Neural Slack Variables for Shape Constraints

Ruben Wiedemann
Imperial College London
London, UK
r.wiedemann22@imperial.ac.uk

Antoine Jacquier
Imperial College London
London, UK
a.jacquier@imperial.ac.uk

Lukas Gonon
University of St. Gallen
St. Gallen, Switzerland
lukas.gonon@unisg.ch

Abstract

Enforcing functional inequality constraints such as monotonicity and convexity in neural networks is a fundamental challenge in many industrial and scientific applications. Classical one-sided penalty methods, along with primal-dual methods gated by complementary slackness, lose their constraint gradients at satisfied locations, allowing previously feasible points to drift back into violation. Architectures that guarantee feasibility by construction, on the other hand, remain largely limited to elementary cases and impose additional inductive biases. We introduce *neural slack variables*, a deep learning native primal-side approach that converts constraint enforcement into a regression problem by coupling the primary network with a jointly learned auxiliary network. The auxiliary network serves as a positive target for the primary network's constraint quantities, inducing feasibility and regularity. On low-dimensional domains where dense supervision is practical, neural slack variables can mitigate constraint drift and bias the constraint profile toward the function class of the auxiliary network, which improves feasibility on our evaluation grids. Finally, in implied volatility surface modelling, an open industrial challenge in quantitative finance, neural slack variables preserve no-arbitrage constraints more reliably than penalty and primal-dual alternatives across both generation and calibration.

1 Introduction

In scientific computing and industrial domains such as control systems and quantitative finance, neural networks serve as learned function approximators on low-dimensional continuous domains. The learned map f^θ must both reproduce data with high fidelity and respect known structural laws such as monotonicity, convexity, stability, or absence of arbitrage. These are *shape constraints*: conditions on the function and its derivatives that must hold everywhere on the domain, expressed as functional inequalities $\mathcal{C}[f^\theta] > 0$. Standard neural architectures offer no intrinsic mechanism to preserve them, and unconstrained training routinely produces models that fit the data while remaining operationally inconsistent.

Specialised architectures that guarantee feasibility by construction (*architectural constraints*) exist for elementary constraint types: Constrained Monotonic Neural Networks (CMNN) [Runje and Shankaranarayana, 2023, Sartor et al., 2025] for monotonicity, Input-Convex Neural Networks (ICNN) [Amos et al., 2017] for convexity. These architectures restrict the hypothesis class in ways that can make them stiffer to train and bias the fit toward low-frequency solutions [Sivaprasad et al., 2021].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

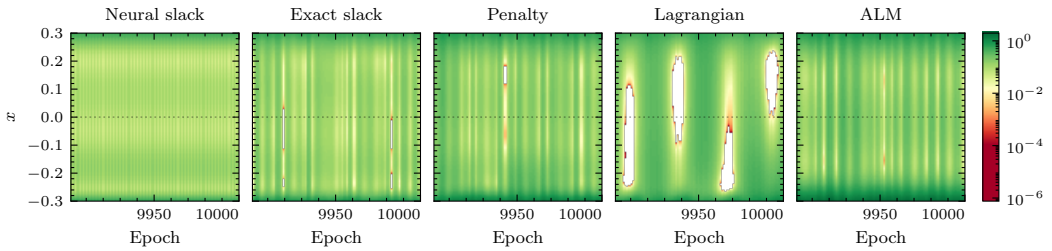


Figure 1: $\partial_x f^\theta$ on $[-0.3, 0.3]$ over the last 100 training epochs of the monotone experiment (Section 4.1.1), on a logarithmic colour scale with violations in white. The penalty method and the exact slack (Section 3) violate the constraint briefly (drift and overshoot) and the Lagrangian over long stretches (oscillation and overshoot), whereas neural slack variables keep a stable constraint profile in this run.

For general constraints and architectures, constraints must be enforced through the training objective instead (*soft constraints*). The default method in deep learning is the *penalty method*: a violation cost term is added to the loss, computed as an aggregate of the constraint residual. Its weight determines how strongly constraint satisfaction is enforced relative to data fit. Ramirez et al. [2025] advocate primal-dual methods, which introduce Lagrange multipliers that adaptively balance fit and feasibility, as the principled alternative.

In the deep learning regime, however, we argue that these classical methods face a dynamical mismatch: their constraint signals reflect static optimality conditions rather than being tailored to stabilise feasibility under long-running, nonconvex, stochastic optimisation with momentum. For the penalty method, the violation term supplies no gradient at any domain point where the constraint is satisfied. For a primal-dual method, projected ascent lets the multiplier of a satisfied constraint decay to zero, as complementary slackness prescribes at a constrained optimum [Platt and Barr, 1987]. Once the constraint signal has vanished in either method, continued optimisation of the data objective can reintroduce violations, starting a cycle of constraint drift and overcorrection. For primal-dual methods, this boundary-crossing behaviour has been recognised as "oscillation and overshoot", motivating controller-based multiplier updates that supplement multiplier ascent with proportional feedback from the current constraint residual to reduce repeated crossings [Stooke et al., 2020, Sohrabi et al., 2024].

The consequence of constraint drift is often unstable optimisation and spurious violations in trained models, which is a real practical challenge. In control systems, a learned barrier certificate carries no guarantee unless it can be formally verified [Dai et al., 2020, 2021, Edwards et al., 2024]. In quantitative finance, an implied volatility surface that admits even slight arbitrage is a significant issue for any downstream pricing and risk model [Gatheral and Jacquier, 2014, Deschâtres, 2024, Buehler et al., 2026]. Ensuring fully satisfied constraints is a high-impact industrial challenge even for low-dimensional parametric approaches.

Motivated by constraint drift, we propose to tie the primary network's constraint profile to a jointly learned positive target, rather than penalising only its negative part. Writing f^θ for the primary network, let $c_\theta := \mathcal{C}[f^\theta] \colon \Omega \to \mathbb{R}^m$ denote the constrained quantities. We introduce an auxiliary neural network s^ϕ—a learned analogue of the classical slack variable in constrained optimisation—whose outputs are positive by construction (through an appropriate output activation function) and provide the valid target in constraint space. The two networks are trained jointly, enforcing the slack-variable form

$$\mathcal{C}[f^\theta](x) - s^\phi(x) = 0, \qquad s^\phi(x) > 0,$$

softly through a quadratic loss in constraint space. This couples the two networks: s^ϕ adapts to the data-driven shape of c_θ, while c_θ is steered toward the positive target s^ϕ. As the residual shrinks, c_θ inherits the positivity of s^ϕ, making f^θ feasible. Crucially, because s^ϕ is a learned, finite-capacity approximation, the residual generally does not fully vanish, which leaves a constraint-space gradient after violations disappear and reduces drift of the constraint profile back into violation.

We term s^ϕ a *neural slack variable*. In our experiments, neural slack variables mitigate the drifting failure mode that we observe for penalty and primal-dual methods (Figure 1). We identify an

additional mechanism for constraint enforcement: the neural slack loss transfers the regularity of s^ϕ to the constraint profile c_θ (Figure 2), making the architecture of s^ϕ an explicit feasible inductive bias on f^θ toward a feasible function class. We use this lever to constrain spectrally expressive primary networks such as SIRENs [Sitzmann et al., 2020]; in fixed-grid settings, this can help keep satisfaction stable between collocation points, while the penalty method overfits the grid.

Viewed structurally, neural slack variables move the architectural constraint from the primary network to an auxiliary model of its constraint profile. In constraint space, feasibility is simply pointwise positivity of c_θ; the difficulty is to realise a valid profile through $\mathcal{C}$. This can be read as an inverse problem, and guaranteeing a positive profile by construction requires a primary architecture that exploits the structure of $\mathcal{C}$, available only for a few elementary constraint types. Neural slack variables instead require only a positive output activation of s^ϕ and solve the inverse problem approximately, by regressing $\mathcal{C}[f^\theta]$ onto s^ϕ during joint training.

Contributions

- We propose *neural slack variables*[1]: a soft constraint method that introduces a positive auxiliary network s^ϕ, jointly trained with the primary network through a matching loss in constraint space.
- We identify a drifting failure mode shared by penalty and primal-dual methods, supported by its visual signature in Figure 1 and an ablation that removes its source (Section 4.2), which neural slack variables can mitigate in the data-driven setting (Figure 5).
- We identify regularity transfer from s^ϕ to f^θ as a controllable inductive bias on the constraint profile, enabling constrained learning with spectrally expressive architectures (Figure 2).
- We demonstrate stronger constraint satisfaction on synthetic monotonicity and convexity test cases, and analyse neural slack dynamics on the data-free barrier certificate benchmark FOSSIL Barr3.
- As the main application, we apply neural slack variables to implied volatility surface modelling and obtain a generative model whose fitted and sampled surfaces show no measured arbitrage on dense evaluation grids, addressing a high-impact industrial challenge (Section 4.3).

2 Background and related work

An extended discussion of related work is provided in Section B.

2.1 Problem setup

On a bounded domain $\Omega \subset \mathbb{R}^d$, we learn a neural network $f^\theta \colon \Omega \to \mathbb{R}$ from data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$, subject to a functional m-dimensional inequality constraint

$$\mathcal{C}[f^\theta](x) > 0, \quad \text{for all } x \in \Omega, \tag{1}$$

where $\mathcal{C}[\cdot]$ is a constraint operator mapping the function to constraint quantities (derivatives, eigenvalues, or other functional properties). We write $\mathcal{L}_{\text{data}}(\theta)$ for the data-fitting loss (e.g., $\mathbb{E}_{(x,y)\sim\mathcal{D}}\|f^\theta(x) - y\|^2$ for the MSE loss).

2.2 Penalty method

The penalty method is the de facto default for enforcing constraints in deep learning. It augments the data loss with

$$\mathcal{L}_{\text{hinge},p}(\theta) = \frac{1}{p}\mathbb{E}_{x\sim\mathcal{U}(\Omega)}\left[\left\|\max\big(0,\ \varepsilon - \mathcal{C}[f^\theta](x)\big)\right\|_p^p\right], \tag{2}$$

where $\mathcal{U}(\Omega)$ is the Uniform distribution over Ω, $p \geq 1$ (typically $p = 1$ to encourage sparse violation patterns), the margin $\varepsilon > 0$ encourages strict satisfaction of the constraint, and "hinge" signals the use of the positive-part function to compute the constraint residual. The total loss becomes

[1] Code is available at `https://github.com/rwicl/neural-slack-variables-for-shape-constraints`.

$\mathcal{L}(\theta) := \mathcal{L}_{\text{data}}(\theta) + \rho\, \mathcal{L}_{\text{hinge},p}(\theta)$, where $\rho > 0$ weights the penalty term versus the data loss. On a grid $\pi = \{x_i\}_{i=1}^{I}$, this discretises to the classical exterior penalty method applied to the sampled constraints $\mathcal{C}[f^\theta](x_i) \geq \varepsilon$ [Nocedal and Wright, 2006, Ch. 17]. Here, the level of ρ governs *exactness* (whether the minimiser coincides with the constrained optimum), so ρ is a crucial tuning parameter in practice. For the linear penalty ($p = 1$), exactness may be attained at sufficiently large finite ρ, while the quadratic penalty ($p = 2$) is generally exact only in the limit $\rho \to \infty$, motivating schedules that increase ρ over training. These considerations are, however, largely set aside in deep learning practice; Ramirez et al. [2025] advocate primal-dual methods instead, citing in particular the burden of penalty weight tuning.

2.3 Primal-dual methods

The *primal-dual* or *Lagrangian* formulation of (1) associates a non-negative multiplier function $\lambda \colon \Omega \to \mathbb{R}^m_{\geq 0}$ with the constraint:

$$\min_{\theta} \max_{\lambda(\cdot)\geq 0} \; \mathcal{L}_{\text{data}}(\theta) + \mathbb{E}_{x\sim\mathcal{U}(\Omega)}\left[\left\langle \lambda(x),\; \varepsilon - \mathcal{C}[f^\theta](x)\right\rangle\right]. \tag{3}$$

On a grid $\pi = \{x_i\}_{i=1}^{I}$, this discretises to one multiplier vector $\lambda_i \in \mathbb{R}^m_{\geq 0}$ per grid point, equivalently one scalar multiplier per grid point and constraint component. *Gradient descent-ascent* (GDA) on (3) is the standard primal-dual scheme in constrained deep learning [Ramirez et al., 2025, Gallego-Posada et al., 2025]. Multiplier ascent acts as integral feedback and can lead to oscillations in the constraint and multiplier. The *augmented Lagrangian* introduces a quadratic term, which can improve convergence [Platt and Barr, 1987]; controller-based updates supplement multiplier ascent with proportional feedback from the current constraint residual [Stooke et al., 2020, Sohrabi et al., 2024].

When Ω is high-dimensional or extended by conditioning variables, materialising one scalar per constraint is impractical, motivating direct parametrisation of λ as a neural network $\lambda_\phi \colon \Omega \to \mathbb{R}^m_{\geq 0}$, a *neural multiplier*. Narasimhan et al. [2020] introduce neural multipliers for heavily-constrained classification, where the multiplier is indexed by a constraint feature vector (directly analogous to the domain coordinate x in our setting). They study how the capacity of λ_ϕ distorts the constrained problem: under-parameterised models effectively enforce averaged transformations of the constraint set rather than individual constraints. Section B.2 details the (augmented) Lagrangian and neural multiplier updates used in our experiments.

2.4 Architectural constraints

Finally, it is possible to guarantee feasibility by explicit architecture choices. Input-Convex Neural Networks [Amos et al., 2017] enforce convexity via non-negative weights and convex activations; Constrained Monotonic Neural Networks [Runje and Shankaranarayana, 2023, Sartor et al., 2025] guarantee monotonicity through paired sign-restrictions on the weights and increasing activation functions. SDP-based Lipschitz Layers (SLLs) [Araujo et al., 2023] similarly guarantee a global 1-Lipschitz bound through an analytic parameterisation of a semidefinite constraint on each layer, with applications in certified robustness and neural distance fields [Coiffier and Béthune, 2024].

3 Method

Neural slack variables Let $c_\theta := \mathcal{C}[f^\theta] \colon \Omega \to \mathbb{R}^m$ denote the constraint profile of the primary network f^θ. We introduce an auxiliary network $s^\phi \colon \Omega \to K$, where its range $K = [\varepsilon, \infty)^m$ is enforced through an appropriate output activation. We propose to enforce the inequality constraint $c_\theta > 0$ by augmenting the data loss with the term

$$\mathcal{L}_{\text{slack}}(\theta, \phi) := \frac{1}{2}\mathbb{E}_{x\sim\mathcal{U}(\Omega)}\left[\left\|T(c_\theta(x)) - T\big(s^\phi(x)\big)\right\|_2^2\right], \tag{4}$$

where the scalar coordinate transform T allows us to adapt the error geometry; we motivate our practical choice $T_\sigma(z) = \sigma\,\operatorname{asinh}(z/\sigma)$ on $z \geq 0$ via its relation to the symmetric relative error in Section A.1, while $T = \mathrm{id}_{\mathbb{R}}$ recovers the ordinary L^2 error. We call s^ϕ a *neural slack variable*: it carries the feasibility role of a classical slack variable, while its agreement with c_θ is learned through the slack loss rather than an algebraic identity. The total loss becomes $\mathcal{L}(\theta, \phi) := \mathcal{L}_{\text{data}}(\theta) + \rho\, \mathcal{L}_{\text{slack}}(\theta, \phi)$, a relaxation of the original constrained problem, minimised jointly in (θ, ϕ).

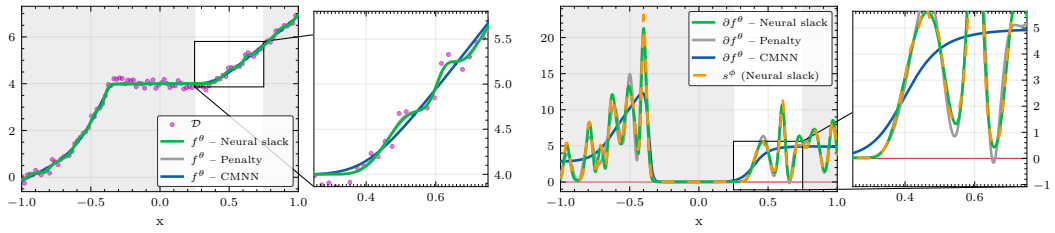


Figure 2: Monotone regression with a SIREN f^θ on a fixed grid π_{train} (Section 4.1.1): f^θ against observations $\mathcal{D}$ (left) and $\partial_x f^\theta$ with neural slack s^ϕ (right). The penalty method overfits to π_{train} with narrow excursions below zero between collocation points although its loss term vanishes (Figure 7), whereas with neural slack variables $\partial_x f^\theta$ follows s^ϕ and remains feasible.

Gradient persistence via variable splitting The construction can be understood as variable splitting on the constraint quantity. Introduce an auxiliary variable s on Ω and replace the original constraint $c_\theta \in K$ by the constraint pair $(s = c_\theta) \wedge (s \in K)$. Variable splitting schemes relax the coupling $s = c_\theta$ (e.g., via penalties or primal-dual methods) and thereby decouple the optimisation of the primary model parameters θ from the formation of a feasible constraint profile s. As the equality residual is reduced, feasibility transfers back to c_θ. The simple square penalty variant (with $T = \mathrm{id}_{\mathbb{R}}$ and $\rho = 1$) yields the constrained problem

$$\min_{\theta, s(\cdot)} \left\{ \mathcal{L}_{\text{data}}(\theta) + \frac{1}{2}\, \mathbb{E}_{x\sim\mathcal{U}(\Omega)} \left[\|c_\theta(x) - s(x)\|^2 \right] \right\}$$
$$\text{subject to } s(\cdot) \in K. \tag{5}$$

For fixed θ, the s-subproblem is solved by the *exact slack* $\Pi_K(c_\theta(x)) = \max(\varepsilon, c_\theta(x))$. Substituted into (5) this gives precisely the objective of the quadratic penalty method.

Instead of optimising (5) over all feasible auxiliary profiles, we restrict the auxiliary profile to the neural slack class $\mathcal{S} = \{s^\phi : \phi \in \Phi\}$. This suggests a nested optimisation scheme: fit s^ϕ at fixed θ before each update of the primary network. In practice, we optimise (θ, ϕ) jointly, a simpler scheme that works well in our experiments (a choice also for neural Lagrangian methods [Narasimhan et al., 2020, Park and Hentenryck, 2023]; Section B.2; Section G.5 includes results for a nested scheme). Neural slack variables thereby replace the exact pointwise s-projection by a learned, finite-capacity surrogate. The approximate projection is the key: The residual $r_{\theta,\phi}(x) = c_\theta(x) - s^\phi(x)$ generally does not collapse to zero throughout the slack region, so that $\mathcal{L}_{\text{slack}}$ continues to provide an "anchoring" gradient even as $c_\theta \geq \varepsilon$, stabilising feasible constraint profiles and preventing drift.

To isolate the role of the approximate projection, we compare against the exact slack $\Pi_K(c_\theta)$ as the target in (4), which gives the quadratic penalty $\mathcal{L}_{\text{hinge},2}$ with violations measured through T. To introduce a nonzero residual at feasible constraint values explicitly, we also use the exponential moving average (EMA) of the exact slack over training as the target, pointwise on a fixed grid (Section D.3) and represented by a positive network for resampled constraint points (Section G.5).

Inductive bias on the constraint profile Restricting the auxiliary profile to the finite-capacity class $\mathcal{S}$ adds a term to the inner-minimised slack loss:

$$\inf_{\phi} \mathcal{L}_{\text{slack}}(\theta, \phi) \;\geq\; \mathcal{L}_{\text{hinge},2}(\theta) \;+\; \frac{1}{2}\, \mathrm{dist}^2_{L^2}\big(\Pi_K(c_\theta),\, \mathcal{S}\big). \tag{6}$$

For $T = \mathrm{id}_{\mathbb{R}}$, the bound is the projection decomposition of $\|c_\theta - s^\phi\|^2$, valid because $\mathcal{S} \subset K$ by construction. If $c_\theta \geq \varepsilon$, the quadratic violation penalty term vanishes and the inner-minimised slack loss reduces to $\frac{1}{2}\, \mathrm{dist}^2_{L^2}(c_\theta, \mathcal{S})$. Therefore, the slack loss selects, among feasible θ of equal data fit, one whose constraint profile lies closest to $\mathcal{S}$. This is an explicit inductive bias on the shape of c_θ, set by the architecture of s^ϕ rather than f^θ. A small $\mathcal{S}$ then may underfit, while a larger class weakens this bias.

Choosing the neural slack class Optimisation also depends on the mobility of s^ϕ, the output change induced by an update of ϕ. Its learning rate α_ϕ sets the size of the parameter update, while architecture and width determine how strongly that update changes the neural slack variable;

Section A.3 gives a heuristic gradient-flow analysis of the joint training dynamics. The differential constraint operators we consider amplify high-frequency content, in tension with an MLP-based neural slack class [Rahaman et al., 2019, Ramasinghe and Lucey, 2022]; we therefore use SIRENs, with Fourier features as another possible choice. We extract a basic slack selection rule: 1) Begin with the primary network's architecture and learning rate α_θ, then sweep the ratio $\alpha_\phi/\alpha_\theta$ around one. 2) If a higher ratio reduces slack error without improving data fit, increase the slack width or consider a more spectrally expressive architecture. The useful regime therefore combines sufficient capacity to match the data-relevant constraint profiles of f^θ with suitable mobility for stable joint optimisation.

Sobolev slack loss For $q \in \mathbb{N}_0$, the transformed profiles can instead be compared in $H^q(\Omega;\mathbb{R}^m)$ through the slack loss

$$\mathcal{L}^{(q)}_{\text{slack}}(\theta,\phi) := \frac{1}{2}\left\|T(c_\theta) - T(s^\phi)\right\|^2_{H^q(\Omega;\mathbb{R}^m)}.$$

The case $q = 0$ gives the L^2-slack loss (4), while $q > 0$ also penalises derivatives up to order q.

4 Experiments

We compare neural slack variables with penalty and primal–dual methods and, where available, architecturally constrained networks.

Training and hyperparameters With the exception of Experiment 4.3, the datasets are small enough to resolve $\mathcal{L}_{\text{data}}$ on the full training set at each optimisation step (full-batch gradient descent). All method-specific constraint objectives are evaluated on a separate constraint grid π_{train}. The comparison methods and their hyperparameters are specified alongside each experiment and in the corresponding appendix.

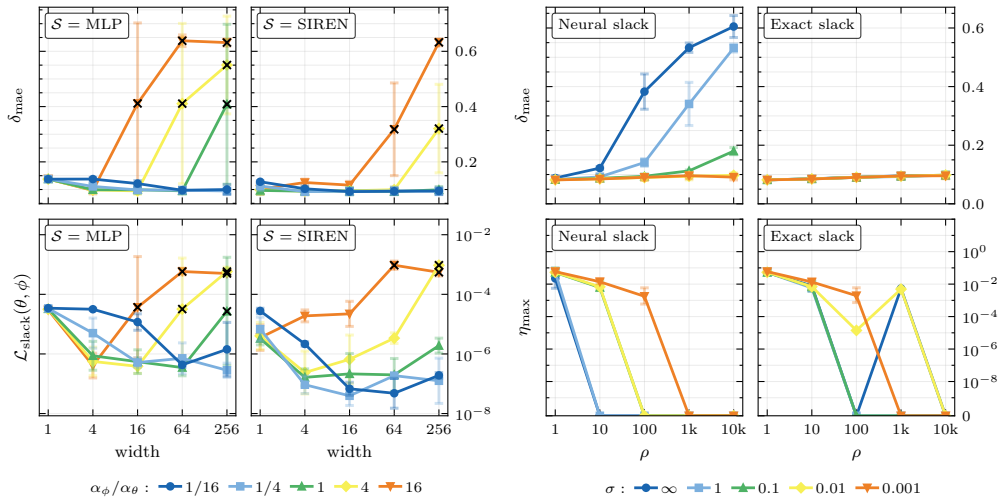
Evaluation metrics Each experiment uses multiple training runs per method, with seeds paired across methods. We measure accuracy by the training mean absolute error δ_{mae}, and constraint satisfaction on a fine evaluation grid π_{eval} by the violation rate η_{rate} and the average and worst-case violation magnitudes η_{mean} and η_{max} over violating points. Across runs, we report the mean and standard deviation of δ_{mae}, the number N_{sat} of runs without violations, and the median, minimum and maximum of each constraint statistic over the remaining runs (definitions in Section C). Zero measured violations on π_{eval} do not certify feasibility between evaluation points (Section C.2).

4.1 Test cases: monotonicity and convexity

4.1.1 Monotonicity ($d = 1$)

We study the properties of neural slack variables on an illustrative one-dimensional monotonicity problem. We learn an increasing function on $\Omega = [-1, 1]$ from 100 noisy observations of a hand-crafted target with an adversarial, slightly negatively sloped terrace in the centre of the interval (discernible from Figure 2; full details in Section D.1). The constraint $\mathcal{C}[f](x) = \partial f(x) \geq 0$ is enforced on a 200-point training grid π_{train} and verified on a 10,000-point evaluation grid π_{eval}. We train full-batch for 10,000 epochs with Adam at learning rate 10^{-3}. As an architecturally constrained baseline, we use CMNN [Sartor et al., 2025] with four hidden layers of width 16 (the minimum depth of the architecture; Section D.1).

Slack selection: architecture $\times$ width $\times$ learning rate We fix f^θ as a $\mathrm{softplus}$-MLP of the same size as the CMNN baseline. We test the slack selection rule by comparing MLP and SIREN slacks at widths $W \in \{1, 4, 16, 64, 256\}$ and relative learning rates $\alpha_\phi/\alpha_\theta \in \{1/16, 1/4, 1, 4, 16\}$. The neural slack uses an exponential output map, $\varepsilon = 10^{-2}$, and the slack error transform T_σ with $\sigma = 0.01$. Figure 3a reports δ_{mae} and $\mathcal{L}_{\text{slack}}(\theta,\phi)$ over five seeds for every cell. The width-one MLP remains inaccurate at every learning rate ratio, while increasing either the width or the ratio moves the MLP slack into the low-error regime. The SIREN slack reaches this regime already at width 1 for $\alpha_\phi/\alpha_\theta \geq 1$, but $\alpha_\phi/\alpha_\theta = 16$ degrades both metrics as its width grows. Infeasible seeds occur exactly in the cells with $W\alpha_\phi/\alpha_\theta \geq 256$ for the MLP slack and $W\alpha_\phi/\alpha_\theta \geq 1024$ for the SIREN slack, which also have the largest δ_{mae}.



(a) Slack selection.

(b) Slack error geometry.

Figure 3: Monotone MLP f^θ, five seeds per cell. **(a)** δ_{mae} (mean ± std) and $\mathcal{L}_{\text{slack}}$ (geom. mean ± geom. std) across slack architecture, W and $\alpha_\phi/\alpha_\theta$; black crosses mark cells with an infeasible seed. Low error requires sufficient capacity and mobility, which a SIREN slack reaches already at width 1, while excessive mobility (large $W\alpha_\phi/\alpha_\theta$) coincides with infeasible seeds. **(b)** δ_{mae} (mean ± std) and $\eta_{\max}$ (median [min, max] over infeasible seeds) across ρ and σ for the neural and the exact slack. Tuning σ permits larger ρ for feasibility at competitive fit. Tables 4 and 5 list the values.

Slack error geometry Figure 3b compares a SIREN slack of width 16 at $\alpha_\phi/\alpha_\theta = 1/2$ with the exact slack $\Pi_K(c_\theta)$ for the softplus-MLP f^θ, across the slack loss weight ρ and the parameter σ of the slack error transform T_σ. Without the transform ($\sigma = \infty$), the neural slack variable is feasible on all seeds from $\rho = 10$, where its δ_{mae} of 0.12 already exceeds the 0.097 of the exact slack at $\rho = 10^4$. Its δ_{mae} increases further to 0.60 at $\rho = 10^4$. For $\sigma < \infty$, T_σ assigns less weight to discrepancies at large, already feasible constraint values, so smaller σ requires larger ρ for feasibility on all seeds. At $\sigma = 0.01$, all seeds are feasible from $\rho = 100$, with δ_{mae} at most 0.097. Tuning σ thus permits larger ρ at competitive fit. The exact slack keeps its fit at every σ, because its projection error is zero at feasible constraint values, but it leaves one of five seeds infeasible at three settings with $\rho \geq 100$.

Constraint drift Figure 1 illustrates constraint drift on one seed at a constant learning rate, comparing neural slack variables with the exact slack, the linear penalty, the Lagrangian method and the augmented Lagrangian method (ALM), at a matched margin, the current minimum of s^ϕ. Over the last 1,000 epochs, the neural slack method has no violation (last 100 epochs in Figure 1), while Figures 5 and 6 show drift and overcorrection for the penalty method and the exact slack (violations in 0.6% and 2.7% of these epochs), and the "oscillation and overshoot" of primal-dual methods [Stooke et al., 2020, Sohrabi et al., 2024] for the Lagrangian method and ALM (23.5% and 0.1%). We attribute the visible anchoring of c_θ to s^ϕ to the dense gradients of the regression onto the finite-capacity s^ϕ: the exact slack shares the violation term of the slack loss (6) but has no gradient above the matched margin. Over 100 seeds of Table 1 (MLP f^θ), a learning rate decay over the last 1,000 epochs damps the drift enough for the penalty method and ALM to end feasible on 99 and 94 seeds, yet every one of their seeds violates the constraint during the decay (Tail). Neural slack variables remain feasible throughout the decay on 28 seeds and attain the lowest δ_{mae} among the soft methods.

Grid overfitting A fixed training grid π_{train} constrains $\partial_x f^\theta$ only at its points. With a spectrally expressive SIREN f^θ, Figure 2 shows a single run from Table 1 (SIREN), in which the penalty method overfits to π_{train} with $\partial_x f^\theta < 0$ at 0.46% of the points in π_{eval}. Over the last 1,000 epochs, displayed in Figure 7, constraint drift and overcorrection of the penalty method are apparent in η_{rate} and δ_{mae}. Neural slack variables are the only method feasible on every seed, over the last 1,000 epochs and at the final epoch, and attain the lowest δ_{mae} among constrained methods, while the Lagrangian method and ALM become unstable. We attribute feasibility with neural slack variables to the explicit

Table 1: Monotone with an MLP or a SIREN f^θ and fixed π_{train}, over $N_{\text{tot}} = 100$ seeds: δ_{mae} (mean $\pm$ std); Tail and Final give $N_{\text{sat}}/N_{\text{tot}}$, the seeds with $\eta_{\text{tail}} = 0$ over the last 1,000 epochs and with $\eta_{\text{rate}} = 0$ at the final epoch. CMNN restricts f^θ to an MLP with constrained weights and has no SIREN counterpart. Penalty method, ALM, and neural slack method ramp their constraint weight ρ adaptively (Section D.1).

	MLP f^θ			SIREN f^θ		
Method	δ_{mae}	Tail	Final	δ_{mae}	Tail	Final
Unconstrained	7.7e-02 ± 5.9e-03	0/100	0/100	9.2e-05 ± 2.0e-04	0/100	0/100
Penalty	9.8e-02 ± 8.5e-03	0/100	99/100	7.9e-02 ± 6.7e-03	51/100	74/100
Lagrangian	9.3e-02 ± 7.8e-03	0/100	65/100	3.1e+00 ± 2.5e-01	0/100	0/100
ALM	9.8e-02 ± 9.1e-03	0/100	94/100	2.8e+00 ± 1.2e+00	9/100	12/100
NSV	9.2e-02 ± 7.4e-03	28/100	100/100	7.6e-02 ± 6.3e-03	100/100	100/100
CMNN	8.7e-02 ± 6.7e-03	100/100	100/100	—	—	—

bias of the neural slack class on f^θ, making narrow excursions globally expensive. Resampling π_{train} uniformly at each step removes the grid overfitting of the penalty method, which is then feasible on every seed at unchanged fit (Table 7). On resampled points, the Lagrangian methods would require neural multipliers.

4.1.2 Convexity ($d \in \{2, 3, 4, 5\}$)

We learn convex functions on $\Omega = [-1, 1]^d$ for increasing dimension $d \in \{2, 3, 4, 5\}$ from a smoothed log-sum-exp target with $M = 10$ random affine pieces (full details in Section E.1). The constraint $\mathcal{C}[f](x) = \text{eig}_{\min} \nabla^2 f(x) \geq 0$ requires the Hessian to be positive semidefinite. We train full-batch with Adam on 10^3 noisy observations for 10,000 epochs (learning rate 10^{-3}, cosine-annealed on the final 10%). At each optimiser step, the constraint objective is evaluated on a newly drawn Sobol grid π_{train} with $|\pi_{\text{train}}| = 10^4$. Evaluation uses an independent Sobol grid π_{eval} with $|\pi_{\text{eval}}| = 10^7$. The primary model f^θ is a width-128 three-layer softplus-MLP, matching the activation class required by the ICNN baseline; the neural slack variable s^ϕ is a width-128 three-layer SIREN with input frequency $\omega_0 = 5$. Since π_{train} becomes sparser as d grows, we additionally include the H^1 variant of the neural slack variable in our comparison (Section 3), alongside the linear penalty method, the ICNN, and an unconstrained baseline, using 5 seeds per method and dimension. The penalty method and the neural slack variables ramp their constraint weight ρ adaptively (Section E.2). Per-dimension metrics (Table 8) and $d = 2$ surface plots (Figure 8) are in Section E.

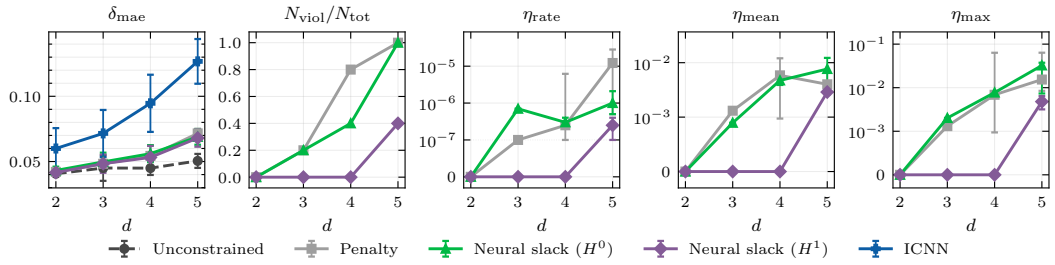


Figure 4: Convexity over d, five seeds per method: δ_{mae} (mean ± std), $N_{\text{viol}}/N_{\text{tot}}$, and η_{rate}, η_{mean}, η_{max} (median [min, max] over infeasible seeds). Supervised feasibility degrades with d, but neural slack with derivative matching (H^1) extends it to higher d, at an increased computational cost.

Results At $d = 2$, all three soft methods are feasible on every seed (Figure 4). At $d = 3, 4, 5$, the penalty method is feasible on 4, 1 and 0 of five seeds, neural slack (H^0) on 4, 3 and 0, and neural slack (H^1) on 5, 5 and 3. At each d, the sets of feasible seeds are nested, penalty $\subseteq$ neural slack (H^0) $\subseteq$ neural slack (H^1). δ_{mae} is comparable across the three soft methods at every d. We read the loss of feasibility with growing d as the curse of dimensionality acting on learned constraint enforcement: at fixed grid budget the sample density thins exponentially, and methods whose feasibility relies on dense pointwise supervision degrade accordingly. Gradient matching in the H^1 slack loss extends

feasibility to higher d, at the cost of more than twice the training time of neural slack (H^0) (Table 8). ICNNs are convex by construction but have higher δ_{mae} at every d, consistent with the slower training dynamics documented for input-convex architectures [Sivaprasad et al., 2021, Hoedt and Klambauer, 2023].

4.2 Neural barrier certificates

We turn to a multi-component safety certificate drawn from the FOSSIL Barr3 benchmark [Edwards et al., 2024]. On the autonomous polynomial system $\dot{x} = F(x)$ with $F(x) = (x_2,\ -x_1 - x_2 + \frac{1}{3}\,x_1^3)$ over $\Omega = [-3, 2.5] \times [-2, 1]$, we train $f^\theta \colon \Omega \to \mathbb{R}$ whose candidate safe set $\{x : f^\theta(x) \geq 0\}$ separates an initial set Ω_{i} from an unsafe set Ω_{u}, both unions of axis-aligned rectangles and disks (full geometry in Section F.1). Folding the boundary sign anchors and the Nagumo / control-barrier inequality [Ames et al., 2019, Dawson et al., 2023] into a single vector-valued constraint:

$$\mathcal{C}[f^\theta](x) = \big(\nabla f^\theta(x) \cdot F(x) + \gamma\, f^\theta(x),\ \mathbf{1}_{\Omega_{\text{i}}}(x)\big(f^\theta(x) - \mu_{\text{i}}\big),\ \mathbf{1}_{\Omega_{\text{u}}}(x)\big(-f^\theta(x) - \mu_{\text{u}}\big)\big)\ \geq\ 0, \qquad (7)$$

with $\gamma = 1$ and strict-positivity margins $\mu_{\text{i}} = \mu_{\text{u}} = 10^{-3}$ on the boundary anchors so that the certificate's safe set has non-empty interior on Ω_{i} and a strict separation from Ω_{u}. We give details about training and evaluation grids in Section F.1.

This setting removes the data-fitting term ($\mathcal{L}_{\text{data}} \equiv 0$), leaving the constraint-enforcement term as the only signal for θ. It therefore removes the source of constraint drift identified in Figure 1. Indeed, once the literature-default two-layer width-10 squared-activation MLP [Edwards et al., 2024] is feasible under the penalty method, the penalty loss vanishes at every sampled point and the updates stop (up to the decaying momentum of the optimiser; Figure 10).

Results For neural slack variables, the absence of a data term makes Barr3 a diagnostic of the method's own training dynamics. Figure 9 repeats the slack selection sweep of Section 4.1.1 with the worst constraint value $c^\theta_{\min}$ on π_{eval} in place of δ_{mae}: widths $W \in \{1, 4, 16, 64, 256\}$ and ratios $\alpha_\phi/\alpha_\theta \in \{1/16, 1/4, 1, 4, 16\}$ for SIREN ($\omega_0 = 5$) and $\tanh$-MLP slacks, with 5 seeds per cell. All five seeds are feasible in 3 of the 50 cells, each a SIREN slack with $\alpha_\phi/\alpha_\theta \leq 1/4$; no seed is feasible at $\alpha_\phi/\alpha_\theta \geq 1$, and the $\tanh$-MLP slack is feasible on no seed (Table 9). Without a data term, settling into a feasible state depends sensitively on neural slack architecture and learning rate.

4.3 No-arbitrage volatility surfaces

Neural approaches for arbitrage-free implied volatility surfaces generally enforce the no-arbitrage shape constraints through the penalty method, and the reported surfaces can retain residual violations [Ackerer et al., 2020, Chataigner et al., 2020, Zheng et al., 2021, Bergeron et al., 2022, Yang et al., 2025, Wiedemann et al., 2025]. Writing $v(\tau, k) := f^\theta(\tau, k; z)\sqrt{\tau}$ for *total volatility* and $w := v^2$ for *total variance*, the no-arbitrage constraint is

$$\mathcal{C}[f^\theta] = \big(\mathcal{C}_{\text{cal}}[f^\theta],\ \mathcal{C}_{\text{str}}[f^\theta]\big) = \big(\partial_\tau w,\ (1 + d_1\,\partial_k v)(1 + d_2\,\partial_k v) + v\,\partial_k^2 v\big)\ \geq\ 0,$$

where $d_{1,2}(\tau, k, v) := -k/v \pm v/2$ (details in Section G.1). Through Dupire's equation, $a^2 = \mathcal{C}_{\text{cal}}/\mathcal{C}_{\text{str}}$, these constraints determine whether the fitted surface yields a well-defined local volatility model for downstream pricing.

Our dataset comprises OptionMetrics snapshots for SPX, NDX, RUT, and DJX over 2020–2024. We rely on variational autodecoding [Park et al., 2019, Zadeh et al., 2021], assigning an eight-dimensional latent code to each listed expiry and using a small causal transformer to produce the FiLM parameters of the slice decoder; total variance is interpolated between adjacent expiries (details in Section G.3). We compare neural slack variables, the penalty method, and the neural augmented Lagrangian (neural ALM), whose multiplier network is updated jointly with the model or by phased regression as in PDL [Park and Hentenryck, 2023] (details in Section G.5, which also includes a neural version of EMA-based slack variables). Every method ramps its constraint weight ρ adaptively. Each of three independent runs is evaluated on its 1,000 fitted surfaces (FITTED) and on 1,000 samples from the aggregate posterior (POSTERIOR), using dense expiry-wise grids. To test new dates, we freeze the seed-0 decoder for each method and optimise only new latent paths for reconstruction under the learned GP prior [Bojanowski et al., 2018, Park et al., 2019] on 660 unseen surfaces from 2025 (HOLDOUT), without including constraints in the calibration objective.

Results The results are summarised in Table 2, with more details in Table 10. With neural slack variables, no violation is measured on any of the 3,000 surfaces in FITTED or POSTERIOR, and 650/660 surfaces are feasible after calibration in HOLDOUT. The penalty method attains lower δ_{mae} and lower calibration loss in HOLDOUT, but has measured violations on 38 surfaces in FITTED and 69 in POSTERIOR, and 582/660 feasible surfaces in HOLDOUT, the second-highest count. Joint and PDL-1 attain the lowest δ_{mae}, but every neural ALM variant has measured strike violations on more than 100 surfaces in every regime, pointing to a fit–feasibility tradeoff. Relative to PDL-1, five dual epochs reduce strike violations at about twice the training time, and joint updates reduce calendar violations at lower cost.

Table 2: Volatility autodecoder comparison. δ_{mae} as mean ± standard deviation over three runs. The remaining columns count surfaces with a calendar or strike violation, $N_{\text{viol}}/N_{\text{tot}}$: among 3,000 dense-grid evaluations (1,000 per run) in FITTED and POSTERIOR, and among 660 unseen 2025 surfaces in HOLDOUT after unconstrained calibration of the seed-0 model for each method. Neural ALM updates its multiplier network by gradient ascent at every step (joint), or alternates each primal epoch with k epochs of multiplier regression (PDL-k). In this table, neural slack variables are the only method without measured violations in FITTED and POSTERIOR and keep the most surfaces feasible in HOLDOUT. Table 10 additionally compares a neural version of EMA-based slack variables (neural EMA), which in this setting is close to neural slack variables and attains comparable feasibility at a higher δ_{mae}.

			FITTED		POSTERIOR		HOLDOUT	
Method		δ_{mae}	$\mathcal{C}_{\text{cal}}$	$\mathcal{C}_{\text{str}}$	$\mathcal{C}_{\text{cal}}$	$\mathcal{C}_{\text{str}}$	$\mathcal{C}_{\text{cal}}$	$\mathcal{C}_{\text{str}}$
Neural slack		0.00514 ± 0.00054	0/3,000	0/3,000	0/3,000	0/3,000	9/660	2/660
Penalty		0.00466 ± 0.00054	8/3,000	30/3,000	30/3,000	41/3,000	36/660	49/660
Neural ALM	joint	0.00424 ± 0.00017	60/3,000	606/3,000	221/3,000	674/3,000	41/660	264/660
	PDL-1	0.00424 ± 0.00009	103/3,000	641/3,000	240/3,000	703/3,000	80/660	285/660
	PDL-5	0.00433 ± 0.00048	89/3,000	421/3,000	217/3,000	436/3,000	25/660	111/660

5 Conclusion

Neural slack variables enforce shape constraints by pairing the primary network with an auxiliary network that is feasible by construction and jointly trained as a target for the constraint profile of f^θ. The construction alleviates the drift, oscillation and overshoot of penalty and primal-dual methods (Figure 1); removing the data loss in the barrier experiment isolates the cause (Section 4.2). In our experiments, neural slack variables can improve feasibility, in particular where the capacity restriction in constraint space helps, as for a SIREN primary network on a fixed grid (Figure 2). In the convexity benchmark, neural slack variables improve feasibility over the penalty method at comparable data error, and derivative matching (the H^1 slack loss) improves it further, to every seed up to $d = 4$. On implied volatility surfaces (Section 4.3), a high-profile open challenge in deep learning for quantitative finance, they are the only method in Table 2 without measured violations on the fitted and sampled surfaces, at a higher data error than the penalty method and neural augmented Lagrangian.

The neural slack architecture is an explicit inductive bias on the constraint profile, where low error requires sufficient capacity and mobility of s^ϕ, while excessive mobility costs feasibility (Figure 3a). Because differential constraint operators amplify high frequencies, the neural slack variable typically needs more spectral expressiveness than f^θ.

Limitations Learned constraint enforcement on a sampled grid scales poorly with input dimension, which neural slack variables do not avoid (Section 4.1.2). The neural slack variable increases parameter count and training time, and adds a substantial tuning surface, as its architecture and learning rate can sensitively impact accuracy and feasibility (Section 4.1.1). In the data-free barrier experiment, only 3 of 50 slack configurations are feasible on every seed (Section 4.2). Neural slack variables carry no feasibility guarantee of their own, and all feasibility results are measured on dense evaluation grids. The off-grid bounds of Section C.2 are not specific to neural slack variables, and with Lipschitz constants estimated on a grid they give numerical evidence rather than a certificate. Architectural constraints therefore remain the appropriate default where available and applicable.

Acknowledgments and Disclosure of Funding

RW is grateful to Nicola Muça Cirone for discussions that encouraged the writing of this paper. AJ gratefully acknowledges financial support from the EPSRC grants EP/W032643/1 and EP/T032146/1. RW was supported by the Department of Mathematics, Imperial College, through the Roth scholarship scheme. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this work. The authors declare no competing interests.

References

Damien Ackerer, Natasa Tagasovska, and Thibault Vatter. Deep smoothing of the implied volatility surface. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11552–11563. Curran Associates, Inc., 2020.

Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *17th European Control Conference, ECC 2019, Naples, Italy, June 25-28, 2019*, pages 3420–3431. IEEE, 2019. doi: 10.23919/ECC.2019.8796030.

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 146–155. PMLR, 2017.

Alexandre Araujo, Aaron J. Havens, Blaise Delattre, Alexandre Allauzen, and Bin Hu. A unified algebraic perspective on lipschitz neural networks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

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: jfds.2022.1.093, April 2022. doi: 10.3905/jfds.2022.1.093.

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 600–609. PMLR, July 2018.

Hans Buehler, Blanka Horvath, Anastasis Kratsios, Yannick Limmer, and Raeid Saqur. SANOS (Smooth strictly Arbitrage-free Non-parametric Option Surfaces), February 2026.

Luiz F. O. Chamon and Alejandro Ribeiro. Probably approximately correct constrained learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*, 2020.

Luiz F. O. Chamon, Santiago Paternain, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained learning with non-convex losses. *IEEE Transactions on Information Theory*, 69(3):1739–1760, 2023. doi: 10.1109/TIT.2022.3187948.

Marc Chataigner, Stéphane Crépey, and Matthew Dixon. Deep Local Volatility. *Risks*, 8(3):82, August 2020. ISSN 2227-9091. doi: 10.3390/risks8030082.

Namkyeong Cho, Junseung Ryu, and Hyung Ju Hwang. Sobolev training for operator learning. *Journal of Computational Physics*, 543:114408, 2025. ISSN 0021-9991. doi: 10.1016/j.jcp.2025.114408.

Guillaume Coiffier and Louis Béthune. 1-lipschitz neural distance fields. *Computer Graphics Forum*, 43(5):i–x, 2024. doi: 10.1111/CGF.15128.

Wojciech M. Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4278–4287, 2017.

Hongkai Dai, Benoit Landry, Marco Pavone, and Russ Tedrake. Counter-example guided synthesis of neural network Lyapunov functions for piecewise linear systems. In *59th IEEE Conference on Decision and Control, CDC 2020, Jeju Island, South Korea, December 14-18, 2020*, pages 1274–1281. IEEE, 2020. doi: 10.1109/CDC42340.2020.9304201.

Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. In Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh, editors, *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021. doi: 10.15607/RSS.2021.XVII.063.

Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023. doi: 10.1109/TRO.2022.3232542.

Fabrice Deschâtres. Convex Volatility Interpolation, May 2024.

Alec Edwards, Andrea Peruffo, and Alessandro Abate. Fossil 2.0: Formal certificate synthesis for the verification and control of dynamical models. In Erika Ábrahám and Manuel Mazo Jr., editors, *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2024, Hong Kong SAR, China, May 14-16, 2024*, pages 26:1–26:10. ACM, 2024. doi: 10.1145/3641513.3651398.

Jose Gallego-Posada, Juan Ramirez, Meraj Hashemizadeh, and Simon Lacoste-Julien. Cooper: A Library for Constrained Optimization in Deep Learning, April 2025.

Jim Gatheral and Antoine Jacquier. Arbitrage-free SVI volatility surfaces. *Quantitative Finance*, 14 (1):59–71, January 2014. ISSN 1469-7688. doi: 10.1080/14697688.2013.819986.

Rahul Golder, Bimol Nath Roy, and M. M. Faruque Hasan. DAE-HardNet: A Physics Constrained Neural Network Enforcing Differential-Algebraic Hard Constraints, December 2025.

Pieter-Jan Hoedt and Günter Klambauer. Principled weight initialisation for input-convex neural networks. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Ignacio Hounie, Alejandro Ribeiro, and Luiz F. O. Chamon. Resilient constrained learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Ashfaq Iftakher, Rahul Golder, Bimol Nath Roy, and M.M. Faruque Hasan. Physics-informed neural networks with hard nonlinear equality and inequality constraints. *Computers & Chemical Engineering*, 204:109418, 2026. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2025.109418.

Peter Jäckel. Let's Be Rational: Let's Be Rational. *Wilmott*, 2015(75):40–53, January 2015. ISSN 15406962. doi: 10.1002/wilm.10395.

James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4475–4482. ijcai.org, 2021. doi: 10.24963/IJCAI.2021/610.

Vladimir Lucic. Normalizing Volatility Transforms and Parameterization of Volatility Smile, April 2021.

Youngjae Min and Navid Azizan. HardNet: Hard-Constrained Neural Networks with Universal Approximation Guarantees, October 2025.

Harikrishna Narasimhan, Andrew Cotter, Yichen Zhou, Serena Lutong Wang, and Wenshuo Guo. Approximate heavily-constrained learning with lagrange multiplier models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*, 2020.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2. ed edition, 2006. ISBN 978-0-387-30303-1.

Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 165–174. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00025.

Seonho Park and Pascal Van Hentenryck. Self-supervised primal-dual learning for constrained optimization. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 4052–4060. AAAI Press, 2023. doi: 10.1609/AAAI.V37I4.25520.

John C. Platt and Alan H. Barr. Constrained differential optimization. In Dana Z. Anderson, editor, *Neural Information Processing Systems, Denver, Colorado, USA, 1987*, pages 612–621. American Institue of Physics, 1987.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron C. Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, pages 5301–5310. PMLR, 2019.

Sameera Ramasinghe and Simon Lucey. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXIII*, Lecture Notes in Computer Science, pages 142–158. Springer, 2022. doi: 10.1007/978-3-031-19827-4_9.

Juan Ramirez, Meraj Hashemizadeh, and Simon Lacoste-Julien. Position: Adopt constraints over penalties in deep learning. *CoRR*, abs/2505.20628, 2025. doi: 10.48550/ARXIV.2505.20628.

Davor Runje and Sharath M. Shankaranarayana. Constrained monotonic neural networks. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 29338–29353. PMLR, 2023.

Davide Sartor, Alberto Sinigaglia, and Gian Antonio Susto. Advancing constrained monotonic neural networks: Achieving universal approximation beyond bounded activations. In *Forty-Second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*, 2020.

Sarath Sivaprasad, Ankur Singh, Naresh Manwani, and Vineet Gandhi. The curious case of convex neural networks. In Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and Jose A. Lozano, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 738–754, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86486-6.

Motahareh Sohrabi, Juan Ramirez, Tianyue H. Zhang, Simon Lacoste-Julien, and Jose Gallego-Posada. On PI controllers for updating lagrange multipliers in constrained optimization. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Forty-First International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 45922–45954. PMLR / OpenReview.net, 2024.

Mihaela C. Stoian, Salijona Dyrmishi, Maxime Cordy, Thomas Lukasiewicz, and Eleonora Giunchiglia. How realistic is your synthetic data? Constraining deep generative models for tabular data. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID lagrangian methods. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, pages 9133–9143. PMLR, 2020.

Ruben Wiedemann, Antoine Jacquier, and Lukas Gonon. Operator deep smoothing for implied volatility. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Yongxin Yang, Wenqi Chen, Chao Shu, and Timothy M. Hospedales. HyperIV: Real-time implied volatility smoothing. In *Forty-Second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net, 2025.

Amir Zadeh, Yao-Chong Lim, Paul Pu Liang, and Louis-Philippe Morency. Variational Auto-Decoder: A Method for Neural Generative Modeling from Incomplete Data, January 2021.

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*, pages 3968–3975, 2021.

A Neural slack variables: training and implementation details

A.1 Slack geometry transform and symmetric relative error

We motivate the slack geometry transform

$$T_\sigma(z) := \begin{cases} \sigma \operatorname{asinh}(z/\sigma), & z \geq 0, \\ z, & z < 0, \end{cases} \qquad \sigma > 0.$$

The identity branch $z < 0$ avoids attenuating the correction of deep violations. On the feasible side $z \geq 0$, the asinh branch relates to the symmetric relative error. For $x, y \geq 0$ define

$$A_\sigma(x, y) := T_\sigma(x) - T_\sigma(y)$$

and the regularised symmetric relative error

$$S_\sigma(x, y) := \frac{x - y}{\frac{1}{2}\left(\sqrt{\sigma^2 + x^2} + \sqrt{\sigma^2 + y^2}\right)}.$$

We have

$$\frac{1}{2} S_\sigma(x, y) = \tanh\left(\frac{A_\sigma(x, y)}{2\sigma}\right).$$

For positive arguments the proposed transform therefore gives rise to a monotone rescaling of the symmetric relative error. Its residual gives less weight to differences at large profile values but remains unbounded.

The scale parameter σ controls the attenuation. For $x = y + \Delta$ with $\Delta \to 0$,

$$\frac{\rho}{2} A_\sigma(y + \Delta, y)^2 = \frac{\rho \sigma^2}{2(\sigma^2 + y^2)} \Delta^2 + \mathcal{O}(\Delta^3).$$

At agreement, the transformed loss therefore behaves locally like a quadratic error with the effective weight

$$\rho_{\text{eff}}(y) := \frac{\rho \sigma^2}{\sigma^2 + y^2}.$$

Near zero, ρ retains its interpretation from the untransformed slack loss:

$$\lim_{y \to 0} \rho_{\text{eff}}(y) = \rho.$$

Reducing σ strengthens the attenuation of differences at large slack values. As $\sigma \to \infty$, the transform recovers the ordinary L^2 slack error with weight ρ.

A.2 Training algorithm

Algorithm 1 Neural slack variable training

1: *Input:* dataset $\mathcal{D}$; constraint operator $\mathcal{C}$; primary network $f^\theta \colon \Omega \to \mathbb{R}$; neural slack variable $s^\phi \colon \Omega \to K$; data batch size B_{d}; constraint batch size B_{g}; weight ρ; slack transform T; iterations N_{iter}.
2: *Initialise:* $\theta^{(0)}, \phi^{(0)}$.
3: **for** $t = 1, \ldots, N_{\text{iter}}$ **do**
4: Sample $\{(x_n, y_n)\}_{n=1}^{B_{\text{d}}} \subset \mathcal{D}$ and $\{\xi_n\}_{n=1}^{B_{\text{g}}} \sim \mathcal{U}(\Omega)$.
5: Evaluate $c_n \leftarrow \mathcal{C}[f^\theta](\xi_n)$ and $s_n \leftarrow s^\phi(\xi_n)$.
6: $\mathcal{L}_{\text{data}} \leftarrow \frac{1}{B_{\text{d}}} \sum_{n=1}^{B_{\text{d}}} \ell(f^\theta(x_n), y_n)$.
7: $\mathcal{L}_{\text{slack}} \leftarrow \frac{1}{2B_{\text{g}}} \sum_{n=1}^{B_{\text{g}}} \|T(c_n) - T(s_n)\|_2^2$.
8: Optimiser step on $\mathcal{L}_{\text{data}} + \rho\, \mathcal{L}_{\text{slack}}$ jointly in (θ, ϕ).
9: **end for**

Algorithm 1 gives the pointwise, or H^0, slack loss. The Sobolev extension is given in Section 3.

A.3 Gradient flow analysis of joint training

Capacity Architecture and width determine which constraint profiles the slack can represent. If its capacity is insufficient, matching the primary constraint profile to the slack class can lead to poor δ_{mae} (biased relaxation of the constrained learning problem). The width-one MLP result in Figure 3a illustrates this capacity limitation.

Response during optimisation Once the slack has adequate capacity, the remaining question is whether it can follow the evolving constraint profile within the training budget. On the grid $\pi = \{x_i\}_{i=1}^{I}$, write $c = (c_\theta(x_i))_{i=1}^{I}$, $s = (s^\phi(x_i))_{i=1}^{I}$, and $r = c - s$. Consider

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{data}}(\theta) + \frac{\rho}{2}\|r\|^2.$$

For the Jacobians $J_c = \frac{\partial c}{\partial \theta} \in \mathbb{R}^{I \times P}$, $J_s = \frac{\partial s}{\partial \phi} \in \mathbb{R}^{I \times Q}$ and learning rates α_θ, α_ϕ, define

$$A_c = \alpha_\theta J_c J_c^\top, \qquad A_s = \alpha_\phi J_s J_s^\top, \qquad v_{\text{data}} = -\alpha_\theta J_c \nabla_\theta \mathcal{L}_{\text{data}}.$$

The gradient flow dynamics are

$$\dot{c} = v_{\text{data}} - \rho A_c r, \qquad \dot{s} = \rho A_s r, \qquad \dot{r} = v_{\text{data}} - \rho(A_c + A_s) r.$$

We assume locally that the residual adjusts faster than the Jacobians and data gradient change, so $\dot{r} \approx 0$. Intuitively, we expect this regime after the initial fitting phase: the primary network moves gradually under the data loss, while the slack remains close to the evolving constraint profile.

Let u be a common eigenvector with $A_c u = a_c u$, $A_s u = a_s u$, and write $r_u = u^\top r$, $\dot{c}_u = u^\top \dot{c}$, $v_u = u^\top v_{\text{data}}$. Then

$$r_u \approx \frac{v_u}{\rho(a_c + a_s)}, \qquad \dot{c}_u \approx \frac{a_s}{a_c + a_s} v_u.$$

Since $\dot{s}_u = \rho a_s r_u$, larger a_s means that the slack responds more strongly to a given residual. This reduces the residual and allows the primary network to follow the data-loss update more closely. The slack learning rate scales this response directly. Increasing width often also raises the response, but also changes the slack class.

Interaction with the constraint operator The same calculation suggests why neural slack training can favour smoother constraint profiles, even when the slack class is expressive (Figure 2). Ignoring boundary effects, let u_ω denote a spatial Fourier mode on the grid. For a linear constraint $c_\theta = \mathcal{C} f^\theta$, let J_f denote the Jacobian of the sampled primary output, and let $k_f(\omega)$ and $k_s(\omega)$ be the Fourier eigenvalues of $J_f J_f^\top$ and $J_s J_s^\top$ at u_ω. If $\mathcal{C}$ has Fourier multiplier $\ell(\omega)$, then

$$a_c(\omega) = \alpha_\theta |\ell(\omega)|^2 k_f(\omega), \qquad a_s(\omega) = \alpha_\phi k_s(\omega).$$

Applying the preceding result to u_ω gives $\dot{c}_\omega \approx H(\omega) v_\omega$, where

$$H(\omega) \approx \frac{a_s(\omega)}{a_c(\omega) + a_s(\omega)} = \frac{\alpha_\phi k_s(\omega)}{\alpha_\theta |\ell(\omega)|^2 k_f(\omega) + \alpha_\phi k_s(\omega)}.$$

For monotonicity, $\mathcal{C} = \partial_x$ and $|\ell(\omega)|^2 = \omega^2$. When $k_f(\omega) \approx k_s(\omega)$,

$$H(\omega) \approx \frac{\alpha_\phi}{\alpha_\theta \omega^2 + \alpha_\phi}.$$

Thus, high-frequency change induced by data fitting is more strongly opposed than low-frequency change.

The slack loss compares the image $c_\theta = \mathcal{C}[f^\theta]$ of the primary network under the constraint operator with s^ϕ, which is parameterised directly in constraint space. For derivative-based constraints, $\mathcal{C}$ amplifies frequency content, so the matched-architecture default via $k_f(\omega) \approx k_s(\omega)$ creates a spectral mismatch in constraint space and the smoothing bias described by $H(\omega)$. Width controls frequency content only indirectly. Architectures designed to represent high-frequency signals, such as SIRENs, provide such a control.

B Related work

B.1 Constrained learning

Theory The PAC-learning framework gives distributional guarantees for constrained learning through Lagrangian duality [Chamon and Ribeiro, 2020, Chamon et al., 2023]. In this setting, Hounie et al. [2023] study adaptive constraint relaxation, with slack adjusted during training according to a user-specified relaxation cost. Neural slack variables also relax constraints, but through a learned slack network rather than per-sample slack updates.

Constrained optimisation learning A related line of work uses neural networks to amortise finite-dimensional constrained optimisation, where the network learns a surrogate solver from problem instances to optimal decisions [Kotary et al., 2021]. Our setting instead trains one function under functional inequality constraints, often involving derivatives, over a continuous domain. Park and Hentenryck [2023] also learn primal and dual networks for instance-amortised constrained optimisation. The closest connection to our method is their target-based dual training, where the dual network is trained toward a projected update rather than only through a violation penalty. Neural slack variables make a related move on the primal side: they learn an auxiliary target and match the constraint operator to it, giving gradient signal beyond the currently violated set.

Projection-based hard constraint methods Several recent works enforce hard constraints through differentiable projection or solver layers [Min and Azizan, 2025, Iftakher et al., 2026, Golder et al., 2025]. These methods resemble neural slack variables because both form explicit feasible targets, but they address different settings. Projection-based methods typically use fixed, non-learnable operators and often require an optimisation or solve inside the forward pass. Neural slack variables instead target functional inequality constraints with a learnable slack network, where the joint learning of the matching objective replaces the explicit projection.

Related neuro-symbolic work also uses symbolic constraint descriptions to enforce linear inequalities in generative models for tabular data [Stoian et al., 2024]. Such approaches suggest a route toward more general hard-constraint mechanisms, which remain limited for many applied constrained learning problems.

Sobolev training When the constraint operator $\mathcal{C}[\cdot]$ involves derivatives (monotonicity via first derivatives, convexity via Hessian spectra), neural slack variables are related to Sobolev training [Czarnecki et al., 2017, Cho et al., 2025]. Both introduce derivative-based matching terms, but neural slack variables learn the target profile s^ϕ jointly with f^θ rather than taking it from derivative labels.

B.2 Primal–dual training with neural multipliers

Constraint and multiplier representation As in Section 3, let $c_\theta := \mathcal{C}[f^\theta] \colon \Omega \to \mathbb{R}^m$ denote the constraint profile, with feasibility defined by $c_\theta(x) > 0$. During training, the margin $\varepsilon > 0$ provides a strict-feasibility buffer through the stronger condition $c_\theta(x) \geq \varepsilon$. Define its signed residual as $g_\theta(x) := \varepsilon - c_\theta(x)$, so that the buffered constraint is $g_\theta(x) \leq 0$. The corresponding Lagrangian is

$$\mathcal{L}(\theta, \lambda) = \mathcal{L}_{\text{data}}(\theta) + \mathbb{E}_{x \sim \mathcal{U}(\Omega)}[\langle \lambda(x), g_\theta(x) \rangle], \qquad \lambda(x) \geq 0. \tag{8}$$

On a constraint grid $\pi = \{x_i\}_{i=1}^I$, a dense representation assigns an independent multiplier vector λ_i to every grid point. The primal objective, the representation of λ, and the optimisation scheme together determine the resulting primal–dual method.

Augmented Lagrangian For inequality constraints, a standard augmented Lagrangian is

$$\mathcal{L}_\rho^{\text{aug}}(\theta, \lambda) = \mathcal{L}_{\text{data}}(\theta) + \mathbb{E}_{x \sim \mathcal{U}(\Omega)} \left[\frac{\|[\lambda(x) + \rho g_\theta(x)]_+\|_2^2 - \|\lambda(x)\|_2^2}{2\rho} \right], \tag{9}$$

where the positive part is applied componentwise and $\rho > 0$ is the augmenting coefficient. An augmented Lagrangian iteration approximately minimises (9) at the current multiplier and then applies the projected update

$$\theta_{k+1} \approx \arg\min_\theta \mathcal{L}_\rho^{\text{aug}}(\theta, \lambda_k), \tag{10}$$

$$\lambda_{k+1}(x) = [\lambda_k(x) + \rho g_{\theta_{k+1}}(x)]_+. \tag{11}$$

For dense multipliers, (11) is evaluated independently at every sampled constraint.

Neural multipliers When the constraint domain is large or continuous, the multiplier can instead be represented by a nonnegative neural function $\lambda_\phi \colon \Omega \to \mathbb{R}^m_{\geq 0}$. Narasimhan et al. [2020] introduce this construction for heavily constrained learning, with a feature vector identifying each constraint. The shared parameters couple multiplier values across constraints, so the multiplier model can generalise between sampled locations while its capacity determines which spatial variation it can represent. Updating the multiplier therefore becomes an optimisation problem in ϕ rather than an independent assignment at each x.

Direct gradient descent–ascent One option is to optimise the primal and multiplier parameters directly by gradient descent–ascent on a common objective $\mathcal{A}$:

$$\theta_{t+1} = \theta_t - \alpha_\theta \nabla_\theta \mathcal{A}(\theta_t, \phi_t), \qquad \phi_{t+1} = \phi_t + \alpha_\phi \nabla_\phi \mathcal{A}(\theta_t, \phi_t). \tag{12}$$

The theoretical algorithm of Narasimhan et al. [2020] alternates multiplier ascent with a global primal best response at the updated multiplier. Their practical algorithm uses (12) with the ordinary Lagrangian (8), evaluating both gradients at the same iterate. Applying (12) to (9) gives the joint neural augmented Lagrangian method used in our experiments. The change in the multiplier function is determined by the multiplier network's Jacobian, the sampled constraints, and the optimiser state.

Regression to multiplier targets The analytical update (11) also defines a supervised target for the multiplier model,

$$t_k(x) := [\lambda_{\phi_k}(x) + \rho g_{\theta_{k+1}}(x)]_+, \tag{13}$$

which can be approximated by minimising

$$\mathcal{L}_{\text{dual}}(\phi) = \mathbb{E}_{x \sim \mathcal{U}(\Omega)} \left[\|\lambda_\phi(x) - t_k(x)\|_2^2 \right]. \tag{14}$$

An exact fit recovers the projected multiplier update on the sampled domain, while finite capacity and finite optimisation determine its functional approximation. At $\phi = \phi_k$, one gradient step on (14) has the same direction as direct ascent in the multiplier variable of (9), up to scaling. Further regression steps continue to approximate the fixed target t_k, whereas further ascent steps continue the parameter dynamics of (12). A phased implementation first updates θ for one or more passes with λ_{ϕ_k} fixed, producing an approximate solution of the primal subproblem in (10). It then fixes θ_{k+1} and optimises (14) for one or more passes. A frozen copy of λ_{ϕ_k} retains the previous outer multiplier while the parameters of λ_ϕ change during the regression phase. The number of dual passes controls the optimisation allocated to fitting a single projected multiplier update. Park and Hentenryck [2023] use projected multiplier targets in Primal–Dual Learning (PDL), where a primal network maps problem instances to candidate solutions and a dual network maps the same instances to their multipliers. Each outer iteration alternates several primal and dual learning steps, with the dual network trained against targets formed from the previous multiplier estimate and the constraint residual of the updated primal network. Their procedure also adapts the augmenting coefficient according to the progress of feasibility and complementarity residuals. In the functional setting considered here, the multiplier network is indexed by the constraint coordinate and approximates the multiplier field over Ω.

Neural augmented Lagrangian baselines The neural multiplier baselines in Section 4.3 use the augmented Lagrangian objective (9) and multiplier decoders with the same hidden architecture. The joint variant updates the primal and multiplier parameters on every mini-batch according to (12). The phased variants alternate each primal epoch with either one or five complete passes of multiplier regression using (13). The comparison evaluates the two optimisation schemes with a common augmented Lagrangian primal form and matched hidden architectures. In Table 2, every neural augmented Lagrangian variant retains strike violations in HOLDOUT, and five dual passes reduce them relative to one but do not remove them. This is consistent with a finite-capacity multiplier enforcing only aggregates of the pointwise constraints [Narasimhan et al., 2020].

C Evaluation

C.1 Evaluation metrics

We report approximation quality via mean absolute error

$$\delta_{\text{mae}} := \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\left|f^{\theta}(x) - y\right|\right],$$

computed on the full training dataset $\mathcal{D}$.

Constraint satisfaction is measured on a fine evaluation grid π_{eval} covering Ω. For a constraint $\mathcal{C}[f^{\theta}]\colon \Omega \to \mathbb{R}^m$, define the pointwise shortfall

$$\eta^{\theta}(x) := \max_{j=1,\dots,m}\left[-\mathcal{C}_j[f^{\theta}](x)\right]_+.$$

We define

$$\begin{aligned}
\eta_{\text{rate}} &:= \mathbb{E}_{x\sim\pi_{\text{eval}}}\left[\mathbf{1}_{\{\eta^{\theta}(x)>0\}}\right],\\
\eta_{\text{mean}} &:= \mathbb{E}_{x\sim\pi_{\text{eval}}}\left[\eta^{\theta}(x) \;\middle|\; \eta^{\theta}(x) > 0\right],\\
\eta_{\text{max}} &:= \max_{x\in\pi_{\text{eval}}} \eta^{\theta}(x).
\end{aligned}$$

η_{rate} is the fraction of evaluation points where the constraint is violated, η_{mean} the average shortfall over these points, and η_{max} the worst-case shortfall. η_{mean} and η_{max} are defined only for $\eta_{\text{rate}} > 0$, and the average shortfall over the full grid factorises as $\mathbb{E}_{x\sim\pi_{\text{eval}}}[\eta^{\theta}(x)] = \eta_{\text{rate}}\,\eta_{\text{mean}}$.

Feasibility at the final epoch depends on where training stops, so we also measure it over a tail of epochs (the last 1,000 of 10,000 in Table 1). Let θ_t denote the parameters after epoch t, and $\mathcal{T}$ the tail epochs. Writing $\eta^{(t)}_{\text{rate}}$ for the violation rate computed from η^{θ_t}, define

$$\eta_{\text{tail}} := \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}} \mathbf{1}_{\{\eta^{(t)}_{\text{rate}}>0\}}.$$

Each method is trained in N_{tot} runs, of which N_{sat} have $\eta_{\text{rate}} = 0$ and $N_{\text{viol}} = N_{\text{tot}} - N_{\text{sat}}$ are infeasible. We report N_{sat} or N_{viol}, the mean and standard deviation of δ_{mae} over all runs, and the median, minimum and maximum of η_{rate}, η_{mean} and η_{max} over the infeasible runs. Conditioning on violation at both levels separates how often violations occur, $N_{\text{sat}}/N_{\text{tot}}$ and η_{rate}, from their magnitude, η_{mean} and η_{max}.

C.2 Off-grid feasibility

From grid values to the continuous domain Let $c\colon \Omega \to \mathbb{R}$ be a scalar constraint profile, and let $\pi = \{x_i\}_{i=1}^{I} \subset \Omega$ be a finite evaluation grid with fill distance

$$h_\pi := \sup_{x\in\Omega}\min_i \|x - x_i\|.$$

If c is L_c-Lipschitz, then every $x \in \Omega$ has a nearest grid point x_i such that $c(x) \geq c(x_i) - L_c h_\pi$. Consequently,

$$\inf_{x\in\Omega} c(x) \geq \min_i c(x_i) - L_c h_\pi. \tag{15}$$

This method-agnostic bound certifies continuous feasibility whenever the minimum sampled margin is at least $L_c h_\pi$. For quasi-uniform grids, h_π generally scales as $I^{-1/d}$, which makes dense evaluation increasingly expensive with the input dimension. A certified global Lipschitz constant gives a formal conclusion, whereas a constant estimated on a finite grid provides only numerical evidence.

Average slack loss and worst-case violation We relate the average slack loss to the worst-case violation $v := \sup_{x\in\Omega}[-c(x)]_+$. Let $r := c - s^{\phi}$ be L_r-Lipschitz, let $\underline{s} \leq \inf_x s^{\phi}(x)$, and let $\widehat{M}_\pi := I^{-1}\sum_i r(x_i)^2$. For a radius $R \geq 0$, let $I_\pi(R) := \min_j \#\{i : \|x_i - x_j\| \leq R\}$ be the smallest number of grid points within radius R of a grid point. Then

$$v \leq \left[\min_{R\geq 0}\left\{L_r(h_\pi + R) + \sqrt{\frac{I\widehat{M}_\pi}{I_\pi(R)}}\right\} - \underline{s}\right]_+.$$

To see this, consider a point with violation v. Its residual has magnitude at least $\underline{s} + v$. At least $I_\pi(R)$ grid points are close enough that their residual magnitudes can be smaller by at most $L_r(h_\pi + R)$. Their squared residuals must appear in the average, which gives the bound.

The term $\underline{s}$ is needed because the slack loss only says how close c and s^ϕ are. It does not say how far the slack lies above zero. Positivity alone gives $\underline{s} = \varepsilon$. A larger value can be obtained if the slack is L_s-Lipschitz: $\underline{s} \geq \min_i s^\phi(x_i) - L_s h_\pi$, which is (15) applied to the slack rather than the constraint profile. Applying (15) directly to c is simpler because it uses the measured constraint margin and avoids converting an average residual into a worst-case residual. Neither bound is specific to neural slack variables.

D Additional details: monotonicity ($d = 1$)

D.1 Setup

Target and data The target is the squared piecewise linear function

$$f(x) = b(x)^2, \qquad b(x) = \begin{cases} (2 - s/3) + 3\,(x + \frac{1}{3}), & -1 \le x < -\frac{1}{3}, \\ 2 + s\,x, & -\frac{1}{3} \le x < \frac{1}{3}, \\ (2 + s/3) + (x - \frac{1}{3}), & \frac{1}{3} \le x \le 1, \end{cases}$$

on $\Omega = [-1, 1]$, where the inner-slope parameter s controls the central segment. The two outer segments retain slopes 3 (left) and 1 (right); their vertical anchors are chosen so b is continuous at $x = \pm\frac{1}{3}$ for any s. With $s = 0$ the central segment is a flat plateau at $f \equiv 4$. We use $s = -0.1$, tilting the plateau into a slight downward terrace; this violates $\partial_x f \ge 0$ in the ground truth, so the constraint actively opposes the data fit.

The training dataset consists of $N = 100$ noisy observations on a uniform grid over Ω, with additive zero-mean Gaussian noise (standard deviation 0.1). The constraint is enforced on a separate uniform training grid π_{train} of 200 points covering $[-1.05, 1.05]$ (a small 0.05 pad past the data domain) and verified on an evaluation grid π_{eval} of 10,000 uniformly spaced points over Ω. Initialisation seeds and noise seeds are paired across runs; exact reproduction is possible from the supplemental code.

Training All runs train full-batch for 10,000 epochs with Adam at PyTorch default settings (learning rate $\alpha_\theta = 10^{-3}$), global gradient clipping at norm 10, and the margin $\varepsilon = 10^{-2}$. In Table 1, the MLP f^θ has four hidden layers of width 16 with $\mathrm{softplus}$ activations, and its learning rate is cosine-annealed to half its value over the last 1,000 epochs. The SIREN f^θ has three hidden layers of width 16 with $\omega_0 = 15$, and its learning rate is cosine-annealed to zero over the whole run. For both, the neural slack variable is a SIREN with three hidden layers of width 16, $\omega_0 = 5$ and $\alpha_\phi/\alpha_\theta = 1/2$, as in Figure 3b. This configuration was fixed independently of the slack selection study of Section D.3; the configuration selected there reaches comparable data error (Table 3).

Comparison methods The penalty method uses the linear hinge penalty (2) with $p = 1$. The Lagrangian method keeps a dense multiplier at every point of π_{train} and takes a projected multiplier-ascent step of size 1 before every primal step. The augmented Lagrangian uses the objective (9) with dense multipliers, updated by (11) before every primal step with the current ρ as step. We use (advanced) Constrained Monotonic Neural Networks (CMNNs) [Sartor et al., 2025]. Following the reference implementation,[2] the first hidden-layer activation is the identity and subsequent layers use a monotone nonlinearity; we use $\mathrm{softplus}$. In Table 1, the constraint weight ρ starts at 1 for the penalty method and at 100 for the augmented Lagrangian and the neural slack variable; it is multiplied by 1.01 at most once every 10 epochs if a violation on π_{train} was measured since its previous increase, up to 100 times its initial value, and held fixed from epoch 9,000.

D.2 Constraint drift

The drift comparison trains the MLP f^θ of Table 1 on one seed without learning rate decay and with constant weights: $\rho = 10^3$ for the exact slack and the augmented Lagrangian and $\rho = 10$ for the penalty method; the other settings follow Section D.1. Figure 6 shows the constraint profiles of the drift comparison over the whole training run, and Figure 5 their minimum against the matched margin over the last 100 epochs.

[2] `https://github.com/AMCO-UniPD/monotonic`

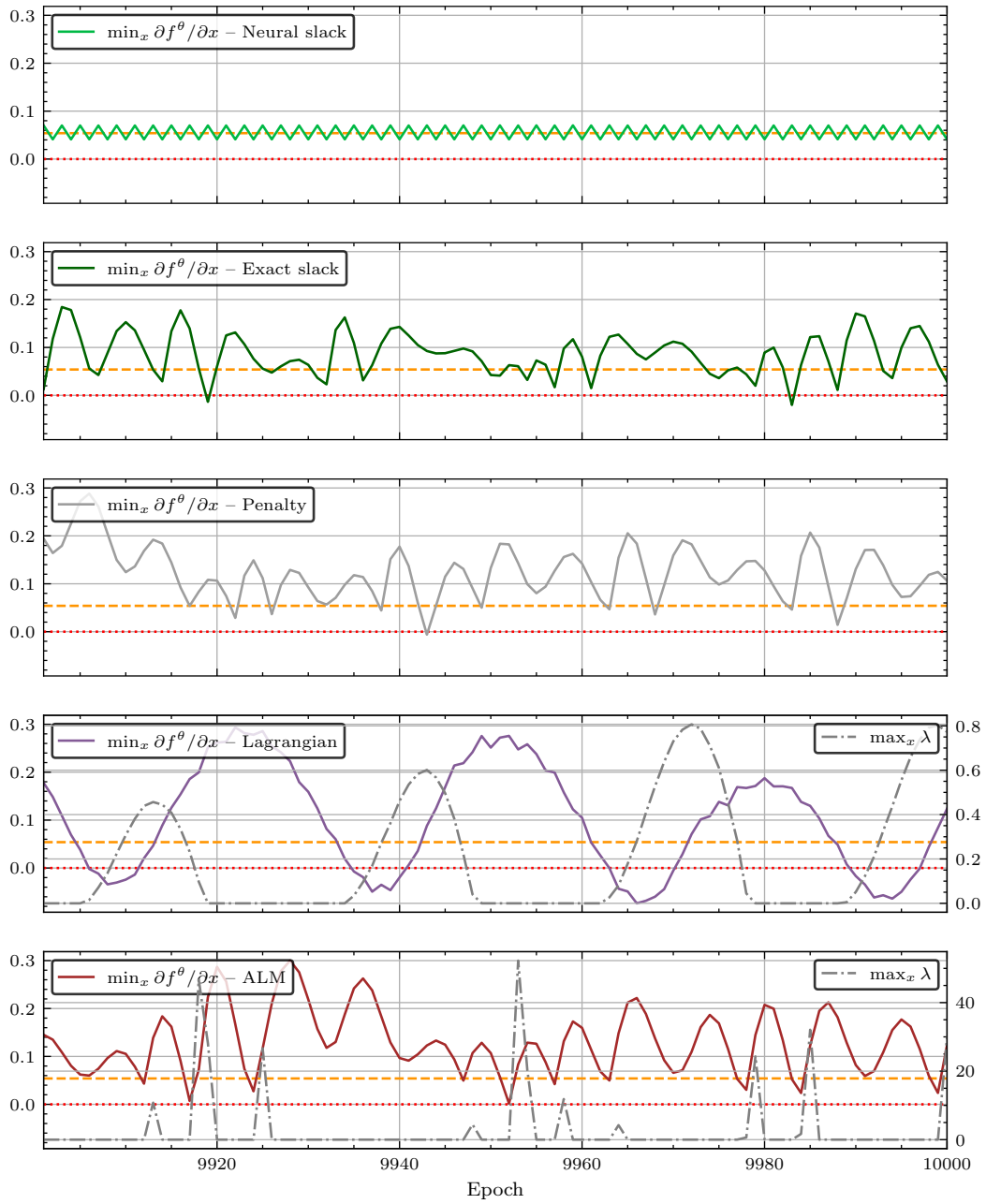


Figure 5: Minimum of $\partial_x f^{\theta_t}$ over $[-0.3, 0.3]$ at each epoch t of the last 100 epochs of the drift comparison. The dashed line marks the matched margin of step t, the minimum $\min_x s^{\phi_{t-1}}$ of the neural slack before its own step t, and the dotted line marks zero. For the Lagrangian and the augmented Lagrangian, the dash-dotted line shows the largest multiplier on the right axis.

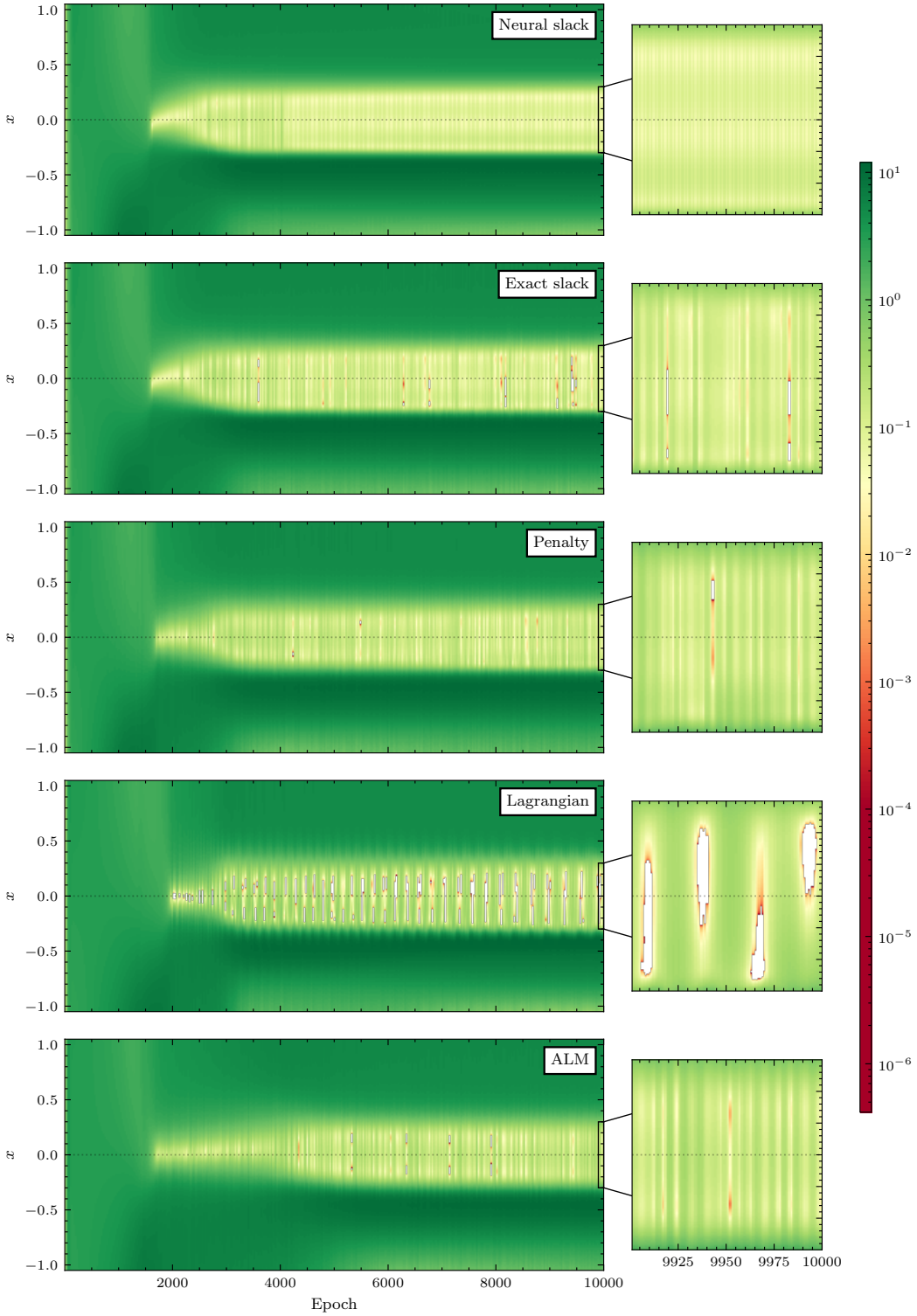


Figure 6: Constraint profile $\partial_x f^\theta(x)$ over training for the drift comparison of Section 4.1.1, with the last 100 epochs on $[-0.3, 0.3]$ magnified on the right. White regions mark violations; feasible values are colourmapped on a logarithmic scale.

D.3 Slack selection

These sweeps use the MLP f^θ of Table 1, five seeds and a constant weight ρ.

Width and learning rate Table 4 lists the sweep of Figure 3a over the slack architecture, the width W and the ratio $\alpha_\phi/\alpha_\theta$ at $\rho = 10^3$ and $\sigma = 0.01$. We select the cell with the lowest δ_{mae} among those feasible on every seed, a SIREN slack with $W = 4$ and $\alpha_\phi/\alpha_\theta = 1/4$ at $\delta_{\text{mae}} = 0.092$. The matched default, an MLP slack with the architecture of f^θ at $\alpha_\phi/\alpha_\theta = 1$, is feasible on every seed at $\delta_{\text{mae}} = 0.099$.

Pointwise EMA slack Table 6 replaces the neural slack variable by the pointwise exponential moving average $s_t = (1-\alpha)\, s_{t-1} + \alpha\, \Pi_K(c_{\theta_t})$ on π_{train}, with $\alpha = 1 - 2^{-1/h}$ for a half-life h, as the target of the same slack loss. The half-life $h = 0$ gives the exact slack, and $h = \infty$ the fixed target $s \equiv \varepsilon + 1$. All runs use the margin $\varepsilon = 10^{-2}$. We cross the untransformed slack error ($\sigma = \infty$) and the transform T_σ at $\sigma = 0.01$ with the weights $\rho \in \{10, 10^2, 10^3\}$ (Figure 3b shows the exact slack and the neural slack variable over this grid). At $\sigma = 0.01$ and $\rho = 10^3$, the setting of the neural slack selection, the half-life with the lowest δ_{mae} among those feasible on every seed is $h = 1$. On this fixed grid, the untransformed exact slack, i.e. the quadratic penalty, at $\rho = 10^2$ and the fixed target with the transform at $\rho = 10$ are also feasible on every seed, at a fit comparable to the selected neural slack variable (Table 4).

Fresh seeds Table 3 evaluates the selected neural slack (SIREN, $W = 4$, $\alpha_\phi/\alpha_\theta = 1/4$), the matched default (MLP, $W = 16$, $\alpha_\phi/\alpha_\theta = 1$) and the selected pointwise EMA slack ($h = 1$) on five further seeds. All three are feasible on every seed. The paired difference in δ_{mae} to the pointwise EMA slack, with its 95% confidence interval, is $-0.0002\ [-0.0017, 0.0013]$ for the selected neural slack and $0.0024\ [0.0012, 0.0037]$ for the matched default.

Table 3: Selected configurations of Section D.3 on five seeds at $\rho = 10^3$ and $\sigma = 0.01$: δ_{mae} per seed and as mean $\pm$ std.

Target	Seed 1	Seed 2	Seed 3	Seed 4	Seed 5	$N_{\text{sat}}/N_{\text{tot}}$	δ_{mae}
Neural slack, matched default	0.0953	0.0911	0.0916	0.0890	0.0839	5/5	0.0902 ± 0.0042
Neural slack, selected	0.0914	0.0885	0.0892	0.0865	0.0823	5/5	0.0876 ± 0.0034
Pointwise EMA, $h = 1$	0.0929	0.0898	0.0880	0.0857	0.0823	5/5	0.0878 ± 0.0040

Table 4: Slack selection sweep of Figure 3a: δ_{mae} as mean $\pm$ std over five seeds, $\eta_{\max}$ as median [min, max] over the infeasible seeds.

$\mathcal{S}$	W	$\alpha_\phi/\alpha_\theta$	$N_{\text{sat}}/N_{\text{tot}}$	δ_{mae}	$\eta_{\max}$
MLP	1	$1/16$	5/5	1.4e−01 ± 6.5e−03	—
		$1/4$	5/5	1.4e−01 ± 6.4e−03	—
		1	5/5	1.4e−01 ± 6.1e−03	—
		4	5/5	1.4e−01 ± 5.8e−03	—
		16	5/5	1.4e−01 ± 7.0e−03	—
	4	$1/16$	5/5	1.4e−01 ± 6.1e−03	—
		$1/4$	5/5	1.1e−01 ± 8.3e−03	—
		1	5/5	9.9e−02 ± 7.9e−03	—
		4	5/5	9.8e−02 ± 1.2e−02	—
		16	5/5	9.6e−02 ± 1.2e−02	—
	16	$1/16$	5/5	1.2e−01 ± 1.1e−02	—
		$1/4$	5/5	9.9e−02 ± 9.9e−03	—
		1	5/5	9.9e−02 ± 1.3e−02	—
		4	5/5	9.5e−02 ± 1.3e−02	—
		16	3/5	4.1e−01 ± 2.9e−01	3.2e−02 [2.9e−02, 3.4e−02]
	64	$1/16$	5/5	9.7e−02 ± 1.0e−02	—
		$1/4$	5/5	9.7e−02 ± 1.2e−02	—
		1	5/5	9.5e−02 ± 1.1e−02	—
		4	4/5	4.1e−01 ± 2.9e−01	6.7e−02
		16	0/5	6.4e−01 ± 2.1e−02	6.0e−02 [1.6e−02, 1.3e−01]
	256	$1/16$	5/5	1.0e−01 ± 1.8e−02	—
		$1/4$	5/5	9.5e−02 ± 1.1e−02	—
		1	4/5	4.1e−01 ± 2.9e−01	7.0e−02
		4	2/5	5.5e−01 ± 1.8e−01	3.7e−02 [3.5e−04, 4.5e−02]
		16	1/5	6.3e−01 ± 1.3e−02	4.0e−02 [3.0e−02, 6.2e−02]
SIREN	1	$1/16$	5/5	1.3e−01 ± 3.7e−03	—
		$1/4$	5/5	1.1e−01 ± 1.6e−02	—
		1	5/5	9.7e−02 ± 1.4e−02	—
		4	5/5	9.9e−02 ± 1.6e−02	—
		16	5/5	1.0e−01 ± 1.9e−02	—
	4	$1/16$	5/5	1.0e−01 ± 9.9e−03	—
		$1/4$	5/5	9.2e−02 ± 1.2e−02	—
		1	5/5	9.4e−02 ± 1.2e−02	—
		4	5/5	9.4e−02 ± 1.3e−02	—
		16	5/5	1.3e−01 ± 1.2e−02	—
	16	$1/16$	5/5	9.2e−02 ± 1.2e−02	—
		$1/4$	5/5	9.4e−02 ± 1.2e−02	—
		1	5/5	9.4e−02 ± 1.3e−02	—
		4	5/5	9.5e−02 ± 1.2e−02	—
		16	5/5	1.2e−01 ± 1.4e−02	—
	64	$1/16$	5/5	9.4e−02 ± 1.2e−02	—
		$1/4$	5/5	9.5e−02 ± 1.4e−02	—
		1	5/5	9.3e−02 ± 1.0e−02	—
		4	5/5	9.9e−02 ± 1.0e−02	—
		16	0/5	3.2e−01 ± 1.7e−01	3.2e−02 [5.9e−03, 5.1e−02]
	256	$1/16$	5/5	9.5e−02 ± 1.3e−02	—
		$1/4$	5/5	9.3e−02 ± 1.1e−02	—
		1	5/5	9.9e−02 ± 1.2e−02	—
		4	4/5	3.2e−01 ± 1.6e−01	1.9e−02
		16	1/5	6.3e−01 ± 1.3e−02	4.2e−02 [4.0e−02, 6.1e−02]

Table 5: Slack error geometry sweep of Figure 3b, where $\sigma = \infty$ denotes the identity transform: δ_{mae} as mean $\pm$ std over five seeds, $\eta_{\max}$ as median [min, max] over the infeasible seeds.

Method	σ	ρ	$N_{\text{sat}}/N_{\text{tot}}$	δ_{mae}	$\eta_{\max}$
Neural slack	∞	1	1/5	8.8e−02 ± 8.4e−03	2.3e−02 [5.5e−03, 6.8e−02]
		10	5/5	1.2e−01 ± 5.4e−03	—
		100	5/5	3.8e−01 ± 6.0e−02	—
		1000	5/5	5.3e−01 ± 1.6e−02	—
		10000	5/5	6.0e−01 ± 3.7e−02	—
	1	1	0/5	8.3e−02 ± 7.3e−03	5.4e−02 [4.5e−02, 9.8e−02]
		10	5/5	9.2e−02 ± 1.1e−02	—
		100	5/5	1.4e−01 ± 1.4e−02	—
		1000	5/5	3.4e−01 ± 7.4e−02	—
		10000	5/5	5.3e−01 ± 1.4e−02	—
	0.1	1	0/5	8.2e−02 ± 7.5e−03	5.2e−02 [4.6e−02, 9.9e−02]
		10	1/5	8.7e−02 ± 9.9e−03	6.2e−03 [4.0e−03, 1.6e−02]
		100	5/5	9.4e−02 ± 8.3e−03	—
		1000	5/5	1.1e−01 ± 3.0e−03	—
		10000	5/5	1.8e−01 ± 1.2e−02	—
	0.01	1	0/5	8.2e−02 ± 7.2e−03	5.4e−02 [4.5e−02, 8.7e−02]
		10	0/5	8.5e−02 ± 8.1e−03	6.8e−03 [5.2e−03, 1.8e−02]
		100	5/5	9.0e−02 ± 1.0e−02	—
		1000	5/5	9.4e−02 ± 1.3e−02	—
		10000	5/5	9.7e−02 ± 8.8e−03	—
	0.001	1	0/5	8.2e−02 ± 7.2e−03	5.9e−02 [5.3e−02, 9.1e−02]
		10	0/5	8.5e−02 ± 7.9e−03	1.4e−02 [1.1e−02, 2.2e−02]
		100	1/5	9.0e−02 ± 1.0e−02	1.8e−03 [5.9e−04, 5.9e−03]
		1000	5/5	9.6e−02 ± 1.0e−02	—
		10000	5/5	9.1e−02 ± 9.4e−03	—
Exact slack	∞	1	0/5	8.2e−02 ± 7.2e−03	5.3e−02 [4.7e−02, 8.5e−02]
		10	0/5	8.5e−02 ± 8.1e−03	7.9e−03 [3.8e−03, 1.5e−02]
		100	5/5	9.1e−02 ± 8.6e−03	—
		1000	4/5	9.5e−02 ± 1.0e−02	5.4e−03
		10000	5/5	9.7e−02 ± 1.0e−02	—
	1	1	0/5	8.2e−02 ± 7.2e−03	5.3e−02 [4.7e−02, 8.5e−02]
		10	0/5	8.5e−02 ± 8.0e−03	5.8e−03 [3.7e−03, 1.5e−02]
		100	5/5	9.0e−02 ± 9.9e−03	—
		1000	5/5	9.3e−02 ± 9.3e−03	—
		10000	5/5	9.8e−02 ± 1.0e−02	—
	0.1	1	0/5	8.2e−02 ± 7.2e−03	5.3e−02 [4.7e−02, 8.5e−02]
		10	0/5	8.5e−02 ± 8.0e−03	7.0e−03 [3.7e−03, 1.5e−02]
		100	5/5	9.0e−02 ± 1.0e−02	—
		1000	5/5	9.4e−02 ± 1.1e−02	—
		10000	5/5	9.8e−02 ± 1.1e−02	—
	0.01	1	0/5	8.2e−02 ± 7.2e−03	5.4e−02 [4.8e−02, 8.6e−02]
		10	0/5	8.5e−02 ± 7.9e−03	6.9e−03 [4.7e−03, 1.6e−02]
		100	4/5	9.0e−02 ± 8.5e−03	1.4e−05
		1000	4/5	9.4e−02 ± 9.6e−03	4.9e−03
		10000	5/5	9.8e−02 ± 1.1e−02	—
	0.001	1	0/5	8.2e−02 ± 7.2e−03	5.9e−02 [5.3e−02, 9.1e−02]
		10	0/5	8.5e−02 ± 7.9e−03	1.4e−02 [1.1e−02, 2.1e−02]
		100	1/5	9.0e−02 ± 1.0e−02	1.9e−03 [7.3e−04, 6.0e−03]
		1000	5/5	9.5e−02 ± 1.1e−02	—
		10000	5/5	9.6e−02 ± 1.0e−02	—

Table 6: Pointwise EMA slack for the monotone MLP f^θ: mean $\pm$ std over 5 seeds. Without the transform at $\rho = 10$, the exact slack and the pointwise EMA slack with $h \leq 16$ are infeasible on every seed, and the pointwise EMA slack is feasible from $h = 64$ at $\delta_{\text{mae}} \geq 0.29$, against 0.12 for the neural slack variable in Figure 3b. At $\rho = 10^2$, the exact slack is feasible without the transform at $\delta_{\text{mae}} = 0.091$, while every smoothed target loses its fit. With the transform at $\rho = 10^2$ and $\rho = 10^3$, the pointwise EMA slack is feasible on every seed for most $h > 0$, at the fit of the neural slack variable. With the transform at $\rho = 10$, only the pointwise EMA slack with $h = 1024$ (on three of five seeds) and the fixed target (on every seed) are feasible. The results suggest that feasibility follows from an inexact projection, obtained by temporal smoothing or by the finite-capacity neural slack, and that without the transform at $\rho = 10$ the spatial coupling of the neural slack obtains it at a lower cost in fit than temporal smoothing.

		$\sigma = \infty$		$\sigma = 0.01$	
ρ		$N_{\text{sat}}/N_{\text{tot}}$	δ_{mae}	$N_{\text{sat}}/N_{\text{tot}}$	δ_{mae}
10	$h = 0$ (exact slack)	0/5	0.085 ± 0.008	0/5	0.085 ± 0.008
	$h = 1$	0/5	0.093 ± 0.010	0/5	0.085 ± 0.008
	$h = 4$	0/5	0.100 ± 0.006	0/5	0.085 ± 0.008
	$h = 16$	0/5	0.137 ± 0.011	0/5	0.085 ± 0.008
	$h = 64$	5/5	0.293 ± 0.018	0/5	0.085 ± 0.008
	$h = 256$	5/5	0.519 ± 0.012	0/5	0.085 ± 0.008
	$h = 1024$	5/5	0.749 ± 0.008	3/5	0.086 ± 0.008
	$h = \infty$ (fixed slack)	5/5	0.994 ± 0.009	5/5	0.087 ± 0.008
10^2	$h = 0$ (exact slack)	5/5	0.091 ± 0.009	4/5	0.090 ± 0.009
	$h = 1$	1/5	0.210 ± 0.017	5/5	0.090 ± 0.010
	$h = 4$	5/5	0.363 ± 0.023	5/5	0.090 ± 0.010
	$h = 16$	5/5	0.497 ± 0.017	4/5	0.090 ± 0.008
	$h = 64$	5/5	0.706 ± 0.011	5/5	0.089 ± 0.010
	$h = 256$	5/5	0.926 ± 0.011	5/5	0.089 ± 0.011
	$h = 1024$	5/5	1.015 ± 0.009	5/5	0.087 ± 0.008
	$h = \infty$ (fixed slack)	5/5	1.049 ± 0.009	5/5	0.093 ± 0.006
10^3	$h = 0$ (exact slack)	4/5	0.095 ± 0.010	4/5	0.094 ± 0.010
	$h = 1$	5/5	0.529 ± 0.020	5/5	0.089 ± 0.009
	$h = 4$	5/5	0.736 ± 0.063	5/5	0.092 ± 0.009
	$h = 16$	5/5	1.023 ± 0.015	5/5	0.093 ± 0.011
	$h = 64$	5/5	1.077 ± 0.014	5/5	0.094 ± 0.012
	$h = 256$	5/5	1.065 ± 0.011	5/5	0.094 ± 0.011
	$h = 1024$	5/5	1.058 ± 0.009	5/5	0.101 ± 0.010
	$h = \infty$ (fixed slack)	5/5	1.055 ± 0.009	5/5	0.129 ± 0.005

D.4 Grid overfitting

Table 7: Monotone with the SIREN f^θ of Table 1 and π_{train} resampled uniformly at each step, over $N_{\text{tot}} = 100$ seeds: δ_{mae} as mean $\pm$ std over all seeds; Tail and Final as in Table 1.

	SIREN f^θ		
Method	δ_{mae}	Tail	Final
Penalty	7.9e-02 ± 7.0e-03	74/100	100/100
NSV	7.6e-02 ± 6.4e-03	100/100	100/100

Table 7 repeats the SIREN column of Table 1 for the penalty method and the neural slack method with π_{train} resampled uniformly at each step. Resampling makes the penalty method feasible at the final epoch on every seed at unchanged δ_{mae}, against 74 of 100 seeds on the fixed grid, but it remains infeasible over the last 1,000 epochs on 26 seeds; the neural slack variable is feasible over the last 1,000 epochs on every seed with either grid.

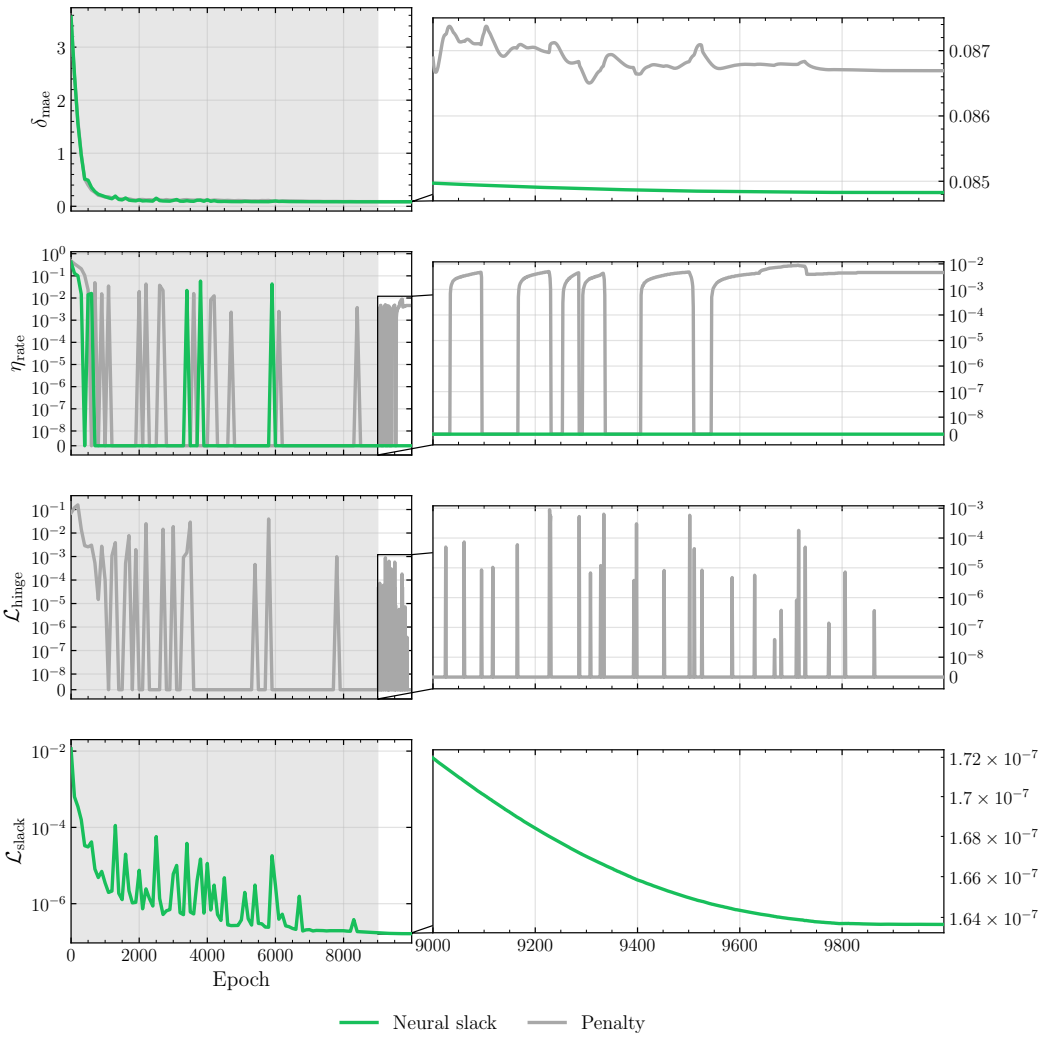


Figure 7: Training dynamics over ten thousand epochs for the monotonicity experiment with a SIREN primary model on the seed of Figure 2, with zoom insets (last 1,000 epochs) on the right. The penalty loss vanishes while violations on π_{eval} persist, whereas the neural slack variable has no violations on π_{eval} over the last 1,000 epochs while its slack loss remains positive.

E Additional details: convexity ($d \in \{2, 3, 4, 5\}$)

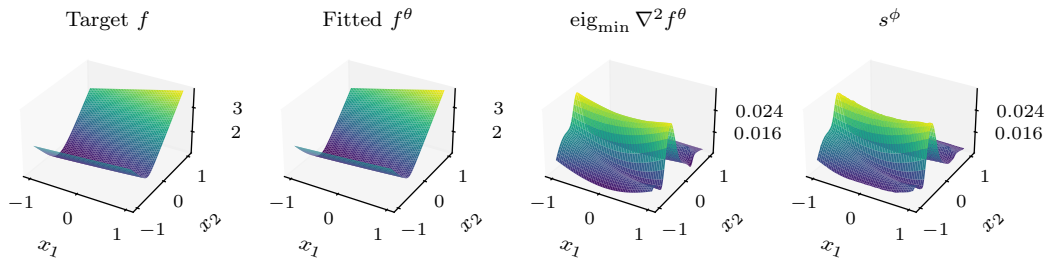


Figure 8: Dimension $d = 2$, neural slack (H^0), seed 3.

E.1 Dataset and target function

The target is the log-sum-exp of $M = 10$ affine functions plus a small quadratic term:

$$f(x) = \tau \log\left(\sum_{k=1}^{M} \exp\left(\frac{a_k^\top x + b_k}{\tau}\right)\right) + \frac{\mu}{2}\|x\|_2^2, \qquad x \in \Omega = [-1, 1]^d.$$

The affine coefficients $a_k \in \mathbb{R}^d$ and offsets $b_k \in \mathbb{R}$ are drawn i.i.d. from $\mathcal{N}(0, I_d)$ and $\mathcal{N}(0, 1)$ per seed; the log-sum-exp temperature is $\tau = 0.2$ and the quadratic weight is $\mu = 10^{-4}$. Training observations are noisy samples drawn as a scrambled Sobol low-discrepancy sequence on Ω (additive zero-mean Gaussian noise, standard deviation 0.05); constraint enforcement uses a separate scrambled Sobol grid on Ω; evaluation uses a denser Sobol Monte Carlo sample.

E.2 Comparison methods

The penalty method uses the linear hinge penalty (2) with $p = 1$, and all methods use the margin $\varepsilon = 10^{-2}$. The constraint weight ρ starts at 1 for the penalty method and at 100 for both neural slack variables; it is multiplied by 1.01 at most once every 10 epochs if a violation on π_{train} was measured since its previous increase, up to 100 times its initial value, and held fixed from epoch 9,000. The Input Convex Neural Network (ICNN) [Amos et al., 2017] has three hidden layers of width 128 with $\mathrm{Softplus}$ activations and a linear output. It trains under the same 10,000-epoch Adam schedule as the other methods.

E.3 Results by dimension

Table 8 reports the metrics of Section C at each d.

Table 8: δ_{mae} as mean $\pm$ std over 5 seeds, violation metrics as median [min, max] over the infeasible seeds, training time in minutes as the minimum over seeds. Apple M3 Pro (MPS); total training time $\sim$17 h.

d	Method	$N_{\text{sat}}/N_{\text{tot}}$	δ_{mae}	η_{rate}	η_{mean}	η_{max}	Time
2	Unconstrained	0/5	4.1e−02 ± 1.1e−03	4.7e−01 [1.3e−01, 6.6e−01]	2.0e−01 [4.0e−02, 4.5e−01]	1.1e+00 [9.0e−02, 2.8e+00]	0.7
	Penalty	5/5	4.2e−02 ± 1.3e−03	—	—	—	7.0
	Neural slack (H^0)	5/5	4.3e−02 ± 2.9e−03	—	—	—	6.8
	Neural slack (H^1)	5/5	4.2e−02 ± 9.0e−04	—	—	—	17.3
	ICNN	5/5	6.0e−02 ± 1.6e−02	—	—	—	0.8
3	Unconstrained	0/5	4.5e−02 ± 9.8e−03	8.8e−01 [6.1e−01, 9.9e−01]	2.2e−01 [1.2e−01, 6.9e−01]	2.2e+00 [1.8e+00, 6.4e+00]	0.7
	Penalty	4/5	4.8e−02 ± 4.8e−03	1.0e−07	1.3e−03	1.3e−03	8.5
	Neural slack (H^0)	4/5	5.0e−02 ± 7.1e−03	7.0e−07	7.9e−04	2.0e−03	8.9
	Neural slack (H^1)	5/5	4.8e−02 ± 7.0e−03	—	—	—	23.8
	ICNN	5/5	7.2e−02 ± 1.8e−02	—	—	—	0.8
4	Unconstrained	0/5	4.5e−02 ± 5.2e−03	8.6e−01 [7.9e−01, 8.9e−01]	4.9e−01 [1.9e−01, 5.4e−01]	4.6e+00 [2.6e+00, 5.8e+00]	0.7
	Penalty	1/5	5.4e−02 ± 8.2e−03	2.5e−07 [1.0e−07, 6.2e−06]	5.9e−03 [9.4e−04, 1.2e−02]	6.9e−03 [9.4e−04, 6.3e−02]	10.8
	Neural slack (H^0)	3/5	5.6e−02 ± 6.5e−03	3.0e−07 [2.0e−07, 4.0e−07]	4.7e−03 [3.9e−03, 5.5e−03]	7.7e−03 [7.7e−03, 7.7e−03]	11.1
	Neural slack (H^1)	5/5	5.3e−02 ± 8.8e−03	—	—	—	30.3
	ICNN	5/5	9.5e−02 ± 2.2e−02	—	—	—	0.8
5	Unconstrained	0/5	5.0e−02 ± 5.4e−03	9.4e−01 [9.3e−01, 9.6e−01]	5.0e−01 [3.3e−01, 6.4e−01]	6.0e+00 [5.2e+00, 8.6e+00]	0.7
	Penalty	0/5	7.2e−02 ± 3.5e−03	1.2e−05 [5.0e−07, 2.8e−05]	4.0e−03 [3.7e−03, 7.1e−03]	1.5e−02 [8.5e−03, 6.4e−02]	13.0
	Neural slack (H^0)	0/5	6.9e−02 ± 6.0e−03	1.0e−06 [5.0e−07, 2.1e−06]	7.6e−03 [3.1e−03, 1.2e−02]	3.2e−02 [7.4e−03, 3.7e−02]	13.5
	Neural slack (H^1)	3/5	6.8e−02 ± 6.7e−03	2.5e−07 [1.0e−07, 4.0e−07]	2.9e−03 [2.6e−03, 3.2e−03]	4.8e−03 [3.2e−03, 6.4e−03]	36.8
	ICNN	5/5	1.3e−01 ± 1.7e−02	—	—	—	0.8

F Additional details: neural barrier certificates

F.1 Domain and constraint geometry

The benchmark is the autonomous polynomial system from FOSSIL's `Barr3` suite [Edwards et al., 2024],[3] $\dot{x}_1 = x_2$, $\dot{x}_2 = -x_1 - x_2 + \frac{1}{3}x_1^3$, on $\Omega = [-3, 2.5] \times [-2, 1]$. The initial set Ω_{i} and unsafe set Ω_{u} are unions of axis-aligned rectangles and disks, lifted verbatim from the FOSSIL reference configuration:

$$\Omega_{\mathrm{i}} = B\big((1.5, 0),\, 0.5\big) \,\cup\, [-1.8, -1.2]\times[-0.1, 0.1] \,\cup\, [-1.4, -1.2]\times[-0.5, 0.1],$$
$$\Omega_{\mathrm{u}} = B\big((-1, -1),\, 0.4\big) \,\cup\, [0.4, 0.6]\times[0.1, 0.5] \,\cup\, [0.4, 0.8]\times[0.1, 0.3],$$

where $B(c, r)$ denotes the closed Euclidean disk. Each component of (7) is trained on its own scrambled Sobol samples, resampled at each training step: 1,000 points on Ω for the condition $\nabla f^\theta \cdot F + \gamma f^\theta \geq 0$, and 400 points on each of Ω_{i} and Ω_{u} for the sign conditions, drawn uniformly by area over their disks and rectangles. The evaluation points π_{eval} are a 1000×1000 lattice on Ω for the first condition and 10^6 Sobol samples on each of Ω_{i} and Ω_{u} for the sign conditions.

Both methods use the constraint weight $\rho = 1$ and the margin $\varepsilon = 10^{-6}$ and train with Adam at learning rate 10^{-3} and $\beta = (0.9, 0.99)$, without gradient clipping. Without a data term, the gradient of the penalty loss scales with the fraction of violated training points and falls by orders of magnitude as training approaches feasibility; the shorter second-moment memory keeps the large early gradients from suppressing the later steps. The loss is then proportional to ρ, which Adam's update is invariant to only without a fixed clipping threshold.

F.2 Slack selection sweep

Table 9 counts the feasible seeds in each cell of Figure 9.

Table 9: Seeds with $c^\theta_{\min} \geq 0$ out of 5 per slack architecture, width W and learning rate ratio $\alpha_\phi/\alpha_\theta$; † marks cells with a diverged seed. Training time as the minimum over the finite runs of each row. The penalty method is feasible on 4/5 seeds, at $\sim$12 s per run. Apple M3 Pro (CPU); total training time $\sim$3.1 h.

		$\alpha_\phi/\alpha_\theta$					
s^ϕ	W	1/16	1/4	1	4	16	Time (s)
tanh-MLP	1	0/5	0/5	0/5	0/5	0/5	22
	4	0/5	0/5	0/5	0/5	0/5	25
	16	0/5	0/5	0/5	0/5	0/5	23
	64	0/5	0/5	0/5	0/5	0/5	32
	256	0/5	0/5	0/5	0/5	0/5†	87
SIREN	1	0/5	0/5	0/5	0/5	0/5	18
	4	0/5	1/5	0/5	0/5	0/5	19
	16	5/5	5/5	0/5	0/5	0/5	23
	64	5/5	3/5	0/5	0/5	0/5†	34
	256	3/5	0/5	0/5	0/5†	0/5†	97

[3] `https://github.com/oxford-oxcav/fossil`

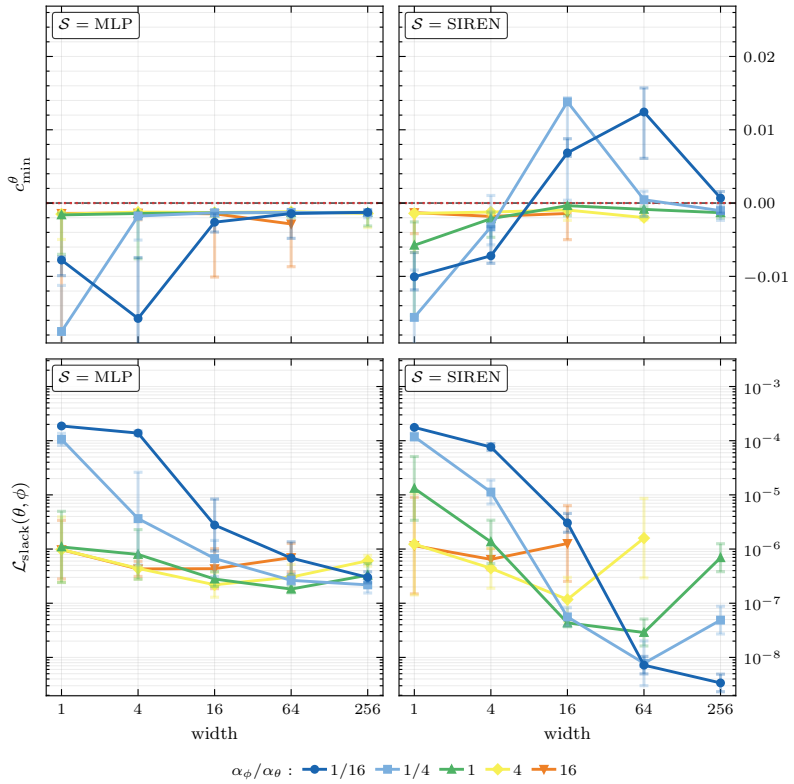


Figure 9: Neural slack variable selection on FOSSIL Barr3, five seeds per cell: $c^{\theta}_{\min}$ (median [min, max]) and $\mathcal{L}_{\text{slack}}$ (geom. mean ± geom. std) across slack architecture, W and $\alpha_\phi/\alpha_\theta$; cells with a diverged seed are omitted. In the $c^{\theta}_{\min}$ panels, the dashed line marks feasibility and the dotted line the margin ε.

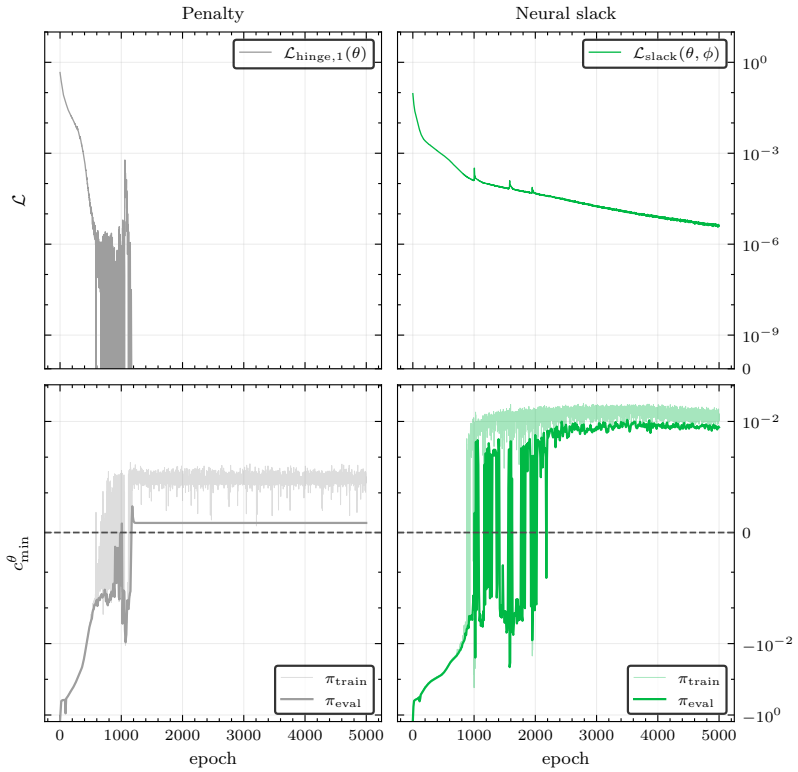


Figure 10: Training trajectories of the penalty method and of neural slack variables (SIREN, $W = 16$, $\alpha_\phi/\alpha_\theta = 1/16$) on FOSSIL Barr3, seed 0: loss (top) and worst constraint value $c^\theta_{\min}$ on the training points π_{train} of each epoch and on the evaluation points π_{eval} every 10 epochs (bottom).

G Additional details: volatility autodecoder

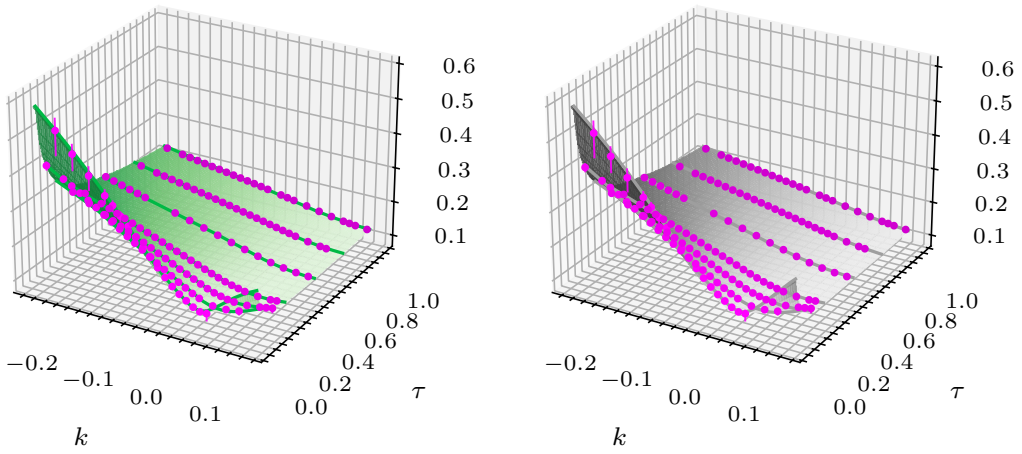


Figure 11: Held-out DJX surface, calibrated with neural slack (left) and penalty (right).

G.1 Options and implied volatility

At a given date t_0, traded options on an equity index give a snapshot

$$\mathcal{O} = \left\{ \left(\tau_i, \left\{ \left(k_{i,j}, y^{\text{bid}}_{i,j}, y^{\text{mid}}_{i,j}, y^{\text{ask}}_{i,j} \right) \right\}_{j=1}^{J_i} \right) \right\}_{i=1}^{I},$$

where

- $0 < \tau_1 < \cdots < \tau_I$ are the times to expiry, $\tau_i = T_i - t_0$ for expiry T_i;
- $k_{i,j} = \log(K_{i,j}/F_{T_i})$ is the log-moneyness of the j-th of the J_i contracts at expiry i, with strike $K_{i,j}$ and forward F_{T_i};
- $y^{\text{bid}}_{i,j} \le y^{\text{mid}}_{i,j} \le y^{\text{ask}}_{i,j}$ are its observed bid, mid and ask implied volatilities.

Implied volatility smoothing fits a continuous, positive surface $(\tau, k) \mapsto f(\tau, k)$ to these observations [Gatheral and Jacquier, 2014, Lucic, 2021]. We measure the calendar and butterfly shape conditions through total volatility $v(\tau, k) := f(\tau, k)\sqrt{\tau}$ and total variance $w := v^2$:

$$\begin{aligned} \mathcal{C}_{\text{cal}}[f] &:= \partial_\tau w, \\ \mathcal{C}_{\text{str}}[f] &:= (1 + d_1(\cdot, v)\, \partial_k v)(1 + d_2(\cdot, v)\, \partial_k v) + v\, \partial_k^2 v, \end{aligned}$$

where $d_{1,2}(\tau, k, v) := -k/v \pm v/2$. The measured surface is feasible where both quantities are nonnegative.

Local volatility Where $\mathcal{C}_{\text{str}}[f]$ is positive, Dupire's equation gives the squared local volatility

$$a^2(\tau, k) = \frac{\mathcal{C}_{\text{cal}}[f](\tau, k)}{\mathcal{C}_{\text{str}}[f](\tau, k)}.$$

Violations can therefore produce negative or undefined local variance, while a small positive denominator can make the recovered local volatility ill-conditioned. Our reported feasibility results concern the two indicators on bounded dense grids and are not a global certificate.

G.2 Data

Source and processing We use OptionMetrics snapshots for SPX, NDX, RUT, and DJX, accessed through WRDS, over 2020–2024. For each index, we retain every tenth available snapshot among

the first 1,250, giving 125 dates per index and 500 observed market surfaces. We augment each surface by reflecting log-moneyness, and assign the reflected copy its own posterior, giving 1,000 fitted codebook entries per run. We retain AM-settled quotes with positive, non-crossed bid and ask prices and compute option mid prices. For each expiry, the discount factor is estimated from box spreads and the forward from put–call parity. We retain out-of-the-money puts for $k \leq 0$ and calls for $k > 0$, and invert bid, mid, and ask prices with the Let's-Be-Rational algorithm [Jäckel, 2015], as implemented in *py-vollib-vectorized*.[4]

Expiry and constraint domains Training uses listed expiries in $[0.01, 1]$ years and retains the closest enclosing expiry on either side when available. No log-moneyness filter is applied to the market observations. At each optimisation step, the two constraints are evaluated at every retained expiry on 128 newly sampled log-moneyness values from $[-1.05, 1.05]$. The model supports at most 20 listed expiries per surface.

G.3 Variational autodecoder

We use a weakly variational variant of the autodecoder [Park et al., 2019, Zadeh et al., 2021]. Each fitted surface has a learned Gaussian posterior over a latent path, with one eight-dimensional code per listed expiry and no encoder. The posterior has an upper-bidiagonal precision factor, while its prior is a Matérn-1/2 Gaussian process in $\sqrt{\tau}$ with learned variance and length scale. A six-layer, four-head causal transformer uses 32 Fourier expiry features to map the latent path to one set of FiLM parameters per expiry. The shared slice decoder has three 32-unit hidden layers, with a $\tanh$ first layer followed by two sinusoidal layers, and takes $k/\sqrt{\tau}$ as input; an exponential output activation ensures positive implied volatility. The FiLM modulator has one 128-unit hidden layer and a learnable scaled-sigmoid activation for its multiplicative coefficients. For off-expiry queries, we decode the two adjacent slices and interpolate total variance linearly, with $w(0, k) = 0$.

G.4 Training

For a snapshot $\mathcal{O}$ and a latent path $z = (z_1, \ldots, z_I)$, write $\hat{y}_{i,j}(z) := f^\theta(\tau_i, k_{i,j}; z)$ for the decoded implied volatilities. The reconstruction loss of the snapshot combines the squared relative mid error with a relative hinge outside the bid–ask interval, averaged first over the contracts of an expiry and then over expiries:

$$\ell(\theta, z) = \frac{1}{I} \sum_{i=1}^{I} \frac{1}{J_i} \sum_{j=1}^{J_i} \left[\Big(\frac{\hat{y}_{i,j}(z) - y^{\text{mid}}_{i,j}}{y^{\text{mid}}_{i,j}} \Big)^2 + \frac{[\hat{y}_{i,j}(z) - y^{\text{ask}}_{i,j}]_+ + [y^{\text{bid}}_{i,j} - \hat{y}_{i,j}(z)]_+}{y^{\text{mid}}_{i,j}} \right]. \tag{16}$$

For the training snapshots $\mathcal{D} = \{\mathcal{O}_n\}_{n=1}^N$, snapshot n with I_n expiries, reconstruction loss ℓ_n and posterior q_n, the data loss is $\mathcal{L}_{\text{data}} := \frac{1}{N} \sum_{n=1}^N \mathbb{E}_{z \sim q_n}[\ell_n(\theta, z)]$, estimated with one reparametrised draw per snapshot and step. The training objective is $\mathcal{L}_{\text{data}} + \mathcal{L}_{\text{var}}$ plus the constraint terms of each method, with the expiry-count-normalised KL divergence to the Matérn-1/2 prior p,

$$\mathcal{L}_{\text{var}} = \frac{\beta}{N} \sum_{n=1}^{N} \frac{1}{I_n} \mathrm{KL}\big(q_n \,\|\, p\big), \qquad \beta = 10^{-3}.$$

We train for 1,000 epochs with Adam, batch size 64, learning rate 3×10^{-4} for the transformer, decoder, latent posteriors, and GP hyperparameters, and gradient clipping at norm 10.

G.5 Comparison methods

The penalty method uses the linear hinge penalty (2) with $p = 1$ for each constraint. The neural augmented Lagrangian uses the objective (9) with a multiplier decoder, updated jointly with the model (joint) or in dual phases of one or five epochs after every primal epoch (PDL-1, PDL-5), as described in Section B.2. The neural slack variable uses a positive two-channel slice decoder and the slack loss with the transform T of Section A.1 at $\sigma = 0.01$. All methods use the margin $\varepsilon = 10^{-3}$ for both constraints. The constraint weight ρ starts at 1 for the penalty and augmented Lagrangian methods and at 100 for the neural slack variable, and is multiplied by 1.01 after every epoch with a

[4] `https://github.com/marcdemers/py_vollib_vectorized`

measured violation on the training points, up to 100 and 10^4 respectively; it is held fixed after epoch 900. The slack and multiplier decoders have the architecture of the volatility decoder, conditioned on the same slice parameters and trained at the same learning rate, but with 64 instead of 32 units per hidden layer, which improved the data fit.

Neural EMA adapts the exponential moving average target $s_t = (1-\alpha)\, s_{t-1} + \alpha\, \Pi_K(c_{\theta_t})$, with $\alpha = 1 - 2^{-1/h}$ for a half-life h, to constraint points that are sampled anew at every step. Since no average can then be stored per point, a positive network s^ϕ with the architecture of the neural slack variable represents the target on the whole constraint domain. At the start of every epoch, with θ fixed, one pass over the training batches regresses s^ϕ by squared error onto $(1-\alpha)\, \bar{s} + \alpha\, \Pi_K(c_\theta)$, where $\bar{s}$ is a frozen copy of s^ϕ taken before the pass, so that h is measured in epochs. The primary network is then trained for one epoch on the data loss and on $\mathcal{L}_{\text{slack}}$ with s^ϕ held fixed, using the same transform T, margin ε and constraint weight schedule as the neural slack variable. This phased schedule follows the multiplier regression of PDL in Section B.2, with the projected multiplier target replaced by the EMA target. Neural EMA thus retains the positive neural slack and adds explicit temporal smoothing to its fit, whereas the neural slack variable minimises $\mathcal{L}_{\text{slack}}$ jointly in (θ, ϕ) without a separate target or half-life. We include it to test whether this smoothing improves on joint fitting, with $h \in \{0, 0.5, 1, 2, 4\}$. For $h = 0$ ($\alpha = 1$), the pass regresses s^ϕ onto the exact slack $\Pi_K(c_\theta)$ without smoothing, so that neural EMA implements a nested scheme for (5) with one regression pass per epoch.

G.6 Evaluation

We evaluate each method in three regimes, FITTED, POSTERIOR and HOLDOUT. In each regime, the constraints are measured at every listed expiry on 1,000 equally spaced log-moneyness values in $[-1, 1]$. For each constraint, Table 10 reports the number N_{viol} of surfaces with a measured violation and the statistics η_{rate}, η_{mean} and η_{max} defined in Section C, computed over these evaluation points for all surfaces in the regime.

FITTED For each of the three independent runs of a method, we evaluate all 1,000 codebook entries at their posterior means, giving 3,000 surface evaluations in total. We report the implied volatility MAE δ_{mae} as mean ± standard deviation over the three runs, and training time as the minimum over the runs.

POSTERIOR In each run, we sample 1,000 fitted posteriors with replacement and draw one latent path from each, again giving 3,000 surface evaluations in total.

HOLDOUT For each method, we freeze its seed-0 decoder and calibrate a new latent path to each of 660 unseen surfaces from 2025, with 165 from each index. This follows the usual autodecoder inference procedure of optimising the latent representation while keeping the shared decoder fixed [Park et al., 2019]. We parametrise the latent path of a held-out snapshot in coordinates whitened by the prior, $z_d = L w_d$ for each latent channel d, where $LL^\top = K$ is the Cholesky factorisation of the prior covariance on the snapshot's expiries. Then $-\log p(z) = \frac{1}{2}\|w\|^2$ up to a constant, and the calibration objective is

$$\mathcal{J}(w) = \ell(\theta, Lw) + \frac{\beta_{\text{cal}}}{I}\, \tfrac{1}{2}\|w\|^2.$$

With $\beta_{\text{cal}} = \beta$, the prior has the weight it has in training, and minimising $\mathcal{J}$ gives the maximum a posteriori path under the trained model; we use $\beta_{\text{cal}} = \beta/10$ for every method. For the line search, the hinge in (16) is smoothed by a softplus with inverse temperature 1,000. We minimise $\mathcal{J}$ from $w = 0$, the prior mean, by 30 L-BFGS steps in double precision. Neither constraint appears in the calibration objective, and both are measured on the same evaluation grid after every step. A final surface is counted as feasible only when neither constraint has a measured violation.

Table 10: Results in the three evaluation regimes. In HOLDOUT, Loss ℓ is the final reconstruction loss (16) averaged over the held-out surfaces, and Steps the mean number of the 30 calibration steps after which a constraint is violated. Training time in hours as the minimum over the three runs; Apple M3 Pro (MPS), total training time ~ 61 h. All tested neural augmented Lagrangian variants retain measured violations, consistent with the interpretation that restricted multiplier classes enforce representable aggregates of the pointwise constraints [Narasimhan et al., 2020]. Neural EMA shares the neural slack variable's spatial coupling and obtains comparable measured feasibility in FITTED and POSTERIOR, but higher δ_{mae} at every tested half-life h. These results are consistent with joint fitting obtaining the benefit of spatial coupling without explicit temporal smoothing or a separate fitting phase.

		$\mathcal{C}_{\text{cal}}$				$\mathcal{C}_{\text{str}}$					
FITTED		$N_{\text{viol}}/N_{\text{tot}}$	η_{rate}	η_{mean}	η_{max}	$N_{\text{viol}}/N_{\text{tot}}$	η_{rate}	η_{mean}	η_{max}	δ_{mae}	Time (h)
Neural slack		0/3,000	0	—	—	0/3,000	0	—	—	0.00514 ± 0.00054	1.4
Penalty		8/3,000	7.4e−06	9.9e−04	2.8e−03	30/3,000	4.3e−06	4.0e−03	1.3e−02	0.00466 ± 0.00054	1.2
Neural ALM	joint	60/3,000	1.3e−04	3.2e−03	1.5e−02	606/3,000	3.3e−04	9.3e−03	9.1e−02	0.00424 ± 0.00017	1.4
	PDL-1	103/3,000	2.5e−04	3.3e−03	1.5e−02	641/3,000	3.8e−04	8.8e−03	8.3e−02	0.00424 ± 0.00009	1.7
	PDL-5	89/3,000	2.4e−04	3.5e−03	2.5e−02	421/3,000	2.0e−04	7.7e−03	7.6e−02	0.00433 ± 0.00048	3.5
Neural EMA	$h = 0$	0/3,000	0	—	—	0/3,000	0	—	—	0.00568 ± 0.00023	2.7
	$h = 0.5$	0/3,000	0	—	—	0/3,000	0	—	—	0.00557 ± 0.00014	1.7
	$h = 1$	0/3,000	0	—	—	0/3,000	0	—	—	0.00582 ± 0.00035	1.7
	$h = 2$	0/3,000	0	—	—	0/3,000	0	—	—	0.00570 ± 0.00022	1.7
	$h = 4$	0/3,000	0	—	—	0/3,000	0	—	—	0.00609 ± 0.00034	1.7

		$\mathcal{C}_{\text{cal}}$				$\mathcal{C}_{\text{str}}$			
POSTERIOR		$N_{\text{viol}}/N_{\text{tot}}$	η_{rate}	η_{mean}	η_{max}	$N_{\text{viol}}/N_{\text{tot}}$	η_{rate}	η_{mean}	η_{max}
Neural slack		0/3,000	0	—	—	0/3,000	0	—	—
Penalty		30/3,000	7.0e−05	2.9e−03	1.3e−02	41/3,000	7.2e−06	3.5e−03	1.4e−02
Neural ALM	joint	221/3,000	8.6e−04	5.8e−03	5.6e−02	674/3,000	5.0e−04	9.0e−03	1.1e−01
	PDL-1	240/3,000	1.0e−03	7.2e−03	6.2e−02	703/3,000	5.1e−04	8.5e−03	1.2e−01
	PDL-5	217/3,000	9.4e−04	5.1e−03	4.9e−02	436/3,000	3.1e−04	8.3e−03	7.9e−02
Neural EMA	$h = 0$	0/3,000	0	—	—	0/3,000	0	—	—
	$h = 0.5$	0/3,000	0	—	—	0/3,000	0	—	—
	$h = 1$	0/3,000	0	—	—	0/3,000	0	—	—
	$h = 2$	0/3,000	0	—	—	0/3,000	0	—	—
	$h = 4$	2/3,000	8.6e−07	7.3e−04	1.5e−03	0/3,000	0	—	—

		$\mathcal{C}_{\text{cal}}$				$\mathcal{C}_{\text{str}}$					Steps	
HOLDOUT		$N_{\text{viol}}/N_{\text{tot}}$	η_{rate}	η_{mean}	η_{max}	$N_{\text{viol}}/N_{\text{tot}}$	η_{rate}	η_{mean}	η_{max}	Loss ℓ	$\mathcal{C}_{\text{cal}}$	$\mathcal{C}_{\text{str}}$
Neural slack		9/660	3.3e−04	3.5e−02	9.4e−02	2/660	5.6e−06	1.0e−02	1.6e−02	0.00453	0.41	0.09
Penalty		36/660	8.2e−04	7.2e−03	7.5e−02	49/660	5.4e−04	7.9e−03	1.1e−01	0.00382	1.60	2.11
Neural ALM	joint	41/660	9.6e−04	9.8e−03	6.9e−02	264/660	3.5e−03	2.0e−02	3.7e−01	0.00385	1.85	11.59
	PDL-1	80/660	2.0e−03	1.4e−02	1.6e−01	285/660	3.8e−03	1.2e−02	2.0e−01	0.00392	3.64	12.73
	PDL-5	25/660	5.4e−04	1.8e−02	1.1e−01	111/660	8.7e−04	1.2e−02	2.5e−01	0.00397	1.15	5.00
Neural EMA	$h = 0$	10/660	3.5e−04	5.5e−02	1.5e−01	2/660	4.4e−06	3.9e−02	9.4e−02	0.00540	0.44	0.08
	$h = 0.5$	7/660	2.3e−04	3.5e−02	9.5e−02	5/660	2.1e−05	5.3e−02	1.3e−01	0.00518	0.32	0.21
	$h = 1$	8/660	2.7e−04	2.1e−02	6.2e−02	3/660	1.0e−04	3.9e−02	1.4e−01	0.00598	0.36	0.14
	$h = 2$	7/660	2.3e−04	3.5e−02	1.2e−01	0/660	0	—	—	0.00589	0.32	0.00
	$h = 4$	7/660	2.6e−04	3.3e−02	1.2e−01	0/660	0	—	—	0.00688	0.33	0.00

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract scopes its empirical claims to low-dimensional domains and to feasibility measured on dense evaluation grids, and the limitations in Section 5 state the tuning cost and the lack of a feasibility guarantee specific to neural slack variables. The method contribution in Section 1 is specified in Section 3, and each empirical contribution maps to an experiment section, including the drift ablation (Section 4.2) and the main application (Section 4.3).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 contains a dedicated Limitations paragraph that discusses the scope of applicability and the computational costs relative to alternative methods.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The off-grid bounds in Section C.2 state their assumptions (Lipschitz continuity and the fill distance of the evaluation grid) and are derived in full there. The projection decomposition in Section 3 follows in one line from $\mathcal{S} \subset K$, and the gradient-flow analysis in Section A.3 is a heuristic, not a formal result.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The neural slack variable algorithm is specified in Section 3 and Algorithm 1. We report architectures, optimiser and schedule, training and evaluation grids, epochs, and seed counts in Section 4; synthetic-data generators and full hyperparameters are documented in the appendix. Code is released (see Q5).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code is available at `https://github.com/rwicl/neural-slack-variables-for-shape-constraints`, with Hydra configs and entry points reproducing the monotonicity, convexity, and FOSSIL Barr3 experiments end-to-end (synthetic-data generators included). The volatility-surface experiment (Section 4.3) cannot be reproduced from public assets because it relies on SPX, NDX, RUT and DJX option quotes from OptionMetrics, accessed under a WRDS licence that prohibits redistribution; the code includes the full preprocessing pipeline and documents how to obtain the raw feed.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 (together with the referenced appendix sections and the released code) reports all training and evaluation details necessary to interpret the results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report δ_{mae} as mean $\pm$ standard deviation over independent training seeds, constraint statistics as median [min, max] over the infeasible seeds, and feasibility as $N_{\text{sat}}/N_{\text{tot}}$ (Section 4, Section C); the variability factor is the seed, which sets the random initialisation and the noise and grid draws and is paired across methods.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: All reported experiments ran on a single Apple M3 Pro laptop. Hardware, per-run training time and total training time are reported in the captions of Table 8 (MPS, about 17 h), Table 9 (CPU, about 3.1 h) and Table 10 (MPS, about 61 h, excluding calibration). The monotone runs (Tables 1 and 7 and the sweeps of Section D.3) take under a minute each on the CPU, about 16 h in total. The reported runs thus sum to roughly 100 h of training time; the project additionally required preliminary and ablation runs not reported in the paper.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conforms with the NeurIPS Code of Ethics: no human subjects, no personally identifiable data, and no foreseeable harmful applications; the only real-world data used (SPX, NDX, RUT and DJX option quotes via OptionMetrics) is anonymous aggregate market data accessed under institutional licence.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work proposes a generic soft-constraint training scheme for shape-constrained regression; the application studies (no-arbitrage volatility surfaces, neural barrier certificates) target settings where stricter feasibility is itself the beneficial property, and we identify no direct path from the method to negative societal applications.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We release training code for a generic soft-constraint method on small-scale function approximation and a per-day implied volatility autodecoder; neither asset constitutes a generative model or scraped dataset with foreseeable misuse potential, so no specific safeguards are warranted.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Third-party code assets are credited in the appendix with their original paper and source URL: the FOSSIL Barr3 benchmark (BSD 3-Clause), the Let's-Be-Rational algorithm [Jäckel, 2015] in its `py-vollib-vectorized` implementation (MIT), and the CMNN implementation of its authors, which states no licence and which we reimplement rather than redistribute; the OptionMetrics option data are accessed under an institutional WRDS licence, which we observe by not redistributing the raw data (see Q5).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The repository at `https://github.com/rwicl/neural-slack-variables-for-shape-constraints` documents installation (`uv sync`), entry-point usage per experiment, and the Hydra config layout in its top-level `README.md`.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing and no research with human subjects; data are synthetic (monotone and convex targets, FOSSIL Barr3 dynamical system) or anonymous market quotes (SPX, NDX, RUT and DJX options via OptionMetrics).

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper involves no human-subjects research, so no IRB approval or equivalent review applies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not part of the core methodology; the proposed neural slack variable framework operates on small networks for shape-constrained regression, with no language-model component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.