓	BA↑	Bfly↓	Cal↓
SSVI	0.0604	0.0424	0.0802	0.0012	**0.0000**	0.2128	0.1807	0.0108	**0.0000**	**0.0000**
PCA ridge	0.0091	0.0054	0.5650	0.1748	0.0047	0.1129	0.0794	0.0837	0.1614	0.0197
GNO	0.0145	0.0090	0.3671	0.0049	0.0030	0.3795	0.2874	0.0195	0.0491	0.0944
HyperIV	0.1076	0.0787	0.0385	**0.0000**	**0.0000**	0.1066	0.0781	0.0388	**0.0000**	**0.0000**
LFP	**0.0055**	**0.0034**	**0.7428**	0.0002	**0.0000**	**0.0444**	**0.0304**	**0.1512**	0.0005	**0.0000**

The LFP row uses LFP in **a** and LFP-9 in **b**.

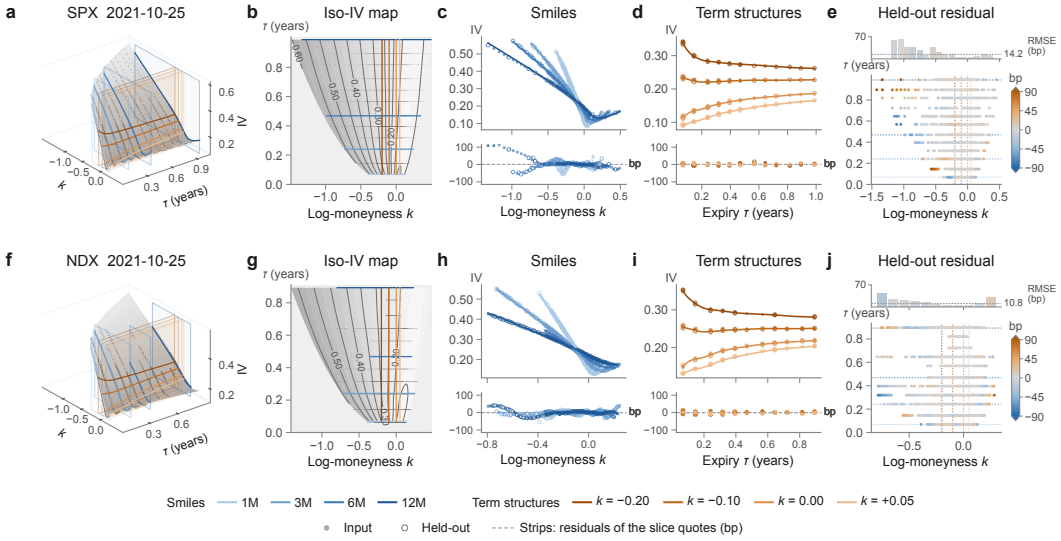

Figure 4.1: Volatility smoothing on SPX (top) and NDX (bottom) using the same SPX-trained LFP model. Residuals are in IV basis points. Bars above the residual maps show unweighted RMSE by log-moneyness bin; dashed lines indicate overall RMSE. Map colours saturate at ±90 basis points, as in the appendix.

to PCA ridge, the strongest baseline on quote accuracy, and by 61.8% relative to GNO. Bid–ask coverage improves by 17.8 and 37.6 percentage points, respectively. These gains accompany near-zero butterfly and calendar violation rates, although HyperIV has lower butterfly violation rates under both protocols.

The advantage persists under the common nine-contract input budget: LFP-9 reduces RW by 58.4% relative to HyperIV and by 60.7% relative to PCA ridge. It also achieves the highest bid–ask coverage, approximately 3.9 times that of HyperIV. Across both protocols, the improvements in RW, MAPE and BA over every baseline are significant (Table D.1). The recovery advantage also holds under random and coverage-based nine-quote locations: LFP-9 leads RW, MAPE and BA under all three location rules (Table D.2).

The SPX example in the first row of Figure 4.1 shows how this recovery translates into a continuous surface. LFP captures the pronounced strike skew and differing term structures while closely following held-out quotes along the smiles. Residuals are small near the money and larger in the wings, revealing spatial variation that aggregate errors alone do not show. Figures D.1–D.2 extend this view to ten SPX dates spanning the test year.

Generalization across assets. The SPX-trained representation transfers effectively to NDX and RUT (Table 4.2). LFP leads all baselines on RW, MAPE and bid–ask coverage for both assets. Relative to PCA ridge, it reduces RW by 47.1% on NDX and 34.8% on RUT, while increasing BA coverage by 20.0 and 12.3 percentage points, respectively. These improvements are significant across all three metrics. Appendix D.2 reports the complete P1 and P2 results, including shape metrics and

Table 4.2: Recovery across assets with standard inputs.

		a NDX			**b** RUT			**c** DJX		
Method	Training	RW↓	MAPE↓	BA↑	RW↓	MAPE↓	BA↑	RW↓	MAPE↓	BA↑
SSVI	Current day	0.0385	0.0253	0.2730	0.0507	0.0345	0.3483	0.1274	0.0789	0.6888
PCA ridge	SPX	0.0092	0.0058	0.6867	0.0117	0.0068	0.8009	**0.0488**	0.0249	**0.9569**
GNO	SPX	0.0198	0.0126	0.3948	0.0253	0.0153	0.5480	0.0982	0.0467	0.9274
HyperIV	SPX	0.1223	0.0817	0.0820	0.1223	0.0817	0.1487	0.5322	0.3876	0.4004
LFP	SPX	**0.0049**	**0.0031**	**0.8871**	**0.0076**	**0.0042**	**0.9244**	0.0533	**0.0248**	0.9523
LFP	Target	0.0044	0.0028	0.9190	0.0075	0.0041	0.9287	0.0448	0.0231	0.9508

The grey row uses target-asset training and is excluded from ranking.

Table 4.3: Training-objective ablation on SPX with 64 input quotes.

	a Quote recovery			**b** Error decomposition			**c** Stability		
Objective	RW↓	MAPE↓	BA↑	$E_{\text{app}}\downarrow$ (10^{-5})	$E_{\text{rec}}\downarrow$ (10^{-3})	$E_{\text{gap}}\downarrow$ (10^{-3})	λ (10^{-3})	γ (10^{-3})	$(\gamma+\lambda)^{-1/2}$
A: full-pool	0.0337	0.0123	0.5289	**2.1577**	1.7793	1.7577	1.2719	0.7071	22.5459
B: input	0.0266	0.0100	0.5780	2.3919	1.2058	1.1818	1.1700	9.0937	9.8693
C: recovery	**0.0236**	**0.0092**	**0.5837**	2.3931	**0.9621**	**0.9382**	1.1825	6.5412	11.3686

d Paired differences from recovery (C)

Contrast	ΔRW (10^{-3})	ΔE_{app} (10^{-6})	ΔE_{gap} (10^{-4})
A − C	$+10.0950$ $[+9.2940, +10.9264]$	-2.3542 $[-2.8865, -1.8214]$	$+8.1955$ $[+7.1596, +9.2278]$
B − C	$+2.9716$ $[+2.4946, +3.4470]$	-0.0124 $[-0.4624, +0.4083]$	$+2.4367$ $[+1.6965, +3.2487]$

Parenthesized powers of ten give column multipliers. λ is a mean; γ and $(\gamma+\lambda)^{-1/2}$ are medians. Brackets contain 95% paired date-block intervals. Best/second-best styling applies to performance and error columns.

paired intervals. The NDX example in the second row of Figure 4.1 illustrates this reuse: fitting new coefficients adapts the same basis to the asset's volatility level, smiles and term structures, with close agreement at held-out quotes. Figures D.3–D.4 provide further examples across assets and dates.

Target-asset training reduces RW by only 2.1% on RUT, with no significant change in MAPE or BA coverage. NDX benefits more from target-asset history, with a further 8.4% reduction in RW.

DJX marks a limit to this transfer advantage: PCA ridge achieves lower RW and higher BA coverage, while MAPE is nearly identical. The SPX-trained representation therefore does not dominate every target asset. Target-asset training lowers LFP's RW from 0.0533 to 0.0448, below PCA ridge's 0.0488, indicating that the training asset matters for recovery. Even with SPX training, LFP outperforms SSVI, GNO and HyperIV on all three quote metrics. Across all training sources, LFP also outperforms HyperIV in all 12 cross-asset directions under P1 (Table D.3).

Ablation study. Recovery training (C) outperforms both reconstruction objectives (Table 4.3). Relative to A and B, it reduces RW by 29.9% and 11.2%, respectively, with significant improvements in MAPE and bid–ask coverage as well. Appendix D.3 extends this comparison with input-fitting errors, shape metrics and paired intervals (Table D.10).

A achieves the lowest approximation error E_{app}, yet produces the largest recovery error E_{rec}. C instead reduces the coefficient recovery error E_{gap} by 46.6% relative to A. Across all three objectives, this gap accounts for 97–99% of the empirical recovery error, so the small approximation advantage of A is outweighed by its poorer recovery from sparse inputs. This reversal is consistent with Proposition 3.2.

B and C do not differ significantly in approximation error or input MAPE, but C reduces E_{gap} by 20.6%. Thus, evaluating the representation on held-out quotes brings an additional benefit beyond fitting coefficients to partial observations. The stability diagnostics complement this result. A has an

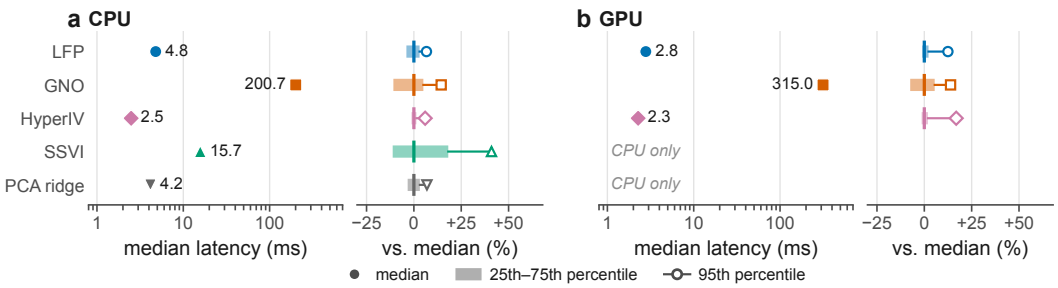

Figure 4.2: End-to-end latency under standard inputs on CPU and GPU. Right-hand panels show deviations from the median. Quantiles pool 24 test dates and 20 repetitions per date; HyperIV reference selection is included.

input-error bound factor $(\gamma + \lambda)^{-1/2}$ approximately twice that of C, consistent with the interaction between representation and quote design in Theorem 3.3. B has a slightly smaller factor than C despite its higher recovery error: this single quantity does not determine prediction quality.

Computational efficiency. LFP combines its recovery gains with millisecond-scale surface construction (Figure 4.2; Table D.11). Its median end-to-end latency yields approximately a 42-fold speedup over the public GNO implementation on CPU and a 113-fold speedup on GPU. On CPU, LFP is 3.3 times faster than SSVI. Its 95th-percentile latency remains within 7% of the median on CPU and 13% on GPU, indicating limited timing variability across the benchmark dates and repetitions. The median daily peak GPU allocation is 78.3% lower than GNO (Table D.14).

Offline training requires approximately 64 GPU-minutes per SPX LFP model (Table D.15). Appendix D.4 provides additional timing and resource comparisons, including common nine-reference inputs (Table D.12), fixed input and query counts (Table D.13), and scaling with these counts (Figure D.5).

5 Conclusion

This study develops a recovery-based approach to volatility smoothing, linking representation selection to the predictions obtained from available market quotes. Its theoretical contribution is to establish why approximation quality alone is insufficient: the separation result proves that representations with zero approximation error can have arbitrarily large ratios of recovery risk under the same observations and fitting rule. The weighted ridge analysis further explains how representation residuals, regularization and input errors affect recovery, exposing the interaction between the learned basis and quote design.

The practical contribution is LFP, which builds on Function Encoders to learn continuous basis functions through quote fitting and held-out prediction, while retaining a single coefficient solve for each new surface. Real-market experiments demonstrate substantial quote-accuracy gains over the evaluated baselines on SPX, NDX and RUT, alongside millisecond-scale construction and near-zero measured shape violations on SPX. The objective ablations connect these practical gains to the theoretical distinction: lower approximation error can accompany worse recovery, whereas training through held-out prediction improves the recovered surface. These findings support selecting representations by their usefulness under partial observation and show how this principle can yield an effective construction method.

LFP nevertheless remains sensitive to data coverage and observation distributions. The DJX results suggest that sparse quotes and cross-asset distribution differences may jointly limit the advantage of a shared representation, although LFP still outperforms the evaluated neural baselines there. Adaptive selection of training surfaces and input masks, ensembles with simpler models, and fallback rules for poorly informative quote sets are promising directions for improving robustness. Financial regularization also provides no guarantee of global absence of arbitrage; the reported violations are measured on finite grids. Stronger financial constraints would therefore complement improvements in recovery robustness.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing and improving the readability of author-written text, as well as with software code development and editing. We did not use generative AI tools for other research tasks requiring disclosure, and the remaining required disclosure categories are not applicable to this work.

All AI-assisted text was reviewed and revised by the authors to ensure accuracy and consistency with the intended scientific content. AI-assisted code was reviewed, tested, and validated by the authors for correctness. We take full responsibility for the final content of this work, including all text, claims, code, and other artifacts produced with the aid of generative AI.

REFERENCES

Damien Ackerer, Natasa Tagasovska, and Thibault Vatter. Deep smoothing of the implied volatility surface. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11552–11563, 2020.

Fabio Baschetti, Giacomo Bormetti, and Pietro Rossi. Deep calibration with random grids. *Quantitative Finance*, 24(9):1263–1285, 2024. doi: 10.1080/14697688.2024.2332375.

Maxime Bergeron, Nicholas Fung, John Hull, Zissis Poulos, and Andreas Veneris. Variational autoencoders: A hands-off approach to volatility. *The Journal of Financial Data Science*, 4(2): 125–138, 2022. doi: 10.3905/jfds.2022.1.093.

Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2019.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995. doi: 10.1137/0916069.

Jacopo Corbetta, Pierre Cohort, Ismail Laachir, and Claude Martini. Robust calibration and arbitrage-free interpolation of SSVI slices. *Decisions in Economics and Finance*, 42(2):665–677, 2019. doi: 10.1007/s10203-019-00249-8.

Richard Everson and Lawrence Sirovich. Karhunen–Loève procedure for gappy data. *Journal of the Optical Society of America A*, 12(8):1657–1664, 1995.

Matthias R. Fengler. Arbitrage-free smoothing of the implied volatility surface. *Quantitative Finance*, 9(4):417–428, 2009. doi: 10.1080/14697680802595585.

Matthias R. Fengler, Wolfgang K. Härdle, and Enno Mammen. A semiparametric factor model for implied volatility surface dynamics. *Journal of Financial Econometrics*, 5(2):189–218, 2007. doi: 10.1093/jjfinec/nbm005.

Jim Gatheral and Antoine Jacquier. Arbitrage-free SVI volatility surfaces. *Quantitative Finance*, 14 (1):59–71, 2014. doi: 10.1080/14697688.2013.819986.

Gaoyue Guo, Antoine Jacquier, Claude Martini, and Leo Neufcourt. Generalized arbitrage-free SVI volatility surfaces. *SIAM Journal on Financial Mathematics*, 7(1):619–641, 2016. doi: 10.1137/120900320.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2111.06377.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. URL https://arxiv.org/abs/1606.08415.

Tyler Ingebrand, Adam Thorpe, and Ufuk Topcu. Function encoders: A principled approach to transfer learning in Hilbert spaces. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 26464–26489, 2025. URL https://proceedings.mlr.press/v267/ingebrand25a.html.

Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A*, 374(2065):20150202, 2016. doi: 10.1098/rsta.2015.0202.

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.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Su Ann Low, Quentin Rommel, Kevin S. Miller, Adam J. Thorpe, and Ufuk Topcu. Function spaces without kernels: Learning compact Hilbert space representations. *arXiv preprint arXiv:2509.20605*, 2025. URL https://arxiv.org/abs/2509.20605.

Krithika Manohar, Bingni W. Brunton, J. Nathan Kutz, and Steven L. Brunton. Data-driven sparse sensor placement for reconstruction: Demonstrating the benefits of exploiting known patterns. *IEEE Control Systems Magazine*, 38(3):63–86, 2018. doi: 10.1109/MCS.2018.2810460. URL https://arxiv.org/abs/1701.07569.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

Dimitris N. Politis and Joseph P. Romano. A circular block-resampling procedure for stationary data. In Raoul LePage and Lynne Billard (eds.), *Exploring the Limits of Bootstrap*, pp. 263–270. John Wiley & Sons, 1992.

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

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*, volume 33, 2020.

Ruben Wiedemann, Antoine Jacquier, and Lukas Gonon. Operator deep smoothing for implied volatility. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=DPlUWG4WMw.

Yongxin Yang, Wenqi Chen, Chao Shu, and Timothy Hospedales. HyperIV: Real-time implied volatility smoothing. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70550–70564. PMLR, 2025. URL https://proceedings.mlr.press/v267/yang25d.html.

Fang Yao, Hans-Georg Müller, and Jane-Ling Wang. Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100(470):577–590, 2005. doi: 10.1198/016214504000001745.

Yu Zheng, Yongxin Yang, and Bowei Chen. Incorporating prior financial domain knowledge into neural networks for implied volatility surface prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3968–3975, 2021. doi: 10.1145/3447548.3467115.

Jirong Zhuang and Xuan Wu. Meta-learning neural process for implied volatility surfaces with SABR-induced priors. *arXiv preprint arXiv:2509.11928v3*, 2025. URL https://arxiv.org/abs/2509.11928v3.

Jirong Zhuang and Xuan Wu. SABR-informed multitask Gaussian process: A synthetic-to-real framework for implied volatility surface construction. *Journal of Computational Science*, 95: 102818, 2026. doi: 10.1016/j.jocs.2026.102818. URL https://arxiv.org/abs/2506.22888v2.

A Theoretical Analysis and Proofs

A.1 Separation of Representation and Recovery

For unit weights, the recovery rule used below is

$$\widehat{f}_{\mathcal{A},S}(x) = \phi(x)^\top (\Phi_S^\top \Phi_S + \lambda I)^{-1} \Phi_S^\top y_S, \tag{A.1}$$

where Φ_S contains the basis values at the input quotes. For a common evaluation measure ν, the two quantities being compared are

$$\mathcal{E}_{\mathrm{rep}}(\mathcal{S}_\phi) = \mathbb{E}_f \|f - \Pi_{\mathcal{S}_\phi} f\|_\nu^2, \qquad \mathcal{R}_{\mathrm{rec}}(\mathcal{A}) = \mathbb{E}_{\mathbb{P}} \|\widehat{f}_{\mathcal{A},S} - f\|_\nu^2, \tag{A.2}$$

where $\Pi_{\mathcal{S}_\phi}$ denotes orthogonal projection in $L^2(\nu)$. The proposition is restated below.

Proposition 3.2 (Separation of representation and recovery)**.** *For every integer $d \geq 2$ and every $M > 0$, there exist a distribution of smooth surfaces, a finite noiseless quote design, a common evaluation measure ν, two bases ϕ_1, ϕ_2 of dimension d orthonormal under ν, and a shared $\lambda > 0$ such that the corresponding rules in Eq.* (3.1)*, using the same quotes and unit weights, satisfy*

$$\begin{aligned} &\mathcal{E}_{\mathrm{rep}}(\mathcal{S}_{\phi_1}) = \mathcal{E}_{\mathrm{rep}}(\mathcal{S}_{\phi_2}) = 0, \\ &\mathcal{R}_{\mathrm{rec}}(\mathcal{A}_2) \geq M\, \mathcal{R}_{\mathrm{rec}}(\mathcal{A}_1) > 0. \end{aligned} \tag{3.5}$$

Proof of Proposition 3.2. Fix $d \geq 2$ and $M > 0$. Choose $d+1$ distinct interior points of a compact domain $\mathcal{X} \subset \mathbb{R}^2$, denoted by $x_A, x_B, x_R, x_{P_1}, \ldots, x_{P_{d-2}}$, and let

$$\nu = \frac{1}{d+1} \left(\delta_{x_A} + \delta_{x_B} + \delta_{x_R} + \sum_{j=1}^{d-2} \delta_{x_{P_j}} \right). \tag{A.3}$$

For each point choose a smooth bump function supported in a neighborhood that does not intersect the other neighborhoods. Denote these functions by $g_A, g_B, g_R, g_{P_1}, \ldots, g_{P_{d-2}}$, scaled so that each takes the value $\sqrt{d+1}$ at its corresponding point. They form an orthonormal family in $L^2(\nu)$.

Let $u = (g_A + g_B)/\sqrt{2}$ and $f = cu$, where c takes the values -1 and 1 with equal probability. Use noiseless observations at every point except x_A, so that $n = d$ and $W_S = I_d$. Define

$$\phi_1 = (u, g_R, g_{P_1}, \ldots, g_{P_{d-2}})^\top, \qquad \phi_2 = (g_A, g_B, g_{P_1}, \ldots, g_{P_{d-2}})^\top. \tag{A.4}$$

For $d = 2$ the padding functions are absent. Both bases are orthonormal under ν, have dimension d, and contain u in their spans. Hence both representation errors are zero.

Write $q = d + 1$. The two observation Gram matrices are

$$G_1 = \Phi_{1,S}^\top \Phi_{1,S} = \mathrm{diag}(q/2, q, \ldots, q), \qquad G_2 = \Phi_{2,S}^\top \Phi_{2,S} = \mathrm{diag}(0, q, \ldots, q). \tag{A.5}$$

In the first basis the target coefficient vector is $(c, 0, \ldots, 0)^\top$. Ridge fitting multiplies its first coefficient by $(q/2)/(q/2 + \lambda)$. Since $\mathbb{E}[c^2] = 1$ and the basis is orthonormal,

$$\mathcal{R}_{\mathrm{rec}}(\mathcal{A}_1) = \left(\frac{\lambda}{q/2 + \lambda} \right)^2. \tag{A.6}$$

In the second basis the target coefficients are $(c/\sqrt{2}, c/\sqrt{2}, 0, \ldots, 0)^\top$. The first fitted coefficient is zero and the second is multiplied by $q/(q + \lambda)$. Consequently,

$$\mathcal{R}_{\mathrm{rec}}(\mathcal{A}_2) = \frac{1}{2} + \frac{1}{2} \left(\frac{\lambda}{q + \lambda} \right)^2. \tag{A.7}$$

For every $\lambda > 0$ the first risk is positive, and

$$\frac{\mathcal{R}_{\mathrm{rec}}(\mathcal{A}_2)}{\mathcal{R}_{\mathrm{rec}}(\mathcal{A}_1)} \geq \frac{(q/2 + \lambda)^2}{2\lambda^2} \geq \frac{q^2}{8\lambda^2}. \tag{A.8}$$

Choosing the shared value $\lambda = q/(4\sqrt{\max\{M, 1\}})$ makes this ratio at least $2 \max\{M, 1\} \geq M$. $\square$

The construction uses smooth functions with specified point values; all projections and norms are taken under the finite measure ν. It also describes recovery at the omitted quote. Under evaluation at x_A alone, the two mean squared prediction errors are

$$\begin{aligned}\mathbb{E}\big|\widehat{f}_{\mathcal{A}_1,S}(x_A) - f(x_A)\big|^2 &= \frac{d+1}{2}\left(\frac{\lambda}{(d+1)/2+\lambda}\right)^2,\\ \mathbb{E}\big|\widehat{f}_{\mathcal{A}_2,S}(x_A) - f(x_A)\big|^2 &= \frac{d+1}{2}.\end{aligned} \tag{A.9}$$

An affine version of the construction gives positive target surfaces. Choose a fixed known constant $m > \|u\|_\infty$ and use $f = m + cu$, fit to $y_S - m$, and return $m + \phi(x)^\top \widehat{a}_S$. The candidate classes are $m + \mathcal{S}_{\phi_j}$, and the errors are unchanged.

A.2 Recovery Error of Weighted Ridge

For the input matrix Φ_S, weights W_S and parameter λ, the fitted coefficients and surface are

$$\widehat{a}_S = (\Phi_S^\top W_S \Phi_S + \lambda I)^{-1} \Phi_S^\top W_S y_S, \qquad \widehat{f}_S(x) = \phi(x)^\top \widehat{a}_S. \tag{A.10}$$

The result below states both the exact error identity and its pointwise bound.

Theorem 3.3 (Recovery error of weighted ridge)**.** *Fix ϕ, input coordinates, nonnegative diagonal weights W_S, and $\lambda > 0$. Let $y_S = f_S + \xi_S$, and define*

$$G = \Phi_S^\top W_S \Phi_S, \qquad H = G + \lambda I, \qquad \gamma = \lambda_{\min}(G). \tag{3.9}$$

For any $a^\circ \in \mathbb{R}^d$, put $h = f - \phi(\cdot)^\top a^\circ$ and $e_S = h_S + \xi_S$. Then, for every $x \in \mathcal{X}$,

$$\widehat{f}_S(x) - f(x) = -h(x) - \lambda \phi(x)^\top H^{-1} a^\circ + \phi(x)^\top H^{-1} \Phi_S^\top W_S e_S. \tag{3.10}$$

Consequently,

$$|\widehat{f}_S(x) - f(x)| \le |h(x)| + \|\phi(x)\|_2 \left(\frac{\lambda \|a^\circ\|_2}{\gamma + \lambda} + \frac{\|W_S^{1/2} e_S\|_2}{\sqrt{\gamma + \lambda}}\right). \tag{3.11}$$

Proof of Theorem 3.3. The assumptions imply $G \succeq 0$ and $H \succeq \lambda I \succ 0$. Substituting $y_S = \Phi_S a^\circ + h_S + \xi_S = \Phi_S a^\circ + e_S$ into the normal equations gives

$$H \widehat{a}_S = G a^\circ + \Phi_S^\top W_S e_S. \tag{A.11}$$

Since $G = H - \lambda I$, it follows that

$$\widehat{a}_S - a^\circ = -\lambda H^{-1} a^\circ + H^{-1} \Phi_S^\top W_S e_S. \tag{A.12}$$

Multiplying by $\phi(x)^\top$ and using $f(x) = \phi(x)^\top a^\circ + h(x)$ proves Eq. (3.10).

To bound the two coefficient terms, set $B = W_S^{1/2} \Phi_S$, so $G = B^\top B$. The least eigenvalue of H is $\gamma + \lambda$, and hence $\|H^{-1}\|_2 = (\gamma + \lambda)^{-1}$. Moreover,

$$\|H^{-1} B^\top\|_2^2 = \max_{g \in \mathrm{spec}(G)} \frac{g}{(g+\lambda)^2} \le \frac{1}{\gamma + \lambda}. \tag{A.13}$$

Indeed, $g/(g+\lambda)^2 \le 1/(g+\lambda) \le 1/(\gamma+\lambda)$ for every eigenvalue $g \ge \gamma \ge 0$. Using $\Phi_S^\top W_S e_S = B^\top W_S^{1/2} e_S$ in Eq. (A.12) therefore yields

$$\|\widehat{a}_S - a^\circ\|_2 \le \frac{\lambda \|a^\circ\|_2}{\gamma + \lambda} + \frac{\|W_S^{1/2} e_S\|_2}{\sqrt{\gamma + \lambda}}. \tag{A.14}$$

Finally, $|\widehat{f}_S(x) - f(x)| \le |h(x)| + \|\phi(x)\|_2 \|\widehat{a}_S - a^\circ\|_2$ proves Eq. (3.11). □

The argument includes singular observation Gram matrices and zero weights. All quantities are fixed within the proof, so the same identity applies to each realization of a data dependent weight rule. To specify the projection as a continuous function, define

$$C_\nu = \int_{\mathcal{X}} \phi(x)\phi(x)^\top \,\mathrm{d}\nu(x), \qquad b_\nu = \int_{\mathcal{X}} \phi(x) f(x) \,\mathrm{d}\nu(x), \qquad a^\circ = C_\nu^\dagger b_\nu, \tag{A.15}$$

where $\dagger$ denotes the Moore–Penrose inverse. If $v \in \ker C_\nu$, then $v^\top \phi = 0$ almost everywhere under ν, so $v^\top b_\nu = 0$. Thus b_ν lies in the range of C_ν, and $C_\nu a^\circ = b_\nu$. It follows that $\phi(\cdot)^\top a^\circ$ represents $\Pi_{\mathcal{S}_\phi} f$ and h is orthogonal to $\mathcal{S}_\phi$. This gives Eq. (3.3), since $\widehat{f}_S - \phi(\cdot)^\top a^\circ$ belongs to that space. For an empirical measure, different continuous functions can represent the same $L^2(\nu)$ projection. Equation (A.15) selects one representative and defines its values at all input and query locations. The two coefficient error terms in Eq. (3.10) need not be orthogonal.

A.3 Kernel Interpretation and Function Regularization

Finite features induce kernels and function norms (Rasmussen & Williams, 2006, Chapters 2 and 6); learned versions are considered by Low et al. (2025). We apply this construction to Eq. (3.1).

Proposition A.1 (Kernel form of weighted ridge)**.** *Fix a continuous feature map $\phi_\theta : \mathcal{X} \to \mathbb{R}^d$, a set of n input coordinates, nonnegative diagonal weights W_S and $\lambda > 0$. Define*

$$k_\theta(x, x') = \phi_\theta(x)^\top \phi_\theta(x'), \qquad K_S = \Phi_S \Phi_S^\top, \qquad k_S(x) = \Phi_S \phi_\theta(x). \tag{A.16}$$

Then k_θ is a continuous positive semidefinite kernel, its Gram matrices have rank at most d, and the predictor in Eq. (3.1) *satisfies*

$$\widehat{f}_S(x) = k_S(x)^\top W_S^{1/2} \big(W_S^{1/2} K_S W_S^{1/2} + \lambda I_n\big)^{-1} W_S^{1/2} y_S. \tag{A.17}$$

Let $\mathcal{H}_\theta = \{\phi_\theta(\cdot)^\top a : a \in \mathbb{R}^d\}$, equipped with the feature norm

$$\|g\|_{\mathcal{H}_\theta}^2 = \min_{\{a:\, \phi_\theta(x)^\top a = g(x) \text{ for all } x \in \mathcal{X}\}} \|a\|_2^2. \tag{A.18}$$

It is the reproducing kernel Hilbert space of k_θ. With $g_S = (g(x_i))_{i \in S}$, the predictor uniquely solves

$$\underset{g \in \mathcal{H}_\theta}{\text{minimize}} \quad \|W_S^{1/2}(g_S - y_S)\|_2^2 + \lambda \|g\|_{\mathcal{H}_\theta}^2. \tag{A.19}$$

Proof. Set $B = W_S^{1/2} \Phi_S$. The matrix identity

$$(B^\top B + \lambda I_d)^{-1} B^\top = B^\top (B B^\top + \lambda I_n)^{-1} \tag{A.20}$$

follows by multiplying by the invertible factors. Substitution into Eq. (3.1) gives Eq. (A.17). The feature representation gives continuity, positivity and the rank bound.

To establish the function space statement, let $N = \{a : \phi_\theta(x)^\top a = 0 \text{ for all } x \in \mathcal{X}\}$. Every $g \in \mathcal{H}_\theta$ has a unique representing vector $a_g \in N^\perp$ of minimum Euclidean norm. Define $\langle g, h \rangle_{\mathcal{H}_\theta} = a_g^\top a_h$. Since $\phi_\theta(x) \in N^\perp$, the function $k_\theta(\cdot, x)$ is represented by $\phi_\theta(x)$, and $\langle g, k_\theta(\cdot, x) \rangle_{\mathcal{H}_\theta} = g(x)$. This gives the reproducing property and Eq. (A.18).

The coefficient ridge objective has a unique minimizer because $\lambda > 0$. Among all coefficient vectors representing the same function, the fitting loss is unchanged and the penalty selects a_g. Minimizing first within each such class and then over functions gives Eq. (A.19). Its positive squared norm penalty makes the function objective strictly convex, so the function solution is unique. □

The kernel values in Eq. (A.17) connect the input and query locations. Learning θ changes these values and the function norm in Eq. (A.18). This norm generally differs from the evaluation norm $\|\cdot\|_\nu$. A change of basis within the same span can therefore alter recovery if the coefficient penalty is kept as λI. This explains the use of the complete rule $\mathcal{A}$ in Definition 3.1. The coefficient formulation computes this predictor by solving a $d \times d$ system.

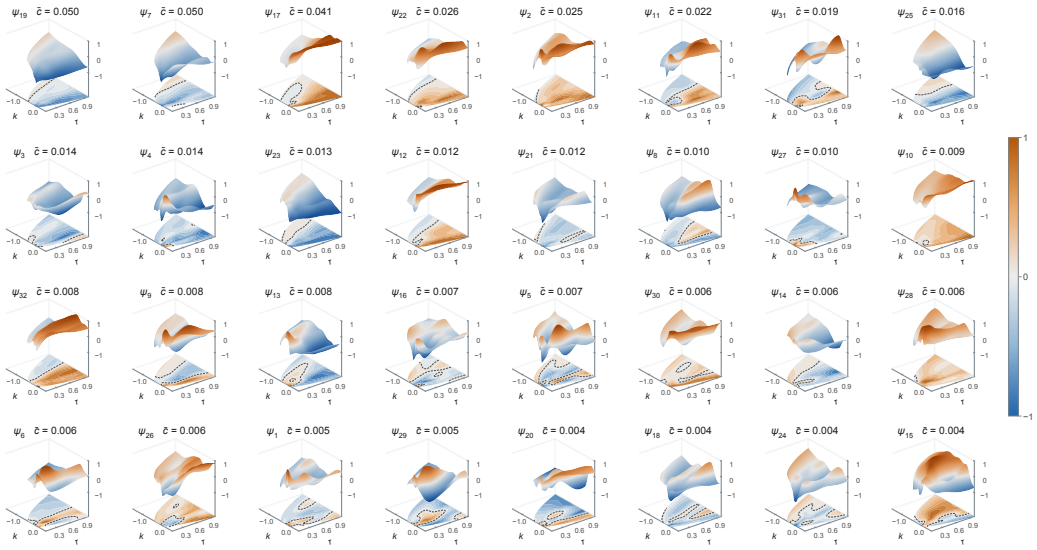

Figure B.1: Learned SPX basis functions, normalized by their maximum absolute value on the displayed domain and ordered by mean vega-weighted RMS contribution $\bar{c}$ on validation surfaces. The constant basis is omitted.

B LFP Implementation Details

B.1 Continuous Basis and Coefficient Solver

For $x = (r, z)$ with $r = \sqrt{\tau}$ and $z = \log(K/F)/\sqrt{\tau}$, we normalize

$$\widetilde{x}_j = \frac{x_j - (a_j + b_j)/2}{(b_j - a_j)/2}, \tag{B.1}$$

where $[a_j, b_j]$ is its modeling interval. For a fixed random matrix $B \in \mathbb{R}^{p\times 2}$, the network input is

$$\eta(x) = \big(\sin(B\widetilde{x}), \cos(B\widetilde{x})\big), \qquad B_{ij} \sim \mathcal{N}(0, \sigma_B^2). \tag{B.2}$$

The matrix B is sampled once and stored with the model. The perceptron outputs $d-1$ learned functions; prepending a constant gives d coefficients. The selected dimensions and training settings are in Appendix C.2. We parameterize the ridge scale as $\lambda(\alpha) = \mathrm{softplus}(\alpha) + 10^{-12}$ and apply the same coefficient penalty to every basis function, including the constant.

The output layer is linear, and predictions retain $\widehat{f}_S = \phi_\theta^\top \widehat{a}_S$. We form $G = \Phi^\top W \Phi$ without division by the input count and differentiate through a linear solve, without forming an inverse. The positive ridge scale makes $G + \lambda I$ invertible in exact arithmetic even when G is singular.

Basis examples. Figure B.1 shows the learned functions underlying the SPX representation.

B.2 Quote Weights and Episodic Learning

For positive observed volatilities, define the vega factor

$$v_{t,i} = r_i \varphi\left(-\frac{z_i}{y_i} + \frac{r_i y_i}{2}\right), \qquad \overline{v}_{t,A} = \frac{1}{|A|}\sum_{i\in A} v_{t,i}, \tag{B.3}$$

where φ is the standard normal density and A is a nonempty quote set. The domain has $r_i > 0$, so these vega factors and their means are positive. After splitting D_t into S and Q, we use

$$w_{t,i}^{\mathrm{fit}} = \max\left\{\frac{v_{t,i}}{\overline{v}_{t,S}}, 1\right\}, \quad i \in S, \qquad \omega_{t,j} = \max\left\{\frac{v_{t,j}}{\overline{v}_{t,D_t}}, 1\right\}, \quad j \in Q. \tag{B.4}$$

The fitting matrix is $W_{t,S} = \mathrm{diag}(w^{\mathrm{fit}}_{t,i})_{i\in S}$. The fitting weights use S alone. Scoring retains the quote pool normalization of Wiedemann et al. (2025), with the same pool and weights for all methods. These weights enter the loss, not the predictor. The ratios are computed using log vega and log-sum-exp normalization to avoid numerical underflow.

The date split, sampling settings and model-selection rules are in Appendices C.1–C.3.

The inner solve remains Eq. (3.1); relative errors and financial penalties train the shared parameters through it. Squaring Eq. (3.7) gives a sum of squared quote errors weighted by $\omega_{t,j}/(|Q|y_j^2)$. The training criterion averages the square roots over surfaces, whereas Section 3.1 studies squared function error against f. Theorem 3.3 describes the fitted predictor for either training criterion.

B.3 Financial and Smoothness Penalties

Write $v(r,z) = \widehat{f}_{t,S}(r,z)$. We use discrete penalties for butterfly and calendar conditions. On the butterfly grid, let $D_z v$ and $D_z^2 v$ denote finite differences. With $\widetilde{v} = \max\{v, 0.005\}$, set

$$d_\pm = -\frac{z}{\widetilde{v}} \pm \frac{r\widetilde{v}}{2}, \qquad b(v) = (1 + d_+ D_z v)(1 + d_- D_z v) + v D_z^2 v. \tag{B.5}$$

The floor applies only inside $d_\pm$; it does not transform the predictor. For a calendar pair with $r_+ > r$, the coordinate at the second maturity is $z_+ = rz/r_+$, so the two locations have the same log moneyness. The condition is

$$c(v; r, z, r_+) = \frac{v(r_+, rz/r_+)}{\max\{v(r,z), 0.001\}} - \frac{r}{r_+}. \tag{B.6}$$

When both surface values are positive and the denominator floor is inactive, $c \geq 0$ is equivalent to nondecreasing total variance along the pair.

With $[a]_+ = \max\{a, 0\}$, the two financial penalties are

$$\mathcal{L}_{\mathrm{but}} = \frac{1}{|\mathcal{G}_{\mathrm{but}}|} \sum_{x\in\mathcal{G}_{\mathrm{but}}} [-b(v)(x) - \varepsilon_{\mathrm{but}}]_+, \qquad \mathcal{L}_{\mathrm{cal}} = \frac{1}{|\mathcal{G}_{\mathrm{cal}}|} \sum_{p\in\mathcal{G}_{\mathrm{cal}}} [-c(v;p) - \varepsilon_{\mathrm{cal}}]_+, \tag{B.7}$$

where $\mathcal{G}_{\mathrm{cal}}$ collects calendar pairs. The selected tolerances and grid spacings are in Appendix C.2. Values below $-\varepsilon$ incur a penalty. On $\mathcal{G}_{\mathrm{but}}$, the smoothness penalties are

$$\mathcal{L}_z = \left[\frac{1}{|\mathcal{G}_{\mathrm{but}}|} \sum_{x\in\mathcal{G}_{\mathrm{but}}} |D_z^2 v(x)|^2\right]^{1/2}, \qquad \mathcal{L}_r = \left[\frac{1}{|\mathcal{G}_{\mathrm{but}}|} \sum_{x\in\mathcal{G}_{\mathrm{but}}} |D_r^2 v(x)|^2\right]^{1/2}. \tag{B.8}$$

Each square root is computed per surface. All terms are averaged over the batch, retaining gradients through coefficient fitting and basis evaluation.

The butterfly grid is rectilinear in (r, z). Calendar pairs connect consecutive r values and share an initial z axis. Grid axes start at the lower modeling limits and stop before the upper limits. The second row of each calendar pair is rescaled as in Eq. (B.6). The first difference at an interior grid point is the average of the forward and backward differences; boundary points use a one sided difference. Second differences apply this same first difference operator twice.

Grid evaluation. Grids are sampled independently of quote values and shared within a batch; basis evaluations are reused. Validation grids have fixed seeds. Evaluation uses separate grids to report prediction positivity and discrete condition violations at specified tolerances.

B.4 Computational Complexity

Let $n = |S|$, let m be the number of query coordinates, and let C_ϕ be the cost of evaluating all d basis functions at one coordinate. Forming $\Phi_S \in \mathbb{R}^{n\times d}$ costs $O(nC_\phi)$. Row weighting is $O(nd)$, forming $G = \Phi_S^\top W_S \Phi_S$ is $O(nd^2)$, and forming the right hand side $\Phi_S^\top W_S y_S$ is $O(nd)$. A dense factorization and solve of the $d \times d$ regularized system costs $O(d^3)$. Query basis values cost $O(mC_\phi)$ and their multiplication by the fitted coefficients costs $O(md)$. Thus the cost for one surface is

$$T(n, m, d) = O\big((n+m)C_\phi + nd^2 + d^3 + md\big). \tag{B.9}$$

Table C.1: Quotes retained by each preprocessing step, 2012–2021.

Asset	Raw rows	AM-settled	Out of the money	In domain	Share (%)
SPX	25278900	8500769	4249951	2885807	11.4
DJX	2014579	2014579	1007286	692848	34.4
NDX	13584718	8971266	4485597	3397858	25.0
RUT	11434802	5164861	2582418	1841603	16.1

Table C.2: Test quote pools after coordinate-domain restriction. All four assets cover 4 January to 31 December 2021. Quote-count quartiles are computed across dates.

Asset	Dates	Median quotes	Lower quartile	Upper quartile
SPX	252	2406.5	2163.75	2512.25
DJX	252	309.0	292.00	328.00
NDX	252	2427.5	2027.50	2741.50
RUT	252	766.0	716.00	824.25

The fitted coefficients are reused for all queries on the same surface. For hidden width h, L hidden layers, and p Fourier frequencies, the dense coordinate network has $C_\phi = O(p+ph+(L-1)h^2+hd)$ for $L \geq 1$. Materializing the basis matrices requires $O((n+m)d+d^2)$ storage, excluding model parameters and network activations. Query batching reduces the md term.

For an episode with g total collocation locations, penalty evaluation adds $O(gC_\phi + gd + g)$ for basis values, surface values and finite difference operations. Shared grids allow basis evaluations to be reused across the surfaces in a batch. Differentiating through the dense solve and network has the same asymptotic arithmetic order as their forward computations, with additional storage for automatic differentiation. These are per episode costs; training over historical surfaces is reported separately from constructing a surface with fixed parameters. Runtime measurement conventions are in Appendix C.5, and empirical cost and resource results are in Appendix D.4.

C Experimental Setup and Evaluation Protocols

C.1 Data and Preprocessing

Source and surface construction. The WRDS OptionMetrics extracts cover 3 January 2012 to 31 December 2021. We use the SPX, DJX, NDX and RUT option chains and retain contracts marked as AM-settled. The loader converts strike prices from their stored units, discards missing required fields, and requires positive time to maturity and midpoint price. Maturity uses a 365-day year. For each date and expiry, a regression of call-minus-put midpoint prices on strike estimates the forward. We retain calls with $K > F$ and puts with $K \leq F$, and invert midpoint prices using the implementation's zero-rate Black–Scholes convention with the forward as the underlying input. Predictions are converted back to prices with the same convention for BA coverage. The coordinate transformation and domain are those of Section 4.1. Beyond positive asks, successful IV inversion and this domain, no further quote-quality filter is applied. Table C.1 counts the quotes retained by each step over 2012–2021. The out-of-the-money step keeps one quote per strike and maturity, so it roughly halves the rows. The completeness, positive-maturity, positive-midpoint, IV-inversion and positive-ask checks remove at most seven quotes per asset and are not shown separately.

Date splits and quote counts. The common split reserves 2021 for testing. From dates before 2021, 750 are selected without replacement for validation using split seed 42; the remaining dates are used for training. The source data documentation records 1,515 training, 750 validation and 252 test dates per asset before episode-specific eligibility checks. All episodes from a surface remain in its assigned split. Table C.2 reports test-pool quote counts from the fixed test quote pools. These are input-pool statistics, not counts of successful model predictions.

Table C.3: LFP architecture and training configuration.

Quantity	Setting
Coordinate intervals	$r \in [0.01, 1]$, $z \in [-1.5, 0.5]$
Fourier features	$p = 32$, $\sigma_B = 4$
Hidden network	Four layers of width 128, GELU (Hendrycks & Gimpel, 2016)
Basis dimension	32 learned outputs plus a constant; $d = 33$
Ridge initialization	$\mathrm{softplus}(\alpha_0) = 10^{-3}$
Optimizer	AdamW; weight decay 10^{-6}
Learning rate and batch size	2×10^{-3}; 64 surfaces
Training budget	300 epochs; 10 warm-up epochs, then cosine decay
Gradient clipping	Global ℓ_2 norm capped at 10
Penalty coefficients	$(\beta_{\mathrm{but}}, \beta_{\mathrm{cal}}, \beta_z, \beta_r) = (10, 10, 0.01, 0.01)$
Training seeds	$0, 1, 2, 3, 4$
Model arithmetic	Single precision

C.2 Model Configurations and Training

LFP configuration. Table C.3 gives the LFP architecture and training configuration. The network and solver are defined in Appendix B.1.

For general LFP training, a quote pool of size N_t is randomly permuted and split with

$$n_S = \min\{N_t - 1, \max\{d + 4, \mathrm{round}(N_t u)\}\}, \qquad u \sim \mathrm{Uniform}(0.3, 0.7). \tag{C.1}$$

Pools with fewer than $d+5$ quotes are excluded from these training episodes. The first n_S quotes form S and the rest form Q. Validation uses an input fraction of 0.5 and fixed random seeds. LFP-9 instead uses the reference selection in Appendix C.3, requiring nine distinct references and a nonempty query set. The training minimum for general LFP does not restrict the ridge predictor, which is defined for any nonempty input.

The butterfly training grid uses a z spacing of 0.01 and an r spacing drawn uniformly from $[0.075, 0.125]$. Calendar pairs use an r spacing of 0.02 and a z spacing drawn from $[0.075, 0.125]$. Both penalty tolerances are 10^{-3}. Validation grids use fixed seeds. The quote-loss coefficient is one.

Objective controls and model selection. For the A/B/C variants of Section 4.1, the common model-selection criterion adds the financial and smoothness penalties to the held-out quote loss after fitting on validation inputs.

Baselines. HyperIV uses the reference-nine rule, batch size 128 and Adam (Kingma & Ba, 2015) with an initial learning rate of 10^{-3}, cosine-annealed to 10^{-5} over 500 epochs. Its specified objective combines its prediction, calendar, butterfly and density-mass terms, with 1,024 sampled query points per surface. Validation selects the model without using test dates. We use the authors' implementation, with a width-128 set encoder containing two Transformer layers and two attention heads per layer. Mean pooling and a linear output generate the 337 parameters of a 2–16–16–1 coordinate network, with tanh hidden activations and a softplus output.

GNO uses the public SPX model with 102,529 parameters, without retraining. Its released inference configuration selects neighbors within distance 0.3 in r and thins them to at most 35 per node. SSVI is calibrated on the available quotes for each surface. We use the power-law form with $\varphi(\theta) = \eta\theta^{-\gamma}(1+\theta)^{\gamma-1}$ (Gatheral & Jacquier, 2014). The ATM total variance $\theta(\tau)$ is nondecreasing and piecewise linear between observed maturities, zero at $\tau = 0$, and extended beyond the longest maturity with its last slope; this extension was selected on validation data. L-BFGS-B (Byrd et al., 1995) minimizes the vega-weighted squared total-variance error plus $100\max\{\eta(1+|\rho|) - 4, 0\}^2$, with $\rho \in [-0.999, 0.999]$, $\eta \in (0, 10]$ and $\gamma \in [0, 1]$, starting from $(\rho, \eta, \gamma) = (-0.3, 1, 0.5)$ and the ATM variances. Calibration requires two maturities and at least as many quotes as parameters; other input sets are recorded as not applicable. PCA learns its mean and components from historical training surfaces represented on a grid, then fits coefficients at the actual input coordinates. It retains 32 components on a 41×81 grid over the domain. Training surfaces are interpolated linearly, with nearest-quote values outside their convex hull; the mean and components are interpolated bilinearly at input and query coordinates. The mean is not penalized. The ridge is 10^{-4} for P1 and the other general

Table C.4: Test dates eligible under each observation protocol. These counts precede model-specific prediction or calibration failures.

Asset	P1	P2	P3	64 inputs
SPX	252	252	252	252
DJX	252	252	252	252
NDX	252	252	252	252
RUT	252	252	252	252

protocols and 10^{-5} for reference-nine inputs. The supplementary fixed-spline representation is a tensor product of four cubic B-splines in r and eight in z with uniform knots, giving 32 coefficients, with ridge 10^{-2}. Ridge values are selected for each representation on validation data, since the same numerical value need not impose the same function penalty across differently scaled bases.

Training cost. Appendix D.4 reports wall-clock training time in GPU-minutes and peak CPU memory by model and training asset.

C.3 Observation Protocols

Common inputs. For each asset, date and observation protocol, participating methods use the same prespecified input and query sets, generated with random seed 20260925. Model failures do not change these sets, and no method receives extra quotes to meet an implementation-specific minimum.

Standard and reference-nine inputs. P1 uses five random permutations per date, taking $n_S = \mathrm{round}(0.5N_t)$ quotes as inputs. HyperIV selects its references within that pool; its unused available quotes are not added to the common query set. P2 selects nine references from D_t and scores $D_t \setminus S_9$. Expiries are matched greedily, without reuse, to 7, 30 and 90 days. Within each selected expiry, distinct contracts are matched to forward deltas 0.5, 0.25 and -0.25, corresponding to the reference rule's ATM, call and put targets. Matching uses observed contracts, so actual maturities and deltas may differ from the targets. The chosen contracts and matching discrepancies are retained. One deterministic reference set is used per date. Dates that cannot supply nine distinct contracts and at least one query quote are ineligible.

Alternative reference locations. P3 holds the P2 trained models fixed and compares the reference rule with five random nine-point sets and one coordinate-coverage set. The common query set excludes the union of all these inputs. The coverage rule uses farthest-point selection in range-normalized (k, τ) coordinates.

Sparse objective comparison. The 64-point ablation reserves 20% of each day's quotes as a common query set, then samples 64 inputs from the remaining pool using five fixed random masks. All three objectives use these same input and query sets.

C.4 Metrics, Diagnostics, and Statistical Analysis

Quote and price metrics. For BA coverage, let Q_{BA} be the common subset with valid pricing fields and $0 \le \mathrm{bid}_j < \mathrm{ask}_j$. Then

$$\mathrm{BA} = \frac{1}{|Q_{\mathrm{BA}}|} \sum_{j \in Q_{\mathrm{BA}}} \mathbf{1}\{\mathrm{bid}_j \le P_j(\widehat{f}_S(x_j)) \le \mathrm{ask}_j\}, \tag{C.2}$$

where P_j is the contract's price under the IV-inversion convention. The size of this scoring subset is reported with the metric. Empty valid subsets yield a missing value, not zero coverage.

Financial shape. The common evaluation grid has 25 equally spaced r values and 81 equally spaced z values, including domain endpoints. Bfly is the fraction of grid points with $b(v) < -10^{-3}$, using Eq. (B.5) and its difference operators. Calendar pairs connect successive r values and transform z to keep $k = rz$ fixed, as in Eq. (B.6); Cal is the fraction of pairs with $c(v) < -10^{-3}$. Supplementary checks use a 49×161 grid, negative-part magnitudes, nonpositive predictions and the two directional roughness measures. Nonfinite predictions are recorded as failures.

Empirical projection diagnostic. For Eq. (4.2), we solve unpenalized weighted least squares on Q using SVD, separately for each fixed learned representation. If a model has a fixed mean function, we subtract that mean from the projection target and add it back to predictions. The same positive weights define E_{app} and E_{rec}, so orthogonal projection gives $E_{\mathrm{gap}} \geq 0$ up to numerical tolerance. Query count and effective rank accompany the diagnostic. We compute and average the three squared quantities per date, mask and seed; E_{rec} is not the square of the overall mean RWRMSE. The projection uses query labels after prediction and is not an alternative deployment fit. Neither the residual nor the gap separately identifies market observation noise.

For the sparse objective ablation, we also report the learned ridge parameter λ, the minimum eigenvalue γ of the input Gram matrix, and $(\gamma + \lambda)^{-1/2}$ from Theorem 3.3. We average λ over the five trained models and take the medians of the latter two diagnostics over all 6,300 date–mask–seed combinations. These are diagnostics in the learned basis coordinates, not a complete evaluation of the theorem's error bound.

Uncertainty and coverage. For each training seed, masks are averaged within date, followed by an equal-weight mean across dates. Across K completed seeds, we report the mean performance, stating the actual K. For preselected method comparisons, seed-averaged daily metrics are differenced on paired dates. We use 2,000 circular block-bootstrap replicates (Politis & Romano, 1992), with blocks of five consecutive trading dates wrapping at the end of the series and the same resampled dates for both methods. The resulting 95% intervals describe date uncertainty conditional on the fitted model collection. Quotes and masks within a date are not independent market replicates. Each result reports protocol eligibility, method success and any common successful subset used for paired scoring. Calibration, numerical and resource failures remain in the coverage record.

C.5 Runtime Measurement Protocol

The representative task uses the standard-input protocol (P1, mask 0) with batch size one: half of each SPX day's quotes are inputs and the remaining quotes are queries. Measurements cover 24 predetermined test dates, two per month, using the same inputs and queries across methods; the median input and query counts are both 1,182.5. CPU and GPU results are reported separately, with GPU measurements restricted to implemented GPU methods. Neural models run in single precision, and PCA and SSVI in double precision. Each method uses four CPU threads, three warm-up runs and 20 timed repetitions per date and task. Measurements use an exclusively allocated node with the hardware specified in Section 4.1.

Total latency starts from supplied input quotes and includes preprocessing, conditioning or calibration, any required graph construction, and all query predictions. Model loading and data download are excluded. We synchronize GPU work around measurements. Conditioning and query components follow the actual implementation dependencies; query-dependent graph work is included in total latency. Equal repetition counts within each method give each date equal weight when pooling timings. We report total p50 and p95, with p25–p75 as a descriptive interval.

The supplementary fixed-size task uses $n = 128$ inputs and $m = 1000$ queries. The supplementary nine-input task provides all methods with the same preselected reference contracts. Scaling measurements vary n over $16, 32, 64, 128$ at $m = 1000$, and m over $100, 1000, 10000$ at $n = 128$. Memory is the peak allocation reported by PyTorch during the timed repetitions for GPU runs, and the peak resident set size of the process for CPU runs. Reported training memory is peak CPU RSS from job accounting. Recorded offline training cost is reported separately from online surface construction, alongside parameter counts and model sizes in Appendix D.4.

D Supplementary Experimental Results

Performance values use the units and aggregation of Appendix C.4. Table highlighting follows Section 4.

Table D.1: Paired SPX recovery comparisons.

Compared model	Metric	Δ	95% CI	Dates
Standard inputs (P1)				
SSVI	RW (10^{-3})	54.8595	[51.8894, 57.8609]	252
PCA ridge	RW (10^{-3})	3.5869	[3.2544, 3.9465]	252
GNO	RW (10^{-3})	8.9735	[8.3973, 9.5835]	252
HyperIV	RW (10^{-3})	102.1026	[97.2458, 107.8337]	252
SSVI	MAPE (10^{-3})	38.9788	[37.0140, 40.8616]	252
PCA ridge	MAPE (10^{-3})	2.0422	[1.8415, 2.2562]	252
GNO	MAPE (10^{-3})	5.6607	[5.3601, 5.9804]	252
HyperIV	MAPE (10^{-3})	75.2659	[72.2402, 78.6562]	252
SSVI	BA (10^{-3})	-662.5621	[−682.6869, −640.7555]	252
PCA ridge	BA (10^{-3})	-177.7694	[−195.0549, −160.9684]	252
GNO	BA (10^{-3})	-375.7370	[−390.6302, −359.4315]	252
HyperIV	BA (10^{-3})	-704.2579	[−724.9196, −680.8985]	252
Reference nine (P2)				
SSVI	RW (10^{-3})	168.4243	[163.5526, 173.5434]	252
PCA ridge	RW (10^{-3})	68.5616	[60.7295, 76.5298]	252
GNO	RW (10^{-3})	335.1811	[323.9507, 346.2244]	252
HyperIV	RW (10^{-3})	62.2485	[57.4522, 67.2260]	252
SSVI	MAPE (10^{-3})	150.3680	[145.8360, 155.2529]	252
PCA ridge	MAPE (10^{-3})	49.0201	[43.2438, 54.8217]	252
GNO	MAPE (10^{-3})	257.0128	[245.0131, 268.7151]	252
HyperIV	MAPE (10^{-3})	47.7524	[45.0133, 50.5259]	252
SSVI	BA (10^{-3})	-140.4406	[−151.6487, −129.2566]	252
PCA ridge	BA (10^{-3})	-67.5229	[−83.5403, −51.5494]	252
GNO	BA (10^{-3})	-131.7663	[−143.1932, −120.4217]	252
HyperIV	BA (10^{-3})	-112.4146	[−122.5477, −102.6852]	252

Differences are compared model minus LFP (LFP-9 for P2). Intervals use paired date blocks.

D.1 Recovery on Market Quotes

Table D.1 supplies the paired comparisons underlying the SPX results. The intervals exclude zero in favor of LFP for RW, MAPE and BA against every baseline under both input protocols, supporting consistent gains in quote recovery.

Alternative reference locations. Table D.2 compares three nine-quote location rules and identifies the SSVI eligibility exception. LFP-9 leads the reported RW, MAPE and BA under all three rules, so its recovery advantage extends beyond the fixed reference locations. Absolute performance still depends on quote placement; SSVI results under random and coverage sampling are conditional on successful calibration.

Additional SPX surface examples. Figures D.1–D.2 show the same SPX-trained model on ten dates spaced through the test year, selected at the 5th, 15th, …, 95th percentiles of the ordered common trading dates. All examples use P1 mask 0 and the five-column conventions of Figure 4.1, including residual colours saturating at ± 90 basis points. The four appendix figures also share the RMSE bar range.

Table D.2: SPX recovery under nine-quote location rules.

Method	RW↓	MAPE↓	BA↑	Bfly↓	Cal↓
Reference nine					
SSVI	0.2127	0.1806	0.0106	**0.0000**	**0.0000**
PCA ridge	0.1127	0.0792	0.0838	0.1614	0.0197
GNO	0.3794	0.2873	0.0189	0.0491	0.0944
HyperIV	0.1064	0.0780	0.0385	**0.0000**	**0.0000**
LFP-9	**0.0443**	**0.0303**	**0.1510**	0.0005	**0.0000**
Random nine					
SSVI	0.3638	0.2183	0.0274	0.0300	**0.0000**
PCA ridge	0.0792	0.0489	0.1471	0.2108	0.0495
GNO	1.4873	0.5937	0.0642	0.0724	0.0624
HyperIV	0.8204	0.6204	0.0103	0.0424	0.0010
LFP-9	**0.0641**	**0.0356**	**0.1876**	**0.0174**	0.0024
Coverage nine					
SSVI	0.2437	0.1583	0.0332	**0.0257**	**0.0000**
PCA ridge	0.1173	0.0800	0.0719	0.2147	0.0843
GNO	22.2705	8.2922	0.0411	0.1566	0.1247
HyperIV	1.7336	1.3127	0.0051	0.0713	0.0046
LFP-9	**0.1000**	**0.0568**	**0.1284**	0.0309	0.0049

SSVI successful input sets: Reference nine: 252/252; Random nine: 804/1260; Coverage nine: 14/252. Other methods succeed on all recorded sets.

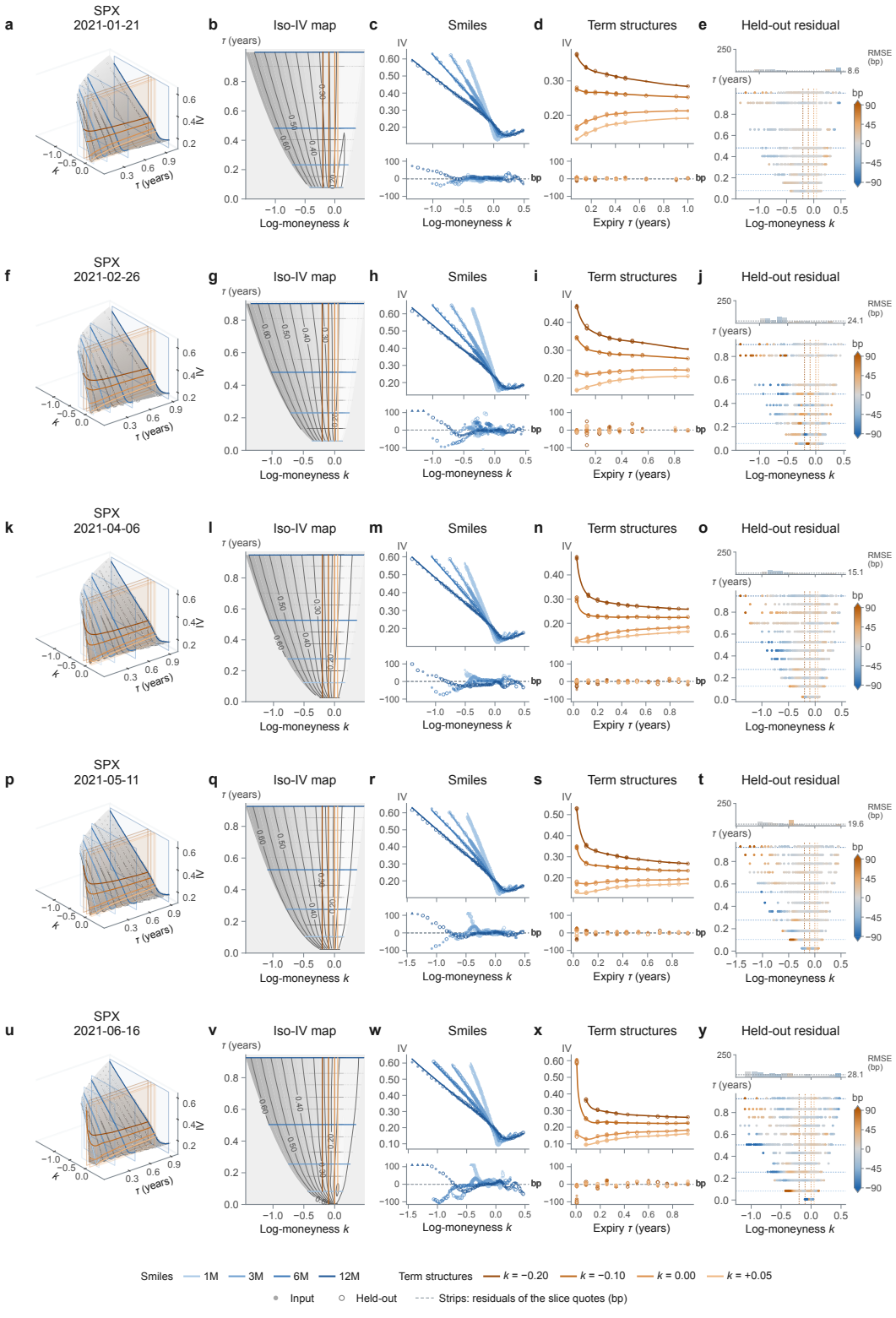

Figure D.1: SPX smoothing on five dates in the first half of 2021.

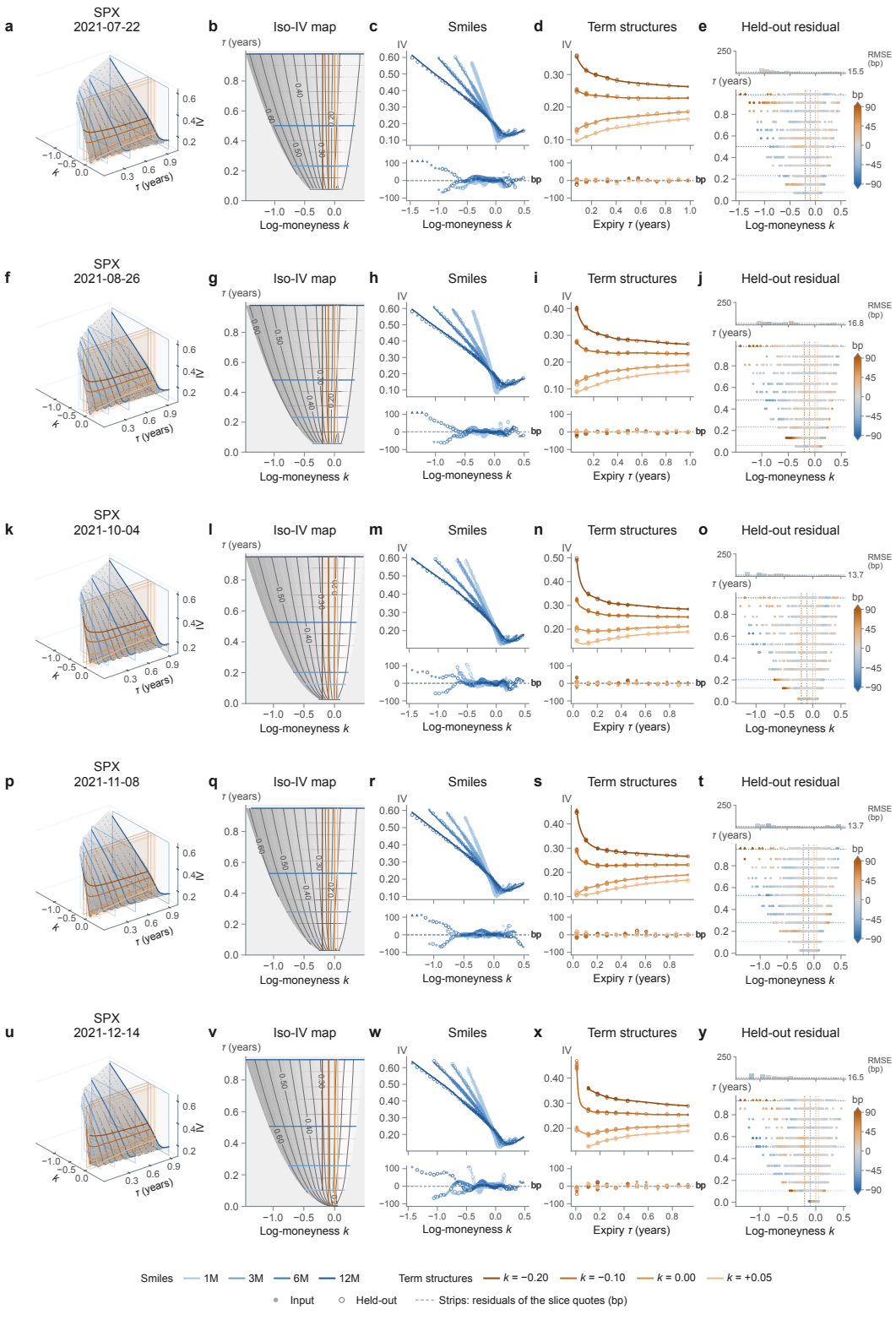

Figure D.2: SPX smoothing on five dates in the second half of 2021.

Table D.3: Cross-asset recovery across all training sources (P1).

		RW↓		MAPE↓		BA↑	
Test	Train	LFP	HyperIV	LFP	HyperIV	LFP	HyperIV
SPX	NDX	**0.0105**	0.1172	**0.0063**	0.0926	**0.4830**	0.0312
	RUT	**0.0103**	0.1365	**0.0059**	0.1054	**0.5372**	0.0272
	DJX	**0.0151**	0.1332	**0.0089**	0.1046	**0.3815**	0.0240
	SPX	0.0055	0.1076	0.0034	0.0787	0.7428	0.0385
NDX	SPX	**0.0049**	0.1223	**0.0031**	0.0817	**0.8871**	0.0820
	RUT	**0.0049**	0.0876	**0.0031**	0.0629	**0.8851**	0.0915
	DJX	**0.0091**	0.1178	**0.0060**	0.0838	**0.6693**	0.0542
	NDX	0.0044	0.0788	0.0028	0.0546	0.9190	0.0943
RUT	SPX	**0.0076**	0.1223	**0.0042**	0.0817	**0.9244**	0.1487
	NDX	**0.0078**	0.0808	**0.0042**	0.0553	**0.9300**	0.1892
	DJX	**0.0113**	0.0951	**0.0068**	0.0640	**0.7824**	0.1630
	RUT	0.0075	0.0794	0.0041	0.0548	0.9287	0.1932
DJX	SPX	**0.0531**	0.5322	**0.0247**	0.3876	**0.9524**	0.4004
	NDX	**0.0416**	0.5917	**0.0202**	0.4346	**0.9662**	0.4312
	RUT	**0.0430**	0.4547	**0.0215**	0.3183	**0.9605**	0.4505
	DJX	0.0447	0.1807	0.0230	0.1175	0.9509	0.5929

Grey rows: target-asset training. Colours compare models within each transfer direction. DJX entries use the 1,253 input sets shared by both models; other assets use all 1,260.

D.2 Generalization across Assets

Tables D.4–D.9 give complete transfer metrics and paired intervals. On NDX and RUT, LFP leads all baselines on RW, MAPE and BA under both protocols, with paired intervals excluding zero. On DJX, LFP-9 leads the P2 point estimates on all three metrics; relative to PCA ridge, the BA improvement excludes zero, whereas the RW and MAPE intervals include zero. PCA retains its P1 advantage in RW and BA, as discussed in the main results. The DJX P1 HyperIV comparison excludes seven input sets on which fewer than nine reference contracts were selected; other recorded P1 and P2 comparisons have complete coverage.

Robustness across training assets. Table D.3 covers all 12 cross-asset directions, with target-trained models as references. LFP leads HyperIV in RW, MAPE and BA in every direction; all 36 paired intervals exclude zero, and RW is reduced by 88.1–96.0%. Relative to target-asset training, the median RW increase is 15.0% for LFP versus 38.1% for HyperIV. Thus, LFP's recovery advantage persists across training sources, with smaller median transfer degradation.

Cross-asset surface examples. Figures D.3–D.4 reuse three of the SPX dates in Figures D.1–D.2 for cross-asset examples. Arrows denote training asset to test asset; the plotting conventions and scales match the SPX examples.

Table D.4: Recovery on NDX under both input protocols.

Method	RW↓	MAPE↓	BA↑	Bfly↓	Cal↓
Standard inputs (P1)					
SSVI	0.0385	0.0253	0.2730	0.0001	**0.0000**
PCA ridge	0.0092	0.0058	0.6867	0.1866	0.0110
GNO	0.0198	0.0126	0.3948	0.0049	0.0020
HyperIV	0.1223	0.0817	0.0820	**0.0000**	**0.0000**
LFP	**0.0049**	**0.0031**	**0.8871**	0.0008	**0.0000**
LFP (target)	0.0044	0.0028	0.9190	0.0008	0.0001
Reference nine (P2)					
SSVI	0.1226	0.0987	0.0441	**0.0000**	**0.0000**
PCA ridge	0.1570	0.1076	0.0794	0.1663	0.0425
GNO	0.3545	0.2512	0.0287	0.0407	0.0800
HyperIV	0.1219	0.0813	0.0828	**0.0000**	**0.0000**
LFP-9	**0.0459**	**0.0293**	**0.2237**	0.0003	**0.0000**

All shared models are trained on SPX. The grey row is target-trained and excluded from ranking.

Table D.5: Paired recovery comparisons on NDX.

Compared model	Metric	Δ	95% CI	Dates
Standard inputs (P1)				
SSVI	RW (10^{-3})	33.6476	[30.3764, 37.1024]	252
PCA ridge	RW (10^{-3})	4.3176	[4.1131, 4.5256]	252
GNO	RW (10^{-3})	14.9382	[13.9985, 15.9508]	252
HyperIV	RW (10^{-3})	117.4338	[112.5173, 122.4313]	252
LFP (target)	RW (10^{-3})	-0.4091	[−0.6089, −0.2193]	252
SSVI	MAPE (10^{-3})	22.1746	[19.9921, 24.4669]	252
PCA ridge	MAPE (10^{-3})	2.7306	[2.5879, 2.8785]	252
GNO	MAPE (10^{-3})	9.4972	[9.0191, 9.9650]	252
HyperIV	MAPE (10^{-3})	78.5545	[75.2377, 81.9479]	252
LFP (target)	MAPE (10^{-3})	-0.3408	[−0.4631, −0.2211]	252
SSVI	BA (10^{-3})	-614.1100	[−638.7873, −589.1831]	252
PCA ridge	BA (10^{-3})	-200.3889	[−218.2624, −183.9926]	252
GNO	BA (10^{-3})	-492.3313	[−514.7369, −471.6821]	252
HyperIV	BA (10^{-3})	-805.1193	[−820.5480, −789.5101]	252
LFP (target)	BA (10^{-3})	31.8975	[23.3791, 40.2179]	252
Reference nine (P2)				
SSVI	RW (10^{-3})	76.7351	[69.6164, 83.7051]	252
PCA ridge	RW (10^{-3})	111.1249	[102.3273, 120.4627]	252
GNO	RW (10^{-3})	308.5771	[291.2539, 327.5498]	252
HyperIV	RW (10^{-3})	75.9720	[69.7189, 82.1445]	252
SSVI	MAPE (10^{-3})	69.3659	[63.0876, 75.6146]	252
PCA ridge	MAPE (10^{-3})	78.2821	[72.0660, 84.5850]	252
GNO	MAPE (10^{-3})	221.9149	[205.7974, 238.2505]	252
HyperIV	MAPE (10^{-3})	52.0019	[48.2030, 55.8849]	252
SSVI	BA (10^{-3})	-179.6128	[−194.9516, −164.1069]	252
PCA ridge	BA (10^{-3})	-144.3493	[−159.1251, −130.1327]	252
GNO	BA (10^{-3})	-195.0554	[−208.2177, −181.1882]	252
HyperIV	BA (10^{-3})	-140.9138	[−153.4920, −128.1206]	252

Differences are compared model minus LFP (LFP-9 for P2). Intervals use paired date blocks.

Table D.6: Recovery on RUT under both input protocols.

Method	RW↓	MAPE↓	BA↑	Bfly↓	Cal↓
Standard inputs (P1)					
SSVI	0.0507	0.0345	0.3483	**0.0000**	**0.0000**
PCA ridge	0.0117	0.0068	0.8009	0.1769	0.0091
GNO	0.0253	0.0153	0.5480	0.0025	0.0067
HyperIV	0.1223	0.0817	0.1487	**0.0000**	**0.0000**
LFP	**0.0076**	**0.0042**	**0.9244**	0.0002	**0.0000**
LFP (target)	0.0075	0.0041	0.9287	0.0003	0.0000
Reference nine (P2)					
SSVI	0.1007	0.0774	0.1154	**0.0000**	**0.0000**
PCA ridge	0.1946	0.1393	0.0791	0.1740	0.0579
GNO	0.4858	0.3348	0.0306	0.0363	0.0803
HyperIV	0.1198	0.0794	0.1548	**0.0000**	**0.0000**
LFP-9	**0.0560**	**0.0377**	**0.2849**	0.0011	0.0001

All shared models are trained on SPX. The grey row is target-trained and excluded from ranking.

Table D.7: Paired recovery comparisons on RUT.

Compared model	Metric	Δ	95% CI	Dates
Standard inputs (P1)				
SSVI	RW (10^{-3})	43.0733	[39.6776, 46.4579]	252
PCA ridge	RW (10^{-3})	4.0873	[3.7957, 4.3902]	252
GNO	RW (10^{-3})	17.6945	[16.1825, 19.2673]	252
HyperIV	RW (10^{-3})	114.6417	[110.2097, 119.6317]	252
LFP (target)	RW (10^{-3})	-0.1638	[−0.3107, −0.0151]	252
SSVI	MAPE (10^{-3})	30.2858	[28.0380, 32.5200]	252
PCA ridge	MAPE (10^{-3})	2.6515	[2.4610, 2.8377]	252
GNO	MAPE (10^{-3})	11.1098	[10.2543, 11.9358]	252
HyperIV	MAPE (10^{-3})	77.5333	[74.2616, 81.0059]	252
LFP (target)	MAPE (10^{-3})	-0.0785	[−0.1781, 0.0181]	252
SSVI	BA (10^{-3})	-576.0925	[−605.3857, −546.3863]	252
PCA ridge	BA (10^{-3})	-123.4605	[−134.0808, −111.9759]	252
GNO	BA (10^{-3})	-376.3471	[−396.3792, −356.1431]	252
HyperIV	BA (10^{-3})	-775.6679	[−792.9940, −756.8316]	252
LFP (target)	BA (10^{-3})	4.3076	[−0.6036, 9.2698]	252
Reference nine (P2)				
SSVI	RW (10^{-3})	44.6562	[40.2130, 49.2329]	252
PCA ridge	RW (10^{-3})	138.5587	[129.9627, 147.3410]	252
GNO	RW (10^{-3})	429.7571	[360.0579, 535.1605]	252
HyperIV	RW (10^{-3})	63.7180	[59.8342, 67.8604]	252
SSVI	MAPE (10^{-3})	39.7834	[35.9511, 43.8432]	252
PCA ridge	MAPE (10^{-3})	101.6521	[95.7775, 107.8352]	252
GNO	MAPE (10^{-3})	297.1982	[269.1989, 325.3984]	252
HyperIV	MAPE (10^{-3})	41.7255	[39.0574, 44.5550]	252
SSVI	BA (10^{-3})	-169.5064	[−184.7717, −154.7409]	252
PCA ridge	BA (10^{-3})	-205.7765	[−222.2865, −189.4257]	252
GNO	BA (10^{-3})	-254.2697	[−272.0520, −236.0430]	252
HyperIV	BA (10^{-3})	-130.0501	[−143.0538, −116.4114]	252

Differences are compared model minus LFP (LFP-9 for P2). Intervals use paired date blocks.

Table D.8: Recovery on DJX under both input protocols.

Method	RW↓	MAPE↓	BA↑	Bfly↓	Cal↓
Standard inputs (P1)					
SSVI	0.1274	0.0789	0.6888	0.0239	**0.0000**
PCA ridge	**0.0488**	0.0249	**0.9569**	0.2406	0.0804
GNO	0.0982	0.0467	0.9274	0.0536	0.0358
HyperIV	0.5322	0.3876	0.4004	**0.0130**	0.0004
LFP	0.0533	**0.0248**	0.9523	0.0552	0.0466
LFP (target)	0.0448	0.0231	0.9508	0.0432	0.0566
Reference nine (P2)					
SSVI	0.1915	0.1491	0.3735	**0.0022**	**0.0000**
PCA ridge	0.1495	0.1010	0.6721	0.1954	0.0548
GNO	0.5768	0.3139	0.3650	0.0486	0.1030
HyperIV	0.2608	0.1811	0.5860	0.0040	0.0005
LFP-9	**0.1402**	**0.0974**	**0.7663**	0.0464	0.0421

All shared models are trained on SPX. The grey row is target-trained and excluded from ranking. HyperIV P1 excludes 35 of 6300 predictions on seven input sets with fewer than nine selected references.

Table D.9: Paired recovery comparisons on DJX.

Compared model	Metric	Δ	95% CI	Dates
Standard inputs (P1)				
SSVI	RW (10^{-3})	74.1401	[63.0712, 85.4130]	252
PCA ridge	RW (10^{-3})	-4.5316	[−8.7667, −0.8576]	252
GNO	RW (10^{-3})	44.8968	[11.2797, 108.7301]	252
HyperIV	RW (10^{-3})	478.8847	[381.3903, 583.1744]	252
LFP (target)	RW (10^{-3})	-8.5387	[−13.0920, −4.6493]	252
SSVI	MAPE (10^{-3})	54.1602	[47.5589, 61.2104]	252
PCA ridge	MAPE (10^{-3})	0.0703	[−1.2374, 1.0719]	252
GNO	MAPE (10^{-3})	21.9303	[9.9037, 44.7524]	252
HyperIV	MAPE (10^{-3})	362.8250	[288.8906, 442.4050]	252
LFP (target)	MAPE (10^{-3})	-1.7163	[−3.2038, −0.3620]	252
SSVI	BA (10^{-3})	-263.4131	[−289.0858, −237.4828]	252
PCA ridge	BA (10^{-3})	4.6819	[1.1109, 8.3347]	252
GNO	BA (10^{-3})	-24.8945	[−34.7048, −16.1212]	252
HyperIV	BA (10^{-3})	-551.8536	[−595.0402, −509.1407]	252
LFP (target)	BA (10^{-3})	-1.4368	[−5.8202, 2.5423]	252
Reference nine (P2)				
SSVI	RW (10^{-3})	51.3147	[26.0230, 73.7913]	252
PCA ridge	RW (10^{-3})	9.3243	[−7.8269, 23.8645]	252
GNO	RW (10^{-3})	436.5991	[251.2362, 709.0941]	252
HyperIV	RW (10^{-3})	120.5923	[49.5488, 208.4092]	252
SSVI	MAPE (10^{-3})	51.6807	[33.2789, 68.3370]	252
PCA ridge	MAPE (10^{-3})	3.6395	[−10.4631, 15.4898]	252
GNO	MAPE (10^{-3})	216.5249	[169.5961, 279.6148]	252
HyperIV	MAPE (10^{-3})	83.7647	[30.0875, 151.0966]	252
SSVI	BA (10^{-3})	-392.8058	[−440.3994, −342.9995]	252
PCA ridge	BA (10^{-3})	-94.2117	[−149.6752, −40.7282]	252
GNO	BA (10^{-3})	-401.3425	[−444.4103, −358.7049]	252
HyperIV	BA (10^{-3})	-180.3543	[−230.0571, −133.7432]	252

Differences are compared model minus LFP (LFP-9 for P2). Intervals use paired date blocks.

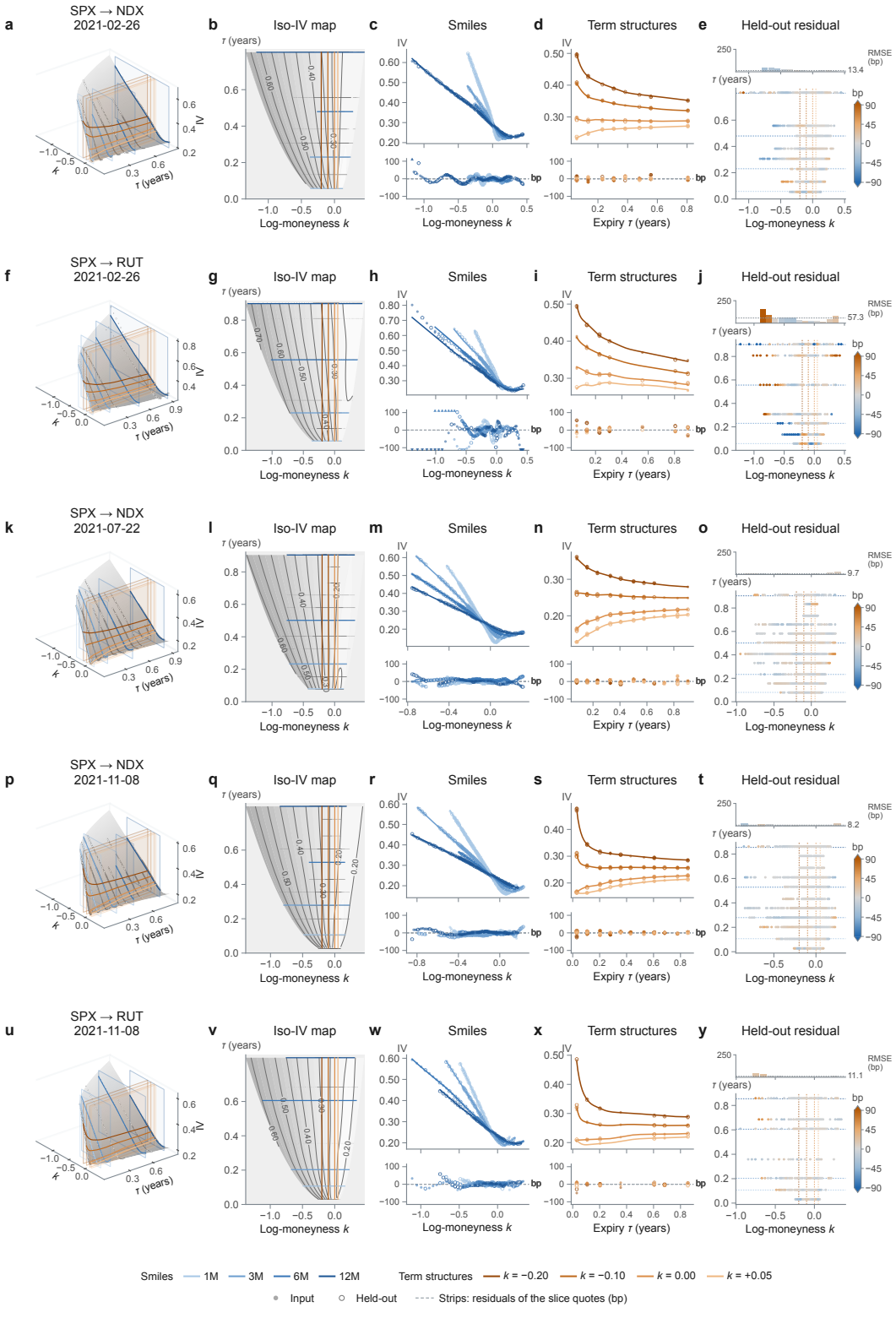

Figure D.3: Transfer of the SPX-trained LFP model to NDX and RUT across dates.

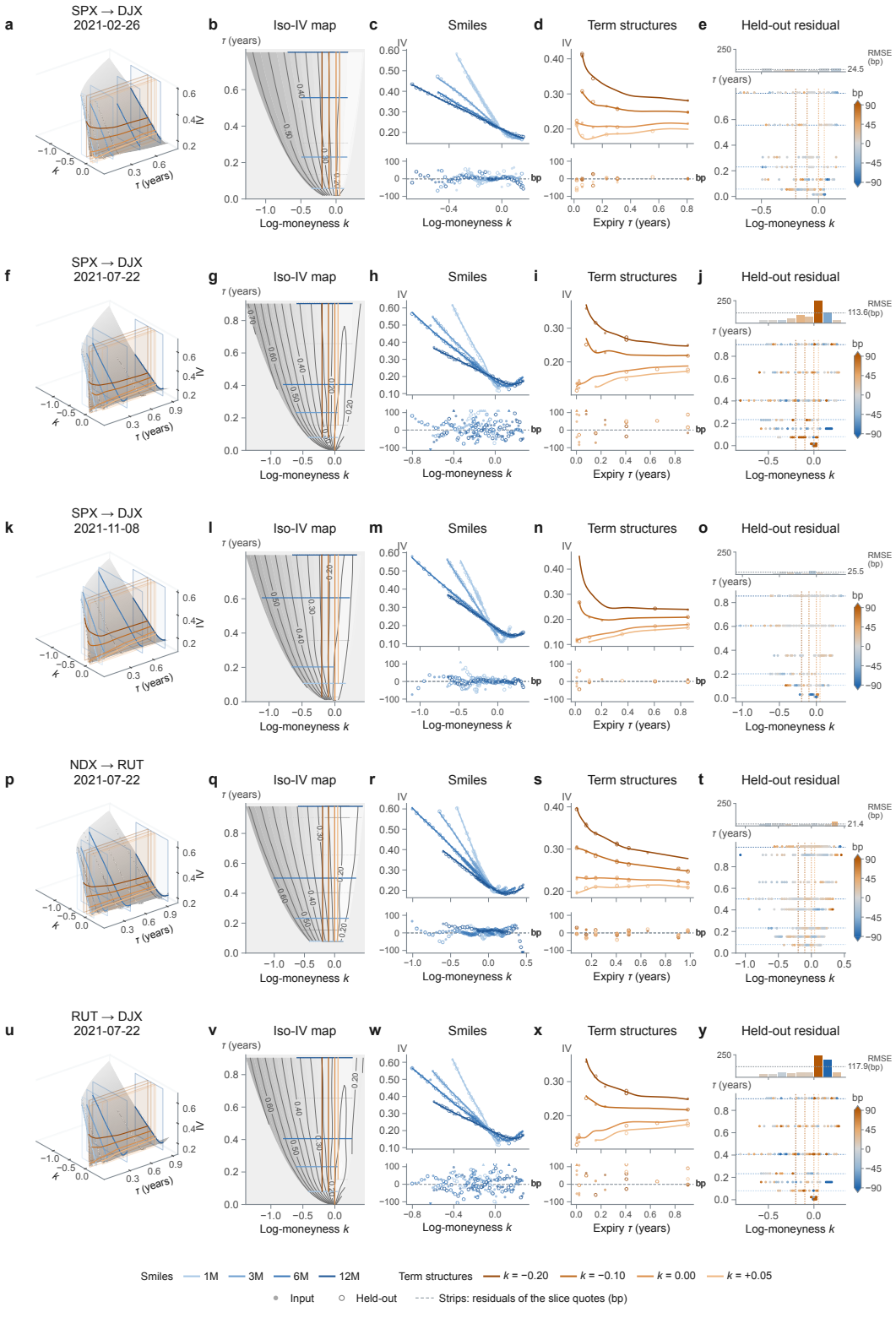

Figure D.4: SPX-to-DJX transfer across dates and additional training-source examples.

Table D.10: Supplementary diagnostics for the 64-quote objective ablation.

Objective	Input RW↓	Input MAPE↓	Bfly↓	Cal↓
A: full-pool	0.0038	0.0025	0.0046	0.0038
B: input	**0.0031**	**0.0020**	0.0026	0.0013
LFP	**0.0031**	0.0021	**0.0024**	**0.0005**

Compared model	Metric	Δ	95% CI	Dates
A: full-pool	RW (10^{-3})	10.0950	[9.2940, 10.9264]	252
B: input	RW (10^{-3})	2.9716	[2.4946, 3.4470]	252
A: full-pool	MAPE (10^{-3})	3.0889	[2.8475, 3.3440]	252
B: input	MAPE (10^{-3})	0.8436	[0.7164, 0.9701]	252
A: full-pool	BA (10^{-3})	-54.7762	[−60.5285, −49.2333]	252
B: input	BA (10^{-3})	-5.7330	[−9.3911, −1.9636]	252
A: full-pool	E_{app} (10^{-6})	-2.3542	[−2.8865, −1.8214]	252
B: input	E_{app} (10^{-6})	-0.0124	[−0.4624, 0.4083]	252
A: full-pool	E_{rec} (10^{-4})	8.1720	[7.1369, 9.2061]	252
B: input	E_{rec} (10^{-4})	2.4365	[1.6917, 3.2484]	252
A: full-pool	E_{gap} (10^{-4})	8.1955	[7.1596, 9.2278]	252
B: input	E_{gap} (10^{-4})	2.4367	[1.6965, 3.2487]	252
A: full-pool	Input MAPE (10^{-3})	0.4552	[0.4176, 0.4922]	252
B: input	Input MAPE (10^{-3})	-0.0154	[−0.0351, 0.0053]	252

LFP is recovery objective C. Paired differences are A or B minus C. Powers of ten give column multipliers.

Table D.11: Standard inputs: construction time in milliseconds.

Method	Total p50↓	Total p95↓	Condition p50↓	Query p50↓
CPU				
LFP	4.7918	5.1099	2.7078	2.0819
GNO	200.6735	229.6050	**0.4899**	200.1675
HyperIV	**2.4892**	**2.6373**	2.0084	0.4812
SSVI	15.7285	22.1740	15.6597	**0.0676**
PCA ridge	4.1623	4.4490	2.3561	1.8101
GPU				
LFP	2.7868	3.1341	1.7802	1.0125
GNO	314.9898	358.6366	**0.3661**	314.6289
HyperIV	**2.2714**	**2.6538**	1.7542	**0.5179**

Component medians need not sum to the total median.

D.3 Ablation Study

Table D.10 complements the sparse objective comparison with input-fitting errors and paired intervals. All paired intervals for held-out RW, MAPE, BA and recovery error favor C over both reconstruction objectives. A's lower approximation error thus does not translate into better recovery. The B–C intervals include zero for input MAPE and approximation error, while the reduction in coefficient recovery error favors C, supporting the benefit of held-out supervision.

D.4 Computational Efficiency and Resource Use

Tables D.11–D.15 supplement the timing results with task-specific latency, inference memory and offline cost. Across the measured tasks, LFP maintains millisecond-scale construction and remains substantially faster than GNO, with much lower GPU allocation under standard inputs. Figure D.5 shows modest growth in LFP's CPU cost over the measured input and query ranges. Time and memory ranks are within comparable groups; parameter counts are descriptive.

Table D.12: Common reference nine: construction time in milliseconds.

Method	Total p50↓	Total p95↓	Condition p50↓	Query p50↓
CPU				
LFP-9	2.9599	3.1970	1.1322	1.8282
GNO	34.6797	43.3201	**0.2368**	34.4460
HyperIV	2.1329	2.2137	1.6337	0.4979
SSVI	**1.3064**	**1.3620**	1.2286	**0.0771**
PCA ridge	2.9815	3.0891	1.1504	1.8317
GPU				
LFP-9	1.9560	2.0884	1.2078	0.7466
GNO	120.7414	129.1286	**0.2857**	120.4637
HyperIV	**1.6013**	**1.6572**	1.1252	**0.4769**

Component medians need not sum to the total median.

Table D.13: Fixed size: 128 inputs and 1000 queries: construction time in milliseconds.

Method	Total p50↓	Total p95↓	Condition p50↓	Query p50↓
CPU				
LFP	3.4039	3.5533	1.5209	1.8841
GNO	84.9125	91.3462	**0.4038**	84.5231
SSVI	14.6341	23.3451	14.5484	**0.0844**
PCA ridge	**3.1840**	**3.2909**	1.2958	1.8881
GPU				
LFP	**2.1005**	**2.1992**	1.3382	**0.7639**
GNO	153.6762	159.6096	**0.2969**	153.3792

Component medians need not sum to the total median. No HyperIV timing was recorded for this task.

Table D.14: Memory use under standard inputs (MiB).

Method	CPU peak RSS↓	GPU peak median↓	GPU peak max↓
LFP	720.0000	34.7195	34.9131
GNO	977.6250	159.8142	199.0298
HyperIV	756.0000	**34.0864**	**34.1069**
SSVI	**684.0000**	–	–
PCA ridge	720.0000	–	–

GPU columns summarize daily PyTorch allocation peaks; CPU RSS includes the whole process. Dashes denote CPU-only implementations.

Table D.15: Offline training cost and model size.

Model	Asset	Parameters	GPU-min↓	CPU peak RSS (MiB)↓
Standard-input models				
LFP	SPX	61985	**64.4167**	1872.0000
HyperIV	SPX	243153	69.9333	**1440.0000**
LFP	NDX	61985	67.1667	1728.0000
HyperIV	NDX	243153	**64.3000**	**1440.0000**
LFP	RUT	61985	**64.6667**	1800.0000
HyperIV	RUT	243153	64.8333	**1440.0000**
LFP	DJX	61985	**65.2500**	1728.0000
HyperIV	DJX	243153	68.3833	**1404.0000**
Alternative input or objective settings (SPX)				
LFP-9	SPX	61985	88.9000	1836.0000
A: full-pool	SPX	61985	65.6667	1836.0000
B: input	SPX	61985	70.8167	1872.0000

Per-model medians across five runs. Cost ranks compare standard-input models within each asset. GNO uses released weights (102529 parameters). PCA basis fitting takes 37.0727 CPU-seconds.

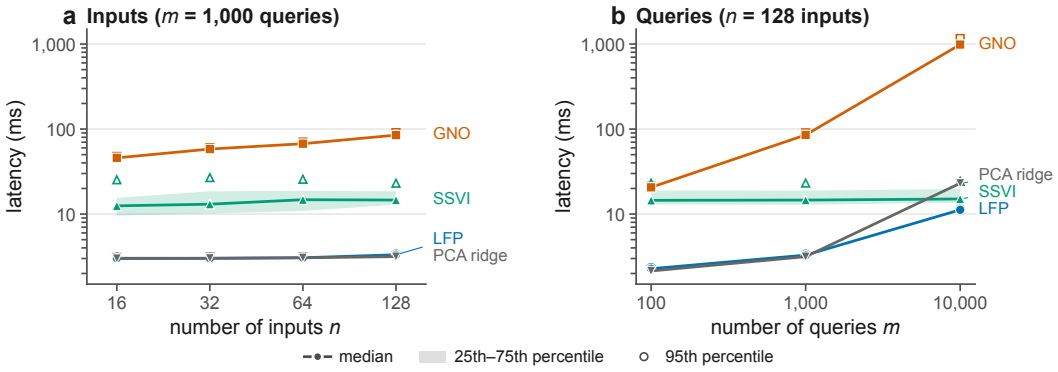

Figure D.5: CPU construction time as input and query counts vary. The other count is held at $m = 1000$ or $n = 128$, respectively.