Spectral and Conformal: Neural Network Methods for Calibrated Trajectory Prediction and Control-Point Intervals

Anonymous authors
Paper under double-blind review

Abstract

Autonomous systems that navigate their environment need methods to quickly assess the feasibility and safety of desired trajectories. Traditional solutions that rely on numerical integration can be computationally heavy, while fast data-driven surrogates lack the formal guarantees that numerical approaches provide. This work demonstrates a framework that uses (1) a neural network with a Koopman-like spectral backbone to learn trajectories and quantile bands with built-in constraints (initial conditions and stability); (2) a modified conformal quantile regression to provide calibrated *joint*, trajectory-level conformal coverage; and (3) interval arithmetic to extend coverage guarantees to control-points. For a quadrotor under a closed-loop linear quadratic regulator, we show that by refactoring the network for inference, the trajectory inference time is 3.5 ms on a Raspberry Pi Zero 2 W, within a total pipeline time of 9.81 ms, enabling operation at approximately 100 Hz, within the range of typical control rates. The framework shows that learned surrogates can compactly express meaningful statistical guarantees at control-relevant speeds on constrained hardware.

1 Introduction

Autonomous systems that navigate their environment need methods to quickly assess the feasibility and safety of desired trajectories. This challenge arises in a broad range of applications where safety constraints must be satisfied despite nonlinear dynamics, disturbances, and modeling uncertainty (for example autonomous vehicles, aerial robots, industrial automation, and medical systems). Several approaches have been developed to address this problem, including reachability analysis (Bansal et al., 2017), invariant-set methods (Blanchini & Miani, 2015), control barrier functions (Ames et al., 2019), reference governors (Garone et al., 2017), and predictive control techniques (Rawlings et al., 2017). While these methods provide different forms of safety guarantees, they often rely on accurate models of the system dynamics and rely on numerical integration that is computationally demanding, especially for highly nonlinear systems (Wabersich et al., 2023).

A common remedy is to replace expensive numerical integration with surrogate models. Data-driven surrogate models can capture unmodeled dynamics, adapt to changing operating conditions, and provide accurate approximations of system behavior over finite horizons. However, their use in safety-critical applications remains challenging; surrogates lack the guarantees of full numerical integration and can carry unquantified, unassessed uncertainties (Hsu et al., 2023). Furthermore, detailed trajectory representations are often not in a form suitable for safety verification or online decision making — for instance, checking constraint satisfaction along a predicted trajectory typically requires sampling or numerically integrating the predicted dynamics with small timesteps.

This work describes a framework that combines uncertainty-aware trajectory learning with a spline-based representation to provide prediction with coverage guarantees in a compact representation of dynamical-system trajectories. The spline representation is particularly useful because B-spline trajectories are convex combinations of their control points. Therefore, safety constraints can be evaluated directly on uncertain control-point sets. Using the surrogate model, such estimation can occur without repeated numerical integration of the underlying dynamics. We apply a modified conformal quantile regression (CQR)(Romano et al., 2019) to provide per-dimension, per-timestep, as well as joint (total trajectory) prediction coverage. This

step is critical to downstream estimates of stability and safety. Our benchmarking studies on a Raspberry Pi Zero 2 W show that real-time safety assessment is feasible with this method.

Our method does not explicitly encode ODE dynamics and could work for any similar stable autonomous ODE. Showing that spline control points can inherit the uncertainty ranges of predicted trajectories supports a variety of control applications from model predictive control (MPC) to safety filters. The primary contributions of this work are as follows:

- A demonstration of a single-forward-pass mapping from an initial condition to an adaptive-knot B-spline trajectory approximation for an autonomous ODE, without numerical integration.
- An architecture that combines a Koopman-inspired spectral decomposition layer with a triplet of decoders to provide (1) point prediction corrections to the spectral backbone and (2) quantile estimates when used with a pinball loss and ramped scaling.
- A method for producing calibrated, verification-ready uncertainty: conformal quantile regression (CQR) yields tight *whole-trajectory* (joint) coverage guarantees, and the resulting quantile bands map to B-spline control point ranges via interval arithmetic.
- Inference benchmarks on constrained hardware showing latency within a 100 Hz control-rate budget enabled by our method, almost 5× faster than a `Numba`-compiled RK4 integrator.

2 Preliminaries

Notation. Throughout, $f(t)$ denotes the true trajectory of the system: the solution of the underlying ODE from an initial condition $x_0 \in \mathbb{R}^{12}$ (12 is the number of dimensions of the LQR equations), sampled at m rollout points. The network maps x_0 to a predicted trajectory $x(t)$, together with lower and upper quantile estimates $x_{\text{lo}}(t)$ and $x_{\text{hi}}(t)$. By construction, $x(t = 0) = x_0$. Note that t is actually a sampled set $t \in \mathcal{T} = \{0, ..., t_m\}$ of the m rollout points.

A set of B-spline basis functions $B \in \mathbb{R}^{m \times n_{cp}}$ is constructed for a given set of knot locations $\{t_j\}$, where n_{cp} denotes the number of control points: for a clamped knot vector with n_{int} interior knots and degree k, $n_{cp} = n_{\text{int}} + k + 1$ (for cubic splines, $n_{\text{int}} + 4$). Throughout, $i \in \{1, \ldots, n_{cp}\}$ indexes basis functions and control points, and $j \in \{1, \ldots, n_{\text{int}}\}$ indexes interior knots. A B-spline of degree k with basis functions $B_i^k(t)$, determined by a knot vector, represents a trajectory through control points: $s(t) = B(t)\, c$ denotes the spline reconstruction of the true trajectory, with $c = B^\dagger f$, while $\hat{s}(t) = B(t)\, \hat{c}$, with $\hat{c} = B^\dagger x$, denotes the spline reconstruction of the *predicted* trajectory. The pipeline outputs $\hat{s}(t)$; $s(t)$ serves as its idealized counterpart.

Treatment and Framing. We treat the trajectory $f(t)$ as any continuous function that can be approximated by a spline. To characterize $f(t)$, it is sampled at m rollout points. Prior work (Authors, 2026) has shown that knot point placement that contributes minimal error while remaining sparse can be achieved by using the relationship between knot density, local curvature, and maximum error (De Boor, 1978). Adaptive knot placement supports the approach described in this paper because it avoids unnecessary dense spline knots that lead to inflated $B^\dagger$ pseudoinverse calculation burden. Assuming the knot vector is found by this or other methods, the spline approximation of the trajectory reduces to finding a fast approximation of the trajectory itself ($x(t)$). Therefore, instead of we desire a spline approximation $s(t)$. The Moore-Penrose pseudoinverse $B^\dagger = (B^T B)^{-1} B^T$ allows one to compute the control points that reconstruct the miniumum least-squares trajectory approximation via $\hat{c} = B^\dagger x$ but requires an estimate of the full trajectory, even though the output spline is a condensed representation. Thus, the primary effort is in estimating the full trajectory, with the control point calculation being a fast, post-prediction step.

The Koopman operator framing recasts a nonlinear dynamical system in terms of a (potentially infinite-dimensional) lifted space of observable functions that evolve *linearly* under the dynamics; finite-dimensional approximations of this lifted evolution are commonly learned via autoencoder architectures (Lusch et al., 2018; Pan & Duraisamy, 2020; Otto & Rowley, 2021; Brunton et al., 2022).

The methods discussed here are tailored for the broad set of autonomous ordinary differential equations (ODEs) that describe trajectories uniquely determined from an initial condition. Ideally, the system is stable or the time horizon short enough that it does not exhibit chaotic or highly nonlinear behavior. For this, we focus on the linear quadratic regulator (LQR), described in Appendix A.

3 Related Work

Learning linear operators such as Koopman or Lie operators through neural networks is well studied (Lusch et al., 2018; Brunton et al., 2016; 2022; Otto & Rowley, 2019; 2021). Specifically, the architecture employed in this paper is similar to architectures in several recent papers including most relevantly Pan & Duraisamy (2020) and Jamil & Kapadia (2026). In particular, similar to previous papers such as Otto & Rowley (2019); Pan & Duraisamy (2020); Jamil & Kapadia (2026), we use a pipeline that includes an encoder that lifts the state dimensions to a higher dimensional space followed by an evolution operator built on spectral decomposition, and ending with a decoder. We differ from Jamil & Kapadia (2026) in that our architecture does not include an inverse fast Fourier transform (IFFT) and lacks the spatial distribution inherent in their partial differential equation (PDE) setting. This is largely an artifact of the problem domains these papers focus on; an IFFT does not benefit this problem and is costly to implement.

We enforce stability by directly capping eigenvalues rather than through negative-semi-definite parameterization such as by Pan & Duraisamy (2020) or skew-symmetric decomposition such as in Jamil & Kapadia (2026). Furthermore, we quantify uncertainty via conformal calibration rather than Bayesian variational inference. Conformal calibration yields distribution-free, finite-sample coverage guarantees without distributional assumptions on the model, whereas a Bayesian posterior provides a full parametric uncertainty model but relies on the correctness of prior assumptions for calibration. Otto & Rowley (2019) differs significantly because theirs is a discrete time system whereas ours is treated as continuous.

Our pinball loss is naturally suited to conformal calibration. Exchangeability, which is required for conformal prediction, is easily satisfied because training, calibration, and test data are drawn as independent and identically distributed (IID) samples from the same input distribution. Applied per-dimension and per-timestep, conformalized quantile regression (CQR) (Romano et al., 2019) yields our cell- and step-level guarantees directly. At the trajectory level, the useful guarantee is joint: coverage holding simultaneously across all time steps and dimensions. Joint coverage can be obtained via Bonferroni-style corrections (Stankeviciute et al., 2021) (overly conservative) or copula-based methods (Sun & Yu, 2022) (requires dependence estimation); we instead follow the supremum-based construction of Diquigiovanni et al. (2022), adapting it to CQR.

In order to express the predicted quantile range using splines, we leverage interval arithmetic to directly map the bounds through the pseudoinverse. This treatment is a parallel, but distinct method from common forward treatments such as Van Loock et al. (2015) and Rousseau et al. (2019) that use point-wise constraints derived from system dynamics to build constraints on the control point values.

4 Method

The composite network comprises two pieces:

- A knot-placement estimator (20K parameters) that estimates the curvature over time, enabling a justified interior knot placement. This is a simple fully connected neural network (FCNN) discussed in Authors (2026) and separate from the work described here.

- A trajectory estimator (75K parameters) that estimates the trajectory from the d-dimensional initial condition x_0^d. This has three sub-components: A FCNN nonlinear encoder ("lift" operation); a spectral layer similar to Lusch et al. (2018); and a triplet of decoders. This is discussed below and shown in Figure 1.

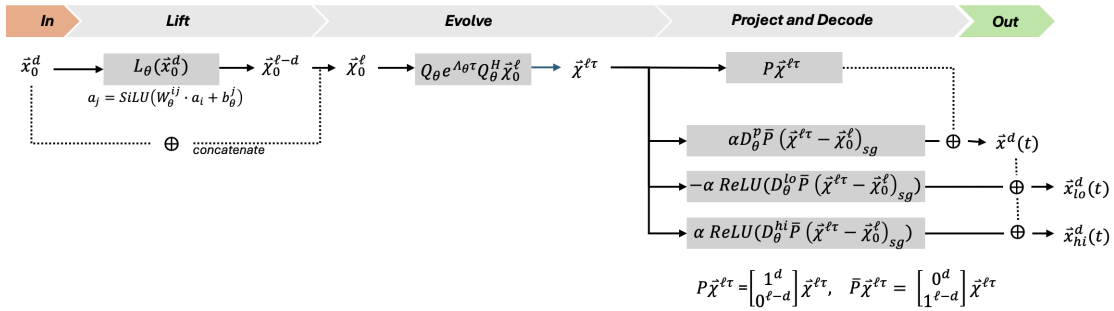

Figure 1: A schematic of the Trajectory network showing the lift, evolve, and project and decode portions of the network in sequence. The initial condition $\vec{x}_0^d$ is concatenated with the output of a SiLU encoder L_θ to form the lifted state $\vec{\chi}_0^\ell$, which the spectral layer $Q_\theta e^{\Lambda_\theta \tau} Q_\theta^H$ evolves linearly to any query time τ. The projection $P\vec{\chi}^{\ell\tau}$ plus a mean-correction head yields $\vec{x}^{\,d}(t)$; two ReLU-gated quantile heads, added around the mean, yield $\vec{x}_{\mathrm{lo}}^d(t)$ and $\vec{x}_{\mathrm{hi}}^d(t)$. All heads act on the stop-gradient (sg) residual $\bar{P}(\vec{\chi}^{\ell\tau} - \vec{\chi}_0^\ell)$, with gate α enabled after warmup; at τ=0 the output equals x_0 by construction

4.1 Trajectory Prediction

The model maps an initial condition $x_0 \in \mathbb{R}^d$ to a predicted trajectory through three stages: a lift into a higher-dimensional observable space, a stable continuous-time spectral evolution, and an anchored quantile decoder. Hereafter, $x(t) \equiv x_{\mathrm{mean}}^d(t)$ denotes the point prediction.

Lift. The initial condition is lifted by an MLP L_θ and concatenated with the state itself,

$$\chi_0^\ell = x_0^d \oplus L_\theta(x_0^d), \qquad a_j = \mathrm{SiLU}\big(W_\theta^{ij} a_i + b_\theta^j\big), \tag{1}$$

where $\oplus$ denotes concatenation, d is the state dimension, and ℓ the lifted dimension.

Evolve. The lifted state is propagated to time τ in closed form through the stable spectral generator,

$$\chi^{\ell\tau} = Q_\theta\, e^{\Lambda_\theta \tau}\, Q_\theta^H\, \chi_0^\ell, \tag{2}$$

with $Q_\theta = \exp(S)$, $S = A - A^H$ unitary and $\mathrm{Re}\,\Lambda_\theta \le -\epsilon_{\min} < 0$ enforcing stability.

Project and Decode. Two projection operators, P and $\bar{P}$, project the output of the spectral generator into the initial state channels and non-state channels respectively. The decoders act only on the *non-state* channels of the evolved lifted state, detached from the computational graph. If $\bar{P}$ projects onto the non-state channels and sg$[\cdot]$ denotes stop-gradient:

$$\tilde{h}(\tau) = \mathrm{sg}\big[\,\bar{P}\,\chi^{\ell\tau}\,\big]. \tag{3}$$

The mean and quantile bands anchor each decoder's output to zero at $\tau = 0$:

$$x_{\mathrm{mean}}^d(\tau) = P\,\chi^{\ell\tau} + \alpha\big(D_\theta^{\mathrm{mean}}(\tilde{h}(\tau)) - D_\theta^{\mathrm{mean}}(\tilde{h}(0))\big), \tag{4}$$

$$x_{\mathrm{lo}}^d(\tau) = x_{\mathrm{mean}}^d(\tau) - \alpha\,\mathrm{ReLU}\big(D_\theta^{\mathrm{lo}}(\tilde{h}(\tau)) - D_\theta^{\mathrm{lo}}(\tilde{h}(0))\big), \tag{5}$$

$$x_{\mathrm{hi}}^d(\tau) = x_{\mathrm{mean}}^d(\tau) + \alpha\,\mathrm{ReLU}\big(D_\theta^{\mathrm{hi}}(\tilde{h}(\tau)) - D_\theta^{\mathrm{hi}}(\tilde{h}(0))\big). \tag{6}$$

The stop-gradient ensures the lift and spectral parameters are trained only through the projected state $P\chi^{\ell\tau}$, while the detached decoders D_θ learn correction terms atop the frozen lifted representation. This architecture propagates lifted-state perturbations to state-space prediction errors through a single linear operator,

$$M(\tau) = P\,Q_\theta\, e^{\Lambda_\theta \tau}\, Q_\theta^H + \alpha\, D_\theta \bar{P}\big(Q_\theta\, e^{\Lambda_\theta \tau}\, Q_\theta^H - I\big)\,. \tag{7}$$

At $\tau = 0$, $Q_\theta\, e^{\Lambda_\theta \cdot 0}\, Q_\theta^H = I$, so $M(0) = P$: the prediction error and band width are zero at the initial condition by construction. Importantly, for $\tau > 0$, $M(\tau)$ decays with the (strictly negative) real parts of the spectral eigenvalues Λ_θ. We employ a constraint on the real parts of $M(\tau)$ to ensure that this holds. A diagram of the network is shown in Figure 1.

5 Experiments

5.1 Baseline Trajectory Prediction

For our experiments in trajectory prediction, we used the same system as described in Authors (2026). For each initial condition, a trajectory is generated via numerical integration of the nonlinear dynamics equations. For subsequent spline approximations, we use cubic B-splines. The physical constants we use in the quadrotor solver are given in Table 3. Inputs are normalized by a radius R, which is set by the outer perimeter of the ball of initial conditions, but more generally is arbitrary.

The baseline model is described in Table 4 in Appendix B. The learnable parameters are approximately evenly distributed between the Lift and Evolve steps with the three Decoder layers parameters accounting for about 12% of the total learnable parameters. Empirically we found that the values used here balanced network size and performance. More details are available in Appendix C.

Loss. The training objective combines a mean trajectory loss, two pinball (quantile) losses, and an optional integral-consistency penalty. For the sampled true trajectory f and pinball loss $\rho_q(u) = \max(qu, (q-1)u)$ at quantile level q,

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{traj}} + \mathcal{L}_{\text{traj}}^{\text{norm}}}_{\text{mean fit}} + \underbrace{\rho_{q_{\text{lo}}}\big(f - x_{\text{lo}}^d\big) + \rho_{q_{\text{hi}}}\big(f - x_{\text{hi}}^d\big)}_{\text{quantile bands}} + \lambda_{\text{int}}\, \alpha\, \mathcal{L}_{\text{int}} \tag{8}$$

where the mean terms are the mean-squared error and its per-channel standard-deviation–normalized counterpart,

$$\mathcal{L}_{\text{traj}} = \big\| x_{\text{mean}}^d - f \big\|_2^2, \qquad \mathcal{L}_{\text{traj}}^{\text{norm}} = \Big\| \big(x_{\text{mean}}^d - f\big) \oslash \sigma \Big\|_2^2, \tag{9}$$

with σ the per-channel standard deviation and $\oslash$ element-wise division. The integral-consistency term enforces the kinematic relation that position increments match the trapezoidal integral of velocity (in physical units), and is active only when $\lambda_{\text{int}} > 0$. Notably, because x_{lo}^d and x_{hi}^d include α-dependent adjustments to the mean value, they are fixed to x_{mean}^d until α turns on.

Data generation. Data was generated using initial conditions randomly selected from within the ball of allowable initial conditions given in Table 3 (weighted towards initial conditions with larger radius). We numerically integrated using a fourth-order Runge–Kutta (RK4) scheme, JIT-compiled with `Numba`.

Training procedure. The gate α follows a linear warm-up schedule, held at 0 until epoch $e_{\text{start}} = 400$, ramped linearly to 1 by epoch $e_{\text{end}} = 500$, and fixed at 1 thereafter, so the decoder correction is introduced only after the spectral backbone has settled. The spectral parameters use a learning-rate multiplier of 3 relative to the rest of the network, and the learning rate begins at $lr = 0.001$. Optimization used Adam with `ReduceLROnPlateau` on the validation trajectory loss; a fixed epoch budget ($E_{stop} = 900$) is used with no early stopping, and the best checkpoint is selected post hoc. The batch size was 2500 samples with 100 batches per epoch. New trajectories were generated every 5 epochs, such that over 900 epochs the effective training dataset was 45M trajectories.

Results The trajectory loss $\mathcal{L}_{\text{traj}}$ and the total loss of Equation 8 is shown for the baseline model in Figure 4 in Appendix B. This shows that the spectral component learns the majority of the trajectory fit prior to the α ramp beginning. Ramping α enables the total loss to drop (through learning meaningful quantiles) and, after experiencing some transient instability, the loss converges steadily. The trajectory loss is the mean squared error (MSE). The validation final mean squared error averaged $(1.11 \pm 0.13) \times 10^{-6}$ over different runs with different initial seeds. Notably, the spectral backbone alone (prior to α ramp-up) is able to achieve an MSE less than $2\times$ the final MSE.

5.2 Conformal Quantile Regression

We use *Conformal Quantile Regression* (CQR) as proposed by Romano et al. (2019) to evaluate and scale the quantiles predicted by our model to guarantee coverage. The general CQR procedure is summarized by three steps:

1 Generate data split into a calibration and hold-out dataset and predict the quantile bounds ((L_{low}, L_{high}), in our case using $(x^d_{\text{lo}}, x^d_{\text{hi}})$) using the calibration set.

2 Compute the score $\eta = \max(L_{low} - f, f - L_{high})$, which quantifies whether each sample is within the quantile band ($\eta < 0$) or not ($\eta > 0$)

3 Create a new calibrated band $(L_{low} - Q_\eta, L_{high} + Q_\eta)$ where Q_η is the $(1-\delta)(1+1/n)$ quantile of η (n is the number of calibration samples and in practice $1/n \ll 1$). Romano et al. (2019) emphasizes that this works for symmetric quantiles, which are the only ones that we consider as well.

Both the raw model outputs and the result of the standard CQR procedure are concerned with marginal coverage. However, control problems necessitate total trajectory guarantees, which requiress *joint coverage*. Thus, we modify this procedure in two ways closely following the supremum-based construction by Diquigiovanni et al. (2022). First, we generate a separate set of predictions to measure the per-dimension accuracy by calculating the absolute error distribution (L_1 distance from the *point prediction*) for each dimension independently and reporting its 95% quantile. We chose the 95% quantile to generally match the quantile of interest for the overall procedure and because we want the scaling to be sensitive to the tail of the distribution. The score in step (2) is computed on the trajectories scaled by $1/(\vec{R} \odot \vec{\omega})$ where $\vec{\omega}$ is the L_1 scaling. To recover the correct units, the conformal factor Q_η in step (3) is multiplied by $\vec{R} \odot \vec{\omega}$. A primary motivation for this is that our model predicts in a normalized coordinate system where each dimension is scaled by the range of initial conditions such that the 12-D ball of initial conditions has unit radius. Physically, this is arbitrary. Without a rescaling prior to the CQR procedure, the resulting scale is influenced by the dimension that has the largest variance under the arbitrary scaling. In addition, the $\vec{\omega}$ prevents the conformal analysis from over-correcting dimensions that rarely contribute to the score in step (2).

Second, we modify the score in step (2) to be the *maximum* value for any dimension or time step. Because the score now equals the per-trajectory worst-case cell, the conformal guarantee in step (3) applies jointly across the trajectory. In our analysis we used 100,000 samples total. 40,000 samples were used to predict the ω scaling and 40,000 samples were used to predict Q_η. A final 20,000 samples were saved as a hold-out sample for evaluation.

We compare two models that are trained to predict the 95% CI (2.5%, 97.5%) and 99% CI (0.5%, 99.5%) quantiles. The CQR procedure was performed to best match the original predicted quantiles (95% for the 95% CI, 99% for the 99% CI). As shown in Table 1, models are well calibrated at the per-dimension, per-time prediction level. However, the 95% CI quantile model fails to capture most *full trajectories* at the 95% level with a little more than 1% of trajectories within the bounds. The same is true, but less dramatically, for the 99% CI quantile model, which has 23% of trajectories within the 99% CI bounds. This justifies conformal scaling. Table 1 demonstrates that the ω scaling is supportive of tighter bounds in the translational position and velocity dimensions. The results suggest that predicting a higher quantile may improve accuracy and reduce the need for post-train CQR adjustments. However, such quantiles may be difficult to train in practice due to the few probable samples in the tail of the distribution.

In Figure 2 the hold-out sample that performed worst prior to CQR adjustments is shown (first six translational dimensions - (u, x, v, y, w, z)) with the CQR adjustments applied to the quantile band for the 95% CI baseline trained model. All dimensions and an example at the 99% CI are provided in Appendix D. The tightness of the interval and the value of the Q_η and $\vec{R} \odot \vec{\omega}$ weights are clear with the true trajectory falling within the 95% interval.

Table 1: Conformal quantile regression on the baseline trajectory model (ℓ=128, m=251, seed 42), trained at two coverage targets. Raw cell-level coverage is close to the trained nominal in both cases, but raw trajectory-level (joint) coverage is far below the target. After conformal correction, the trajectory-level coverage reaches the target, with mean band width roughly 2× the raw band width. Per-dimension weights $\omega[d]$ reflect difficulty in R-normalized units: body rates (p, q) and angles (ϕ, θ) are hardest, while translational positions (x, y) are easiest.

Target	**Raw coverage**		**Post CQR coverage**		**Raw width**	**Post CQR width**	Q_η
	cell	joint	cell	joint	(mean)	(mean)	
95%	94.94%	1.10%	99.95%	95.31%	0.00327	0.00775	0.00093
99%	99.05%	22.64%	99.99%	99.00%	0.00433	0.00855	0.00087

Target	u	x	v	y	w	z	p	ϕ	q	θ	r	ψ
95%	0.874	0.244	0.877	0.231	1.396	0.398	2.819	2.771	2.694	2.763	0.453	1.584
99%	0.883	0.248	0.901	0.240	1.423	0.415	2.840	2.841	2.703	2.830	0.384	1.509

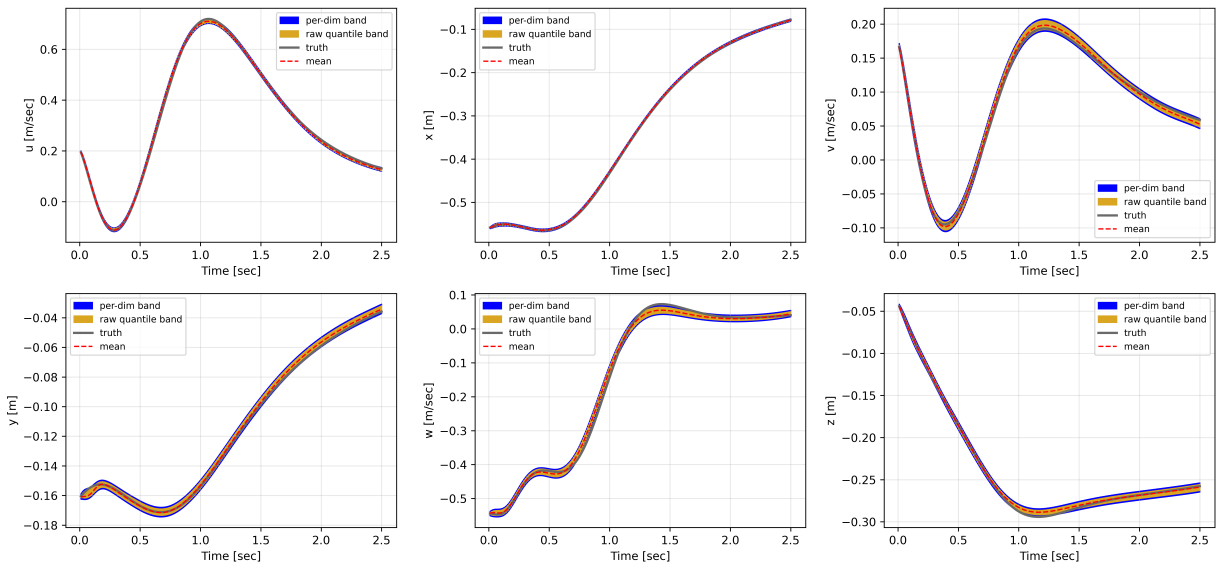

Figure 2: Conformal-calibrated trajectory bands for the emphworst-case hold-out trajectory under the 95% quantile model (baseline: ℓ=128, m=251, seed 42) prior to conformal calibration. Solid gray is the true trajectory; red dashed is the model's mean prediction; gold is the raw quantile band before calibration; blue is the per-dim conformal-calibrated band at 95% joint coverage. The "worst case" is the test trajectory whose raw band has the largest miss before calibration; the calibrated band still contains the true trajectory at every (t, d).

5.3 Ablation Studies

We performed ablation studies, focusing on the effect of rollout size and latent lift dimensions on the fit through the total and validation trajectory loss. We did not ablate the encoder MLP architecture; the network's ability to express dynamics lives within the spectral layer. The full results are in Appendix C. We find that the lift dimension is by far the most influential parameter.

5.4 Initial Condition Uncertainty Propagation

The conformal analysis performed above relies on the exchangeability of the data. We implicitly assumed that the inference distribution is the same as the training distribution. This assumption is certainly violated

for real systems which always have sensor-induced noise. Tibshirani et al. (2019) provides the requisite analysis to enable one to extend the method above under covariate shift between training and inference due to noise (see also Joshi et al. (2026) for a treatment that includes both CQR and covariate shift). However, such analysis relies on noise-induced distribution shifts that must be justified either through empirical data or a verified noise model — which we currently do not have. Appendix E shows how initial-condition noise yields probabilistic bands without coverage guarantees (e.g.,when only the current point's uncertainty is known).

5.5 Inference Speed on Constrained Hardware

We evaluated inference speeds on a Raspberry Pi Zero 2 W with a quad-core 64-bit ARM Cortex-A53 processor and 512 MB of SDRAM in order to mimic deployment on constrained edge hardware. We used ONNX version 1.12.0 (opset 17) for inference implementation. Because ONNX is unable to support complex numbers, the conversion to ONNX format necessitated the expansion of the spectral decomposition into real and imaginary components. We leveraged the linearity between the spectral layer and the projection and decoder operations by noting that by using $c_j = \sum_{\ell'} Q_\theta^H{}_{j,\ell'} \vec{\chi}_0^\ell$, we can rewrite

$$\left(\Gamma^T Q_\theta \, e^{\Lambda_\theta \tau} \, Q_\theta^H \, \vec{\chi}_0\right)_d = \sum_\ell \Gamma_{\ell d} \sum_j Q_\theta^{\ell j} \, e^{\lambda_j \tau} \underbrace{\sum_{\ell'} Q_\theta^H{}_{j\ell'} \, \vec{\chi}_0^{\ell'}}_{c_j} = \sum_j \underbrace{\left[\sum_\ell \Gamma_{\ell d} \, Q_\theta^{\ell j}\right]}_{\Psi_{dj}} \, c_j \, e^{\lambda_j \tau} \tag{10}$$

Where $\Gamma_{\ell d}$ represents any of the linear projection and decoder operation (there are four versions of Γ, one for each of: P, $D_\theta^{mean}\bar{P}$, $D_\theta^{lo}\bar{P}$, and $D_\theta^{hi}\bar{P}$. Refer to Equations 4, 5, and 6). Both $\Psi_{dj} = \sum_\ell \Gamma_{\ell d} Q_\theta^{\ell j}$ and $e^{\lambda_j \tau}$ can be precomputed. Because $\Psi_{dj} c_j$ is smaller than $c_j e^{\lambda_j \tau}$ ($n_{dim} \times \ell_{\text{lift}}$ versus $m \times \ell_{\text{lift}}$ where $n_{dim} \sim 12$ and $m \sim 251$ and ℓ_{lift} is the lift dimension), computing Ψ_{dj} first in the right-most expression of Equation 10 reduces memory use on the memory constrained Raspberry Pi. We found this reordering to be critical in improving the inference speed, providing a roughly 10× reduction in inference time when applied in addition to the precomputed matrices.

We ran 11,000 initial conditions through a pipeline that included both the activity model used to identify knot locations, spline basis construction, trajectory prediction, and finally control point calculation via Python Scipy's optimized `make_lsq_spline` function. The data was processed as sequential samples (unbatched). The first 1000 samples were used as warm-up and discarded. In Table 2, we show the mean, median, and 95th percentile run times for each step and those values summed to an estimated total time. We find that the total pipeline inference time 9.81 ms per inference (95% below 10 ms) with 3.45 ms on average coming from the trajectory prediction. This is comparable with controller speeds ($\sim 100\,\text{Hz}$) for drones (Mandaokar & Rinner, 2026). For comparison, we substituted the trajectory model (3(A) in Table 2) with the `Numba` compiled RK4 integration scheme that is used to generate training data on the Raspberry Pi (3(B) in Table 2). We found that the average time for numerical integration alone was 16.10 ms (95% below 21.65 ms), increasing the average total time to 22.46 ms. This is above what is common for controller speeds and suggests that an ML-supported trajectory prediction enables safety filters that work at regimes where numerical integration is not a viable option.

6 Computing Control Points with Coverage Guarantees

Thus far we have demonstrated a strong, coverage-guaranteed estimate of the trajectory. We return to the problem of providing a compact representation of the trajectory through a spline representation. As discussed previously, the point estimate spline representation can be found simply via $c^d = B^\dagger \cdot x^d_{\text{mean}}$. However, it is unclear how the coverage estimated by the quantiles can be incorporated into the spline representation. The naive approach of approximating the upper and lower quantile bands using splines provides a spline-encoded description of the bands, but is absent of information regarding the valid range for each control point in the set. We suggest the following method, which finds the range of control point values admissible by the set of functions within the quantile range.

Table 2: Per-stage inference latency on the Raspberry Pi Zero 2 W. The focus of this paper is the trajectory prediction component. 3(A) and 3(B) represent alternative ways to calculate the trajectory — through our method (3(A)) or traditional numerical integration (3(B)). The Activity Model refers to a method applied in Authors (2026).

Stage	Description	Mean (ms)	Median (ms)	95th pct (ms)
1	Activity Model (identifies knot locations)	2.07	2.04	2.12
2	Spline Basis Construction	1.48	1.46	1.55
3(A)	**Trajectory Prediction (ML - This Paper)**	**3.45**	**3.41**	**3.49**
3(B)	*Numerical Integration (alternate)*	*16.10*	*16.04*	*21.65*
4	Control Point Calculation	2.81	2.77	2.97
	Total with ML (1+2+3A+4)	**9.81**	**9.69**	**10.00**
	Total with integration (1+2+3B+4)	*22.46*	*22.31*	*28.29*

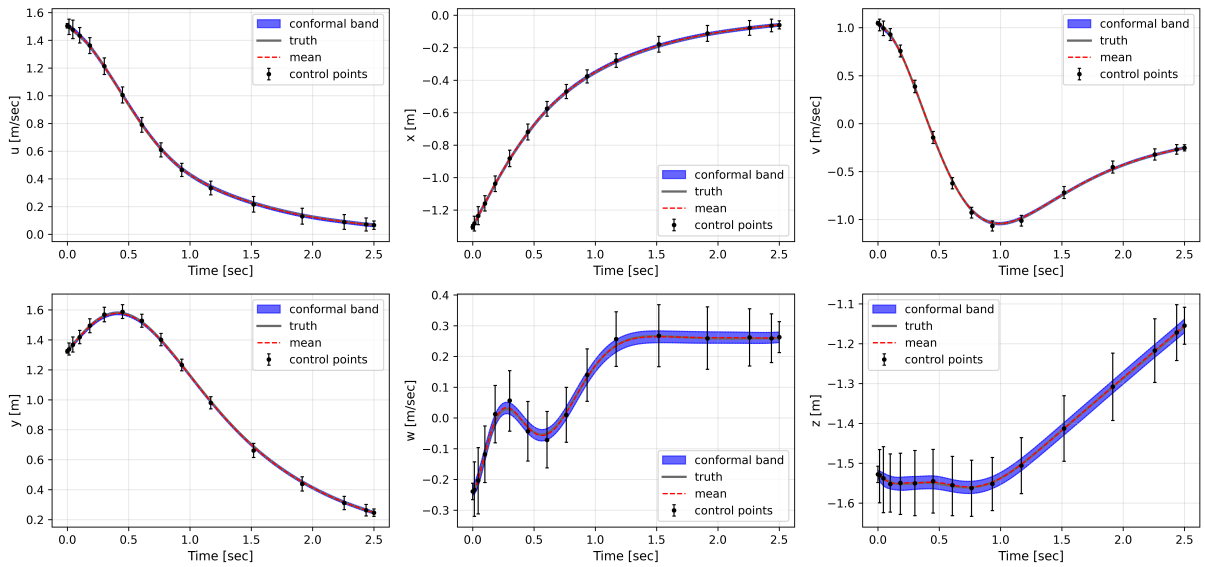

Figure 3: Calibrated trajectory bands and control-point intervals for one test trajectory, across six of twelve state dimensions (u, x, v, y, w, z) (ℓ=128, m=251, 95% joint coverage). Gray solid is the true trajectory; red dashed is the mean prediction; the blue band is the per-dim conformal-calibrated band; black dots with vertical bars show the derived control points ($c^d_{i,\text{mid}}$) and their intervals $[c^d_{i,\text{lo}}, c^d_{i,\text{hi}}]$ at the Greville abscissae of the knot vector. The control-point intervals contain the spline coefficients of any band-respecting trajectory.

The conformal calibration procedure described in the previous section produces, for each state component d, a prediction interval with joint coverage at least $1-\delta$; that is, the true trajectory f^d satisfies $x^d_{\text{lo}}(t) \leq f^d(t) \leq x^d_{\text{hi}}(t)$ for all $t \in \mathcal{T}$ with probability at least $1-\delta$. Let

$$\mathcal{F}^d_{in} = \left\{ g^d : x^d_{\text{lo}}(t) \leq g^d(t) \leq x^d_{\text{hi}}(t), \quad \forall t \in \mathcal{T} \right\} \tag{11}$$

denote the set of admissible trajectories for component d. Given the knot vector $\{t_j\}$, let B denote the corresponding B-spline collocation matrix evaluated at the sampling instants. For any $g^d \in \mathcal{F}^d_{in}$, the associated control points are $c^d = B^\dagger g^d$, where $B^\dagger$ denotes the Moore–Penrose pseudoinverse, so the i-th control point is

$$c^d_i(g^d) = \sum_{t \in \mathcal{T}} B^\dagger_{it}\, g^d(t). \tag{12}$$

Our goal is to determine an interval containing all admissible values of c^d_i induced by the trajectory uncertainty band. To this end, we define $c^d_{i,\text{hi}}$ and $c^d_{i,\text{lo}}$ as follows

$$c^d_{i,\mathrm{hi}} = \sup_{g^d \in \mathcal{F}^d_{in}} c^d_i(g^d), \qquad c^d_{i,\mathrm{lo}} = \inf_{g^d \in \mathcal{F}^d_{in}} c^d_i(g^d). \tag{13}$$

Since c^d_i is linear in g^d, the supremum and infimum over $\mathcal{F}^d_{in}$ decouple across sampling instants. The maximizing trajectory selects $x^d_{\mathrm{hi}}(t)$ where $B^\dagger_{it} > 0$ and $x^d_{\mathrm{lo}}(t)$ where $B^\dagger_{it} < 0$, and symmetrically for the minimizer. Writing $(B^\dagger_{it})^\pm = \max(\pm B^\dagger_{it}, 0)$, the extremal control-point values are

$$c^d_{i,\mathrm{hi}} = \sum_{t \in \mathcal{T}} (B^\dagger_{it})^+ \, x^d_{\mathrm{hi}}(t) - (B^\dagger_{it})^- \, x^d_{\mathrm{lo}}(t), \tag{14}$$

$$c^d_{i,\mathrm{lo}} = \sum_{t \in \mathcal{T}} (B^\dagger_{it})^+ \, x^d_{\mathrm{lo}}(t) - (B^\dagger_{it})^- \, x^d_{\mathrm{hi}}(t). \tag{15}$$

Every admissible trajectory $g^d \in \mathcal{F}^d_{in}$ therefore generates control points satisfying $c^d_{i,\mathrm{lo}} \le c^d_i \le c^d_{i,\mathrm{hi}}$. These bounds admit a compact matrix formulation. Define the midpoint (mid) and half-width (hw) for both the prediction interval and control points:

$$x^d_{\mathrm{mid}}(t) = \frac{x^d_{\mathrm{hi}}(t) + x^d_{\mathrm{lo}}(t)}{2}, \quad x^d_{\mathrm{hw}}(t) = \frac{x^d_{\mathrm{hi}}(t) - x^d_{\mathrm{lo}}(t)}{2} \quad c^d_{i,\mathrm{mid}} = \frac{c^d_{i,\mathrm{hi}} + c^d_{i,\mathrm{lo}}}{2}, \quad c^d_{i,\mathrm{hw}} = \frac{c^d_{i,\mathrm{hi}} - c^d_{i,\mathrm{lo}}}{2}. \tag{16}$$

Denoting by $|B^\dagger|$ the matrix with entries $|B^\dagger_{it}|$, we observe that

$$|B^\dagger_{it}| = (B^\dagger_{it})^+ + (B^\dagger_{it})^-. \tag{17}$$

$$B^\dagger_{it} = (B^\dagger_{it})^+ - (B^\dagger_{it})^-. \tag{18}$$

Adding equations equation 14 and equation 15 and dividing by two yields the midpoint; subtracting and dividing by two yields the half-width:

$$c^d_{\mathrm{mid}} = B^\dagger \, x^d_{\mathrm{mid}}, \qquad c^d_{\mathrm{hw}} = |B^\dagger| \, x^d_{\mathrm{hw}}. \tag{19}$$

Equation equation 19 is the standard midpoint–radius form of interval arithmetic for linear maps (see, e.g., Moore et al., 2009; Neumaier, 1990), applied here to the spline pseudoinverse. The interval midpoints are the least-squares projection of the band midpoint, exactly as in the point-estimate case, while the interval widths are linear in the band half-widths through the entrywise non-negative matrix $|B^\dagger|$. Zero trajectory uncertainty yields zero control-point uncertainty. The structure of $|B^\dagger|$ — in particular, which of its entries are large — determines which control points are most sensitive to trajectory uncertainty.

Because $\mathcal{F}_{in}$ contains the true trajectory with probability $\ge 1 - \delta$ by the conformal guarantee, the resulting control-point intervals contain the control points of the least-squares spline fit to the true trajectory with at least that probability. Notably, the true trajectory and its spline fit can differ; for $f \in C^2$, this residual is bounded by a Jackson-type inequality (DeVore, 1998; Authors, 2026),

$$\|f - B(t)\, B^\dagger f\|_\infty \;\le\; C_d \max_j \, h_j^2 \, \|\ddot{f}\|_{\infty,[t_j,\, t_{j+1}]} \;=:\; \epsilon_{\mathrm{knot}}, \tag{20}$$

where h_j is the local knot spacing and C_d depends only on the spline degree. The least-squares fit attains the best-approximation rate up to the stability constant of the projection, which is bounded independently of the knot vector (Shadrin, 2001). With dense sampling of the trajectory (satisfied in this work with m=251 samples against a typical $n_{cp} < 25$), the discrete projector inherits this behavior. Prior work (Authors, 2026) enables one to choose knot locations to explicitly minimize ϵ_{knot}.

Setting ϵ_{knot} aside momentarily, because B-spline curves lie within the convex hull of their control points, any convex constraint satisfied at every point of the control-point intervals is satisfied by the corresponding spline functions. The control point intervals contain the *spline fit of* the true trajectory and thus it satisfies the convex constraint with probability at least $1 - \delta$. This is a conservative estimate for two reasons. First, the mapping from $\mathcal{F}_{in}^d$ to control-point intervals is not invertible: the resulting set of control-point intervals contains $B^\dagger \mathcal{F}_{in}^d$ as a strict subset and thus admits control point vectors that generate functions which lie outside the quantile bands. Second, the convex hull represents a loose outer containment that is dependent on the number of control points (De Boor, 1978). By tightening the convex constraint set by ϵ_{knot} from Equation equation 20, we can extend the guarantee to the true trajectory itself: a reference declared admissible is then safe with probability at least $1 - \delta$.

The time to compute these bounds on the control points is negligible compared to the trajectory prediction and control point calculation. An example trajectory with computed control points and their intervals (shown as error bars) is shown in Figure 3. The linear relationship between control point intervals and the width of the conformal band is visible across the 6 translational dimensions (see Appendix D for the full 12 dimensions). The overall tightness of the control point intervals empirically demonstrates the value of this approach.

Imposing smoothness constraints on the admissible set is a promising direction to further tighten the bounds on control points. Restricting $\mathcal{F}_{in}^d$ to functions with bounded increments, $|g^d(t_{a+1}) - g^d(t_a)| \leq L^d \, \Delta t_a$, excludes non-physical extremizers; the constrained extremization is no longer decoupled over sampling instants. Notably, the rate bound L^d need not be assumed: evaluating the system dynamics over the extreme values of the conformal band yields a certified bound on $|\dot{f}^d|$, so the true trajectory remains in the restricted set whenever it lies in the band, and the $1-\delta$ guarantee transfers to the tighter intervals unchanged. Leveraging additional features as in Rousseau et al. (2019) and Van Loock et al. (2015) may also reduce the intervals.

7 Discussion

By combining Koopman-inspired trajectory prediction, conformal uncertainty quantification, adaptive knot placement, and control-point derivation, the proposed architecture produces both a trajectory prediction and an associated finite-dimensional uncertainty representation. For the LQR problem, our neural network's spectral layer and triplet of decoders demonstrated both accurate point prediction and accurate marginal distribution estimates across the trajectory.

The spectral layer enforces system stability by constraining $\operatorname{Re} \Lambda_\theta \leq -\epsilon_{\min} < 0$ and serves as a backbone on top of which each decoder head is responsible only for nonlinear corrections. Evidence validating the Koopman framing is embedded in the validation loss during training (Figure 4). Prior to the α ramp-up, the model is purely the spectral backbone. However, after the α ramp-up, excepting some training instability related to the warm-up period, the course of trajectory loss improvement is unchanged. The decoders provide *corrections*, but most of the dynamics can be estimated using a linear evolution on the lifted embedding.

The uncertainty associated with the predicted trajectory can be transferred to the spline control points. As a result, the predicted system behavior can be represented through a finite collection of control-point intervals rather than through a dense set of trajectory samples. This provides a compact uncertainty-aware description of the trajectory that can be manipulated directly and may be more suitable for downstream decision-making algorithms than pointwise trajectory representations.

A key motivator was the application of this representation in safety assessment for autonomous systems. As shown in Section 6, convex constraints can be evaluated directly on the control-point intervals — conservatively, and without numerical integration. This supports an admissibility test by confidence interval that can be used to enable probabilistic safety filters that determine whether a desired reference is admissible. Extending such filters to *repair* an inadmissible reference, that is, computing a nearby admissible one satisfying state and terminal constraints, is left to future work.

More broadly, the proposed framework establishes a bridge between uncertainty-aware learning and classical spline-based representations commonly used in planning and control. While this work focuses on trajectory prediction and uncertainty quantification, the resulting control-point representation may support

future developments in safety verification, reference governors, reachability analysis, trajectory planning, and learning-enabled autonomy architectures requiring fast evaluation of uncertain trajectories.

7.1 Future Work

Several directions that merit future study are identified below.

Other Dynamical Systems This paper focuses on the linear quadratic regulator as a model system, but does not explicitly leverage its dynamical properties except for stability. A natural next step is to demonstrate how well these same methods generalize to other autonomous ODEs such as the Duffing oscillator and ODEs of increasing complexity. Relaxing constraints on stability and fixed points and testing on systems such as the Lorenz attractor will stress-test our assumptions and the usefulness of the coverage bands.

Conformal Prediction under Covariate Shift The initial-condition noise propagation of Section 5.4 provides probabilistic bands but no coverage guarantee. Extending the conformal procedure to sensor-noise covariate shift via the weighting of Tibshirani et al. (2019) (see also Joshi et al., 2026) could restore guarantees, at the cost of requiring an empirically justified or verified noise model.

Safety Filters and Reference Repair The admissibility test of Section 6 determines whether a desired reference is safe. Extending it to *repair* an inadmissible reference requires search over candidate references and a formalization of the constraint sets. This is one of the most direct paths from this representation to a deployed safety filter.

Smoothness-Informed Control Points Section 6 suggests a path to tighten control-point intervals while preserving coverage by restricting the admissible set to those functions that are dynamically valid. Implementing and benchmarking the resulting linear programs is a natural next step.

8 Conclusion

In this work we have demonstrated an end-to-end neural-network-based method to map initial conditions to control-point intervals that contain the least-squares fit of the true trajectory for an autonomous ODE (in this case a quadrotor under closed-loop LQR) with at least $(1 - \delta)$ coverage under the standard exchangeability assumption of conformal prediction. We leveraged and modified several established procedures to accomplish this. First, we trained a neural network with a spectral layer to provide the backbone expected-value point prediction. We add three decoder heads to improve the point estimate and generate quantile estimates while retaining the dynamical insights of the learned spectral layer (noise and ablation studies are also provided). After training, we used a modified conformal quantile regression from Romano et al. (2019) with a supremum across dimensions and timesteps, closely following Diquigiovanni et al. (2022) to provide joint conformal coverage. To our knowledge the application of these techniques together is new. We benchmark the performance on a Raspberry Pi Zero 2 W and show that after refactoring for inference the trajectory model is almost $5\times$ faster than our original numerical integrator on constrained hardware and can support control processes.

Most importantly, our extensions to the spline assessment demonstrate that this can provide control points with coverage intervals that guarantee the location of the least-squares spline approximation to the true trajectory with probability at least $(1 - \delta)$. The residual between this least-squares approximation and the true trajectory is minimized by the knot point placement algorithm. Notably, the core components — the spectral decomposition, activity-based knot placement, spline pseudoinverse, and conformal calibration — are not specific to the quadrotor and should extend to other autonomous ODE systems near a stable fixed point.

Acknowledgements

Acknowledgements held for blind review.

References

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pp. 3420–3431. Ieee, 2019.

Current Authors. A neural network approach towards real-time knot point density estimates for the nonlinear quadrotor. In Review, 2026.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J. Tomlin. Hamilton-Jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2242–2253. IEEE, 2017.

Franco Blanchini and Stefano Miani. *Set-Theoretic Methods in Control*. Systems & Control: Foundations & Applications. Birkhäuser, Basel, 2 edition, 2015.

Steven L Brunton, Bingni W Brunton, Joshua L Proctor, and J Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PloS one*, 11(2):e0150171, 2016.

Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern koopman theory for dynamical systems. *SIAM Review*, 2022.

Carl De Boor. *A practical guide to splines*, volume 27. springer New York, 1978.

Ronald A DeVore. Nonlinear approximation. *Acta numerica*, 7:51–150, 1998.

Jacopo Diquigiovanni, Matteo Fontana, and Simone Vantini. Conformal prediction bands for multivariate functional data. *Journal of Multivariate Analysis*, 189:104879, 2022.

Emanuele Garone, Stefano Di Cairano, and Ilya Kolmanovsky. Reference and command governors for systems with constraints: A survey on theory and applications. *Automatica*, 75:306–328, 2017.

Kai-Chieh Hsu, Haimin Hu, and Jaime F Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

Shafayeth Jamil and Rehan Kapadia. Lie generator networks for nonlinear partial differential equations. *arXiv preprint arXiv:2603.29264*, 2026.

Sunay Joshi, Shayan Kiyani, George J Pappas, Edgar Dobriban, and Hamed Hassani. Conformal inference under high-dimensional covariate shifts via likelihood-ratio regularization. *Advances in Neural Information Processing Systems*, 38:16079–16120, 2026.

Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, 2018.

Dnyandeep Mandaokar and Bernhard Rinner. Distributionally robust acceleration control barrier filter for efficient uav obstacle avoidance. *IEEE Robotics and Automation Letters*, 11(5):5914–5921, 2026.

Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. SIAM, Philadelphia, 2009.

Arnold Neumaier. *Interval Methods for Systems of Equations*, volume 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1990.

Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

Samuel E. Otto and Clarence W. Rowley. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:59–87, 2021. doi: 10.1146/annurev-control-071020-010108.

Shaowu Pan and Karthik Duraisamy. Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability. *SIAM Journal on Applied Dynamical Systems*, 19(1):480–509, 2020.

James B. Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, Madison, WI, 2 edition, 2017.

Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. *Advances in neural information processing systems*, 32, 2019.

Gauthier Rousseau, Cristina Stoica Maniu, Sihem Tebbani, Mathieu Babel, and Nicolas Martin. Minimum-time b-spline trajectories with corridor constraints. application to cinematographic quadrotor flight plans. *Control Engineering Practice*, 89:190–203, 2019.

Alexei Yu. Shadrin. The L_∞-norm of the L_2-spline projector is bounded independently of the knot sequence: A proof of de Boor's conjecture. *Acta Mathematica*, 187(1):59–137, 2001.

Kamile Stankeviciute, Ahmed M Alaa, and Mihaela Van der Schaar. Conformal time-series forecasting. *Advances in neural information processing systems*, 34:6216–6228, 2021.

Sophia Sun and Rose Yu. Copula conformal prediction for multi-step time series forecasting. *arXiv preprint arXiv:2212.03281*, 2022.

Ryan J Tibshirani, Rina Foygel Barber, Emmanuel Candes, and Aaditya Ramdas. Conformal prediction under covariate shift. *Advances in neural information processing systems*, 32, 2019.

Wannes Van Loock, Goele Pipeleers, and Jan Swevers. B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction. *Mechanical Sciences*, 6(2):163–171, 2015.

Kim P Wabersich, Andrew J Taylor, Jason J Choi, Koushil Sreenath, Claire J Tomlin, Aaron D Ames, and Melanie N Zeilinger. Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Magazine*, 43(5):137–177, 2023.

A Quadrotor Dynamics and Data-Generation Parameters

We consider a six-degree-of-freedom rigid-body model of a quadrotor. The system can be described with the following equations:

$$\dot{u} = rv - qw - g\sin\theta \tag{21}$$

$$\dot{v} = pw - ru + g\cos\theta\sin\phi \tag{22}$$

$$\dot{w} = qu - pv + g\cos\theta\cos\phi - F_1/m_{kg} \tag{23}$$

$$\dot{p} = \frac{(J_y - J_z)}{J_x}qr + F_2/J_x \tag{24}$$

$$\dot{q} = \frac{(J_z - J_x)}{J_y}pr + F_3/J_y \tag{25}$$

$$\dot{r} = \frac{(J_x - J_y)}{J_z}pq + F_4/J_z \tag{26}$$

where (u, v, w) are the translational velocities; (p, q, r) the rotational velocities; m_{kg} is the mass; (J_x, J_y, J_z) are the principal moments of inertia; g, the acceleration due to gravity; F_1 the total thrust force generated by the rotors; (F_2, F_3, F_4), the rolling, pitching, and yawing moment thrusts. These are combined with the six integral equations, where we use a small-angle approximation for the kinematic mapping and the closed loop law:

$$\mathbf{F} = \begin{bmatrix} F_f \\ 0 \\ 0 \\ 0 \end{bmatrix} - K(\mathbf{x} - \mathbf{x}_{sp}) \tag{27}$$

where F_f is the trim thrust, K the LQR gain matrix, and $\mathbf{x}_{sp}$ the setpoint state. The full values for the quadrotor system are given below in Table 3.

Parameter	**Value**
Quadrotor Dynamics	
Total system mass	0.600 kg
Quadrotor body mass	0.410 kg
Propeller radius	0.0504 meters
Arm length	0.108 meters
Propeller mass	0.00311 kg
Motor + propeller mass	0.0391 kg
Initial Conditions Range (maximum magnitudes)	
$x^{ic}_{\max}$	5.0 meters
$\dot{x}^{ic}_{\max}$	3.0 meters / second
$\theta^{ic}_{\max}$	$\pi/4$ radians
$\dot{\theta}^{ic}_{\max}$	$\pi/2$ radians / second
Discretization and Processing	
t_{samp}	0.005 seconds
T (total time)	2.5 seconds

Table 3: Parameter values used in data generation and processing. Table repeated from Authors (2026).

B Architecture and Training Details

Here we present the details of the neural network architecture and an example of the training loss. Table 4 shows the sizes of each layer in the network. The spectral decomposition output is complex-valued and is equivalent to twos 128-sized real-valued tensors, thus the shape (2, 128). The loss equations for the total loss and trajectory loss shown in Figure 4 are given by Equation 8 and Equation 9 respectively. Excepting some training instability related to the warm-up period, the course of trajectory loss improvement is unchanged after α ramp-up. The total loss decreases precipitously because the upper and lower bounds are no longer fixed to the expected value.

Table 4: The baseline architecture used to predict the trajectory and quantile bands from initial conditions. The spectral decomposition evolves an ℓ=128 lifted state; decoders read the $2\ell - d = 244$ non-state lifted channels and are bias-free.

Layer Type	**Input**	**Output**	**Param.**
Linear / SiLU	12	128	1664
Linear / SiLU	128	64	8256
Linear / SiLU	64	128	8320
Linear (lift)	128	116	14964
Spectral decomposition	128	$(2, 128)$	33024
Decoder: mean (Linear)	244	12	2928
Decoder: low (Linear)	244	12	2928
Decoder: high (Linear)	244	12	2928
Total			**75012**

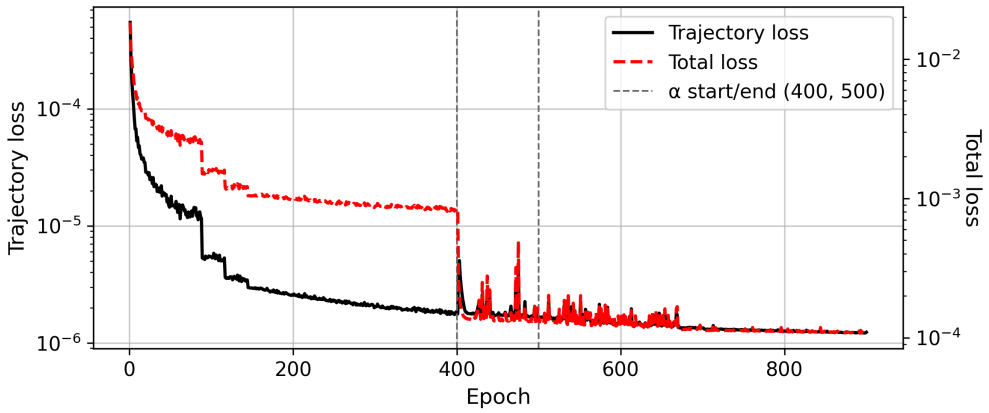

Figure 4: Validation loss curves for the baseline model (ℓ=128, m=251): trajectory loss (mean MSE, black, left axis) and total loss including pinball band terms (red dotted, right axis). Vertical lines mark the α warmup window (epochs 400-500), during which band decoders activate and the loss briefly shows some instability before stabilizing and converging slowly. There are 62,500 samples in the validation dataset. These samples are generated once (unlike training data that is regenerated every 5 epochs) such that model performance is evaluated across the same data each epoch.

C Ablation Studies

Here we present results of an ablation study of our baseline network, varying the most impactful components of the *lift* and *evolve* framework. We find that the lift dimension is by far the most influential parameter. For a lift dimension of 64, the trajectory loss increased by about 50% from 1.12×10^{-6} to 1.69×10^{-6}. This, despite increasing the stopping epoch from $E_{stop} = 900$ to $E_{stop} = 1200$ because the models with lower lifted dimension are slower to converge after α is ramped to 1.0. Increasing lifted dimension from 64 to 128 reduced trajectory loss by $\sim 34\%$; further increasing from 128 to 256 reduced it by only $\sim 22\%$ despite adding roughly $4\times$ more spectral parameters ($\mathcal{O}(\ell^2)$ scaling). This suggests diminishing returns and motivates $\ell = 128$ as a practical operating point. Integral consistency loss was largely ineffective, likely because such consistency was learned by the system and did not contribute significantly to the loss. Using a 99% instead of 95% quantile reduced the trajectory loss almost negligibly. The total loss for the 99% quantile model is significantly lower, however it is not directly comparable to the 95% quantile bin loss. Rollout dimension had little effect on the overall loss.

Architecture		Validation Loss	
Lifted Dim	Rollout Steps	Trajectory Only (mean MSE, $\times 10^{-6}$)	Total (total, $\times 10^{-4}$)
Baseline (lifted = 128, rollout = 251, 3 seeds)			
128	251	1.12(13)	1.050(94)
Lifted dimension sweep (rollout = 251, 3 seeds)			
64 (1200 epochs)	251	1.69(19)	1.144(64)
256	251	0.875(53)	0.841(22)
Rollout sweep (lifted = 128, 3 seeds)			
128	101	1.03(11)	0.971(55)
128	501	1.090(59)	1.000(28)
Single Runs Tests (lifted = 128, rollout = 251, seed 42)			
Integral Consistency ($\lambda_{\text{int}} = 0.25$)		1.24	1.09
99% Quantile interval†		0.985	0.303

Table 5: Effect of lifted dimension, rollout length, physics consistency, and trained quantile interval on fit. Architecture rows report mean ± std over three seeds; ablation rows are single-seed. Trajectory Only is the MSE on the mean prediction (interval-invariant); Total is the total validation loss including pinball band terms. *Trained for 1200 epochs rather than 900. †The validation loss at the 99% interval is not directly comparable to the 95% rows: the pinball weights q, $1-q$ are smaller at more extreme quantile levels.

D Full Twelve-Dimension Results

The following figures present examples of full results - across all 12 dimensions for the model through its various stages of post-training conformal quantile regression and spline fitting.

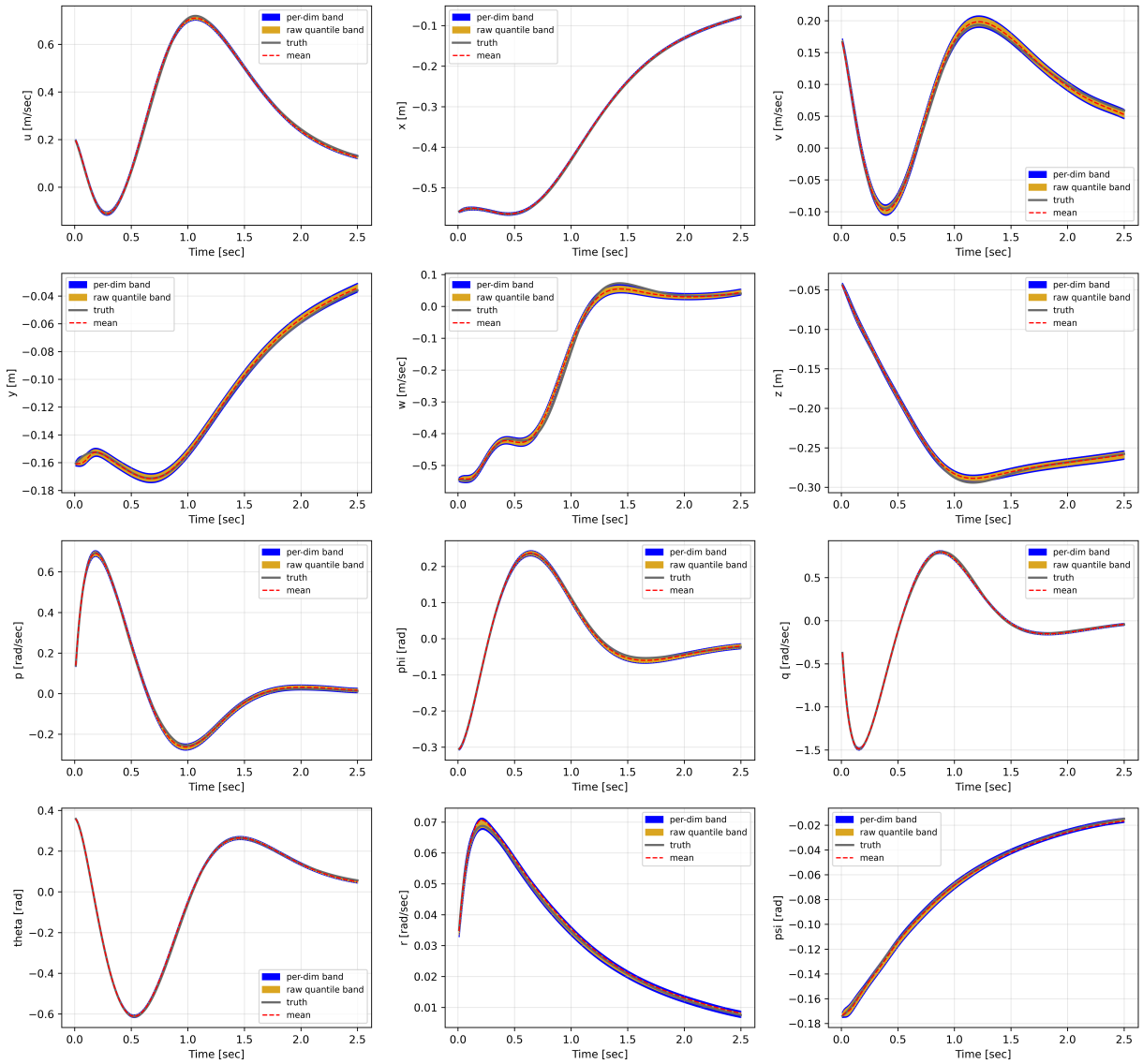

Figure 5: Full twelve-dimension version of Figure 2: conformal-calibrated trajectory bands for the *worst-case hold-out trajectory* under the 95% quantile model (baseline: ℓ=128, m=251, seed 42) prior to conformal calibration. Solid gray is the true trajectory; red dashed is the model's mean prediction; gold is the raw quantile band before calibration; blue is the per-dim conformal-calibrated band at 95% joint coverage.

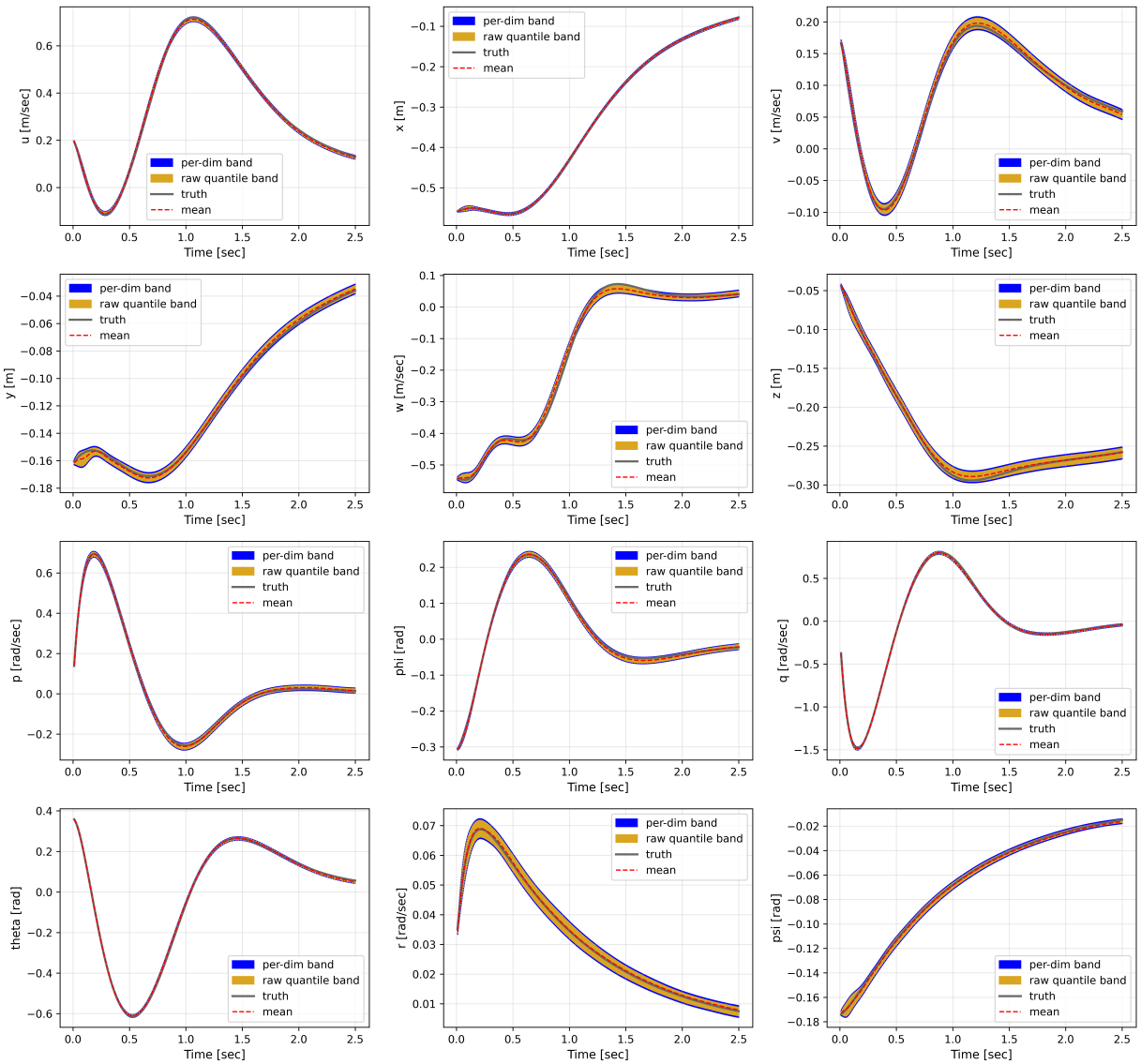

Figure 6: Same as Figure 5, but for the 99% quantile model. The raw quantile bands (gold) are wider than at the 95% target, and the conformal correction is correspondingly smaller; the calibrated band (blue) achieves 99% joint coverage at a band width comparable to the 95% case.

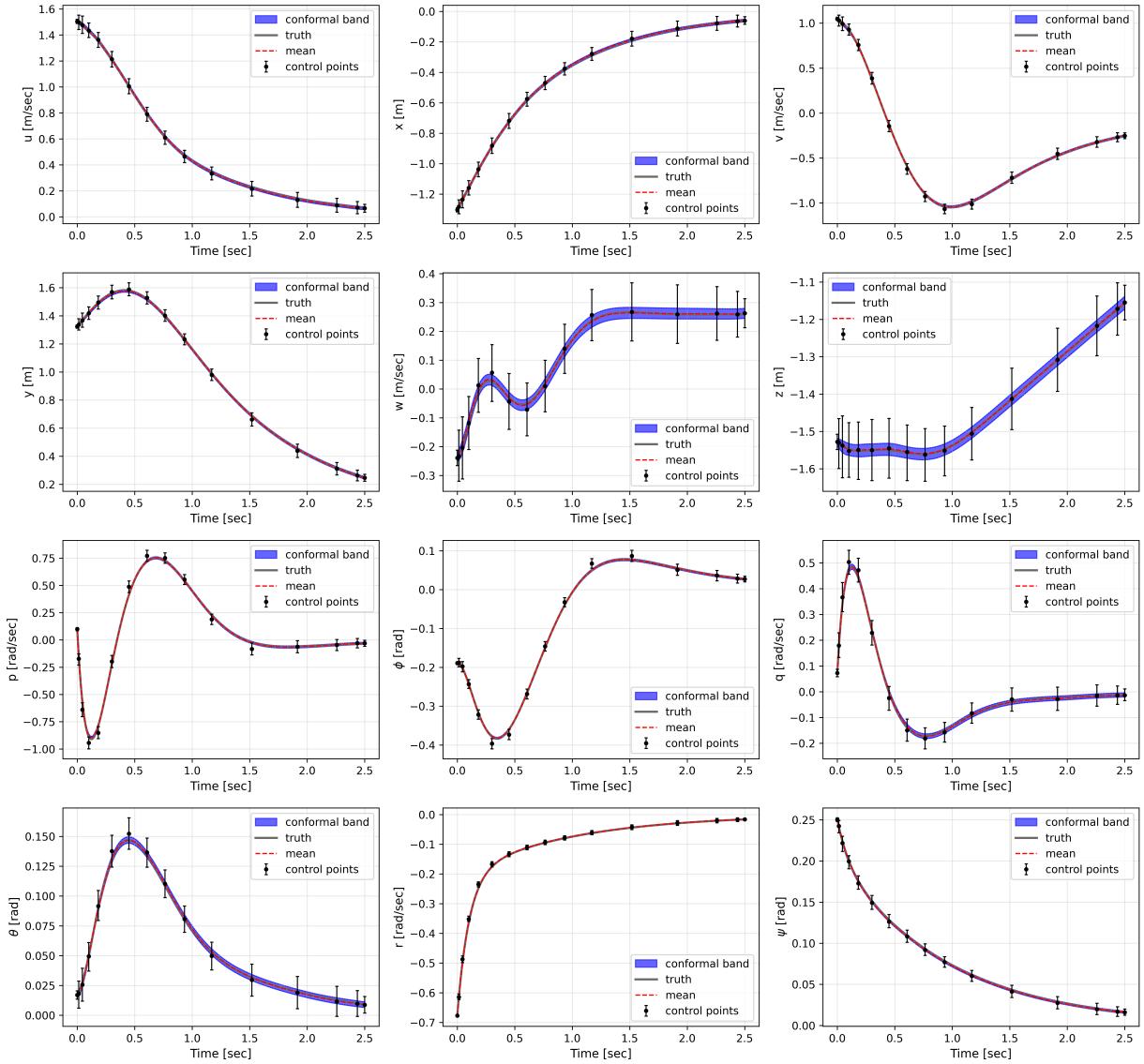

Figure 7: Full twelve-dimension version of Figure 3: calibrated trajectory bands and control-point intervals for one test trajectory (ℓ=128, m=251, 95% joint coverage). Gray solid is the true trajectory; red dashed is the mean prediction; the blue band is the per-dim conformal-calibrated band; black dots with vertical bars show the computed control points and their intervals at the Greville abscissae of the knot vector. There is no simulated noise on the initial condition. The generation process is identical to Figure 5 with the added computation of spline control points and their intervals, per Section 6.

E Initial Condition Uncertainty Propagation

Consider an initial condition x_0 with uncertainty, assumed to be Gaussian ($N(\mu = x_0, \sigma_0)$). Sampling and computing trajectories for a ball of initial condition values near x_0 is embarrassingly parallel and the combined set of quantiles can be used to create a marginal probability and uncertainty on the upper and lower quantiles of the trajectory. The perturbation set is a single batched forward pass; we assume knots are fixed based on x_0. For a given bound y and sampled points from the distribution x' we use

$$p(y \mid x_0) = \int p(y \mid x')\, p(x' \mid x_0)\, dx'$$

to develop probabilistic upper and lower quantiles. This approach combines uncertainty from measurement noise with model uncertainty to create a reasonable guess of how initial condition uncertainty affects the trajectory's upper and lower quantile bands. An illustrative example is shown below in Figure 8. Note that the magnitude of the noise in each dimension was set arbitrarily (near 0.01) and the true effect of initial condition uncertainty depends on the assumed covariate shift. We set aside a full covariate-shift analysis based on Tibshirani et al. (2019) as future work.

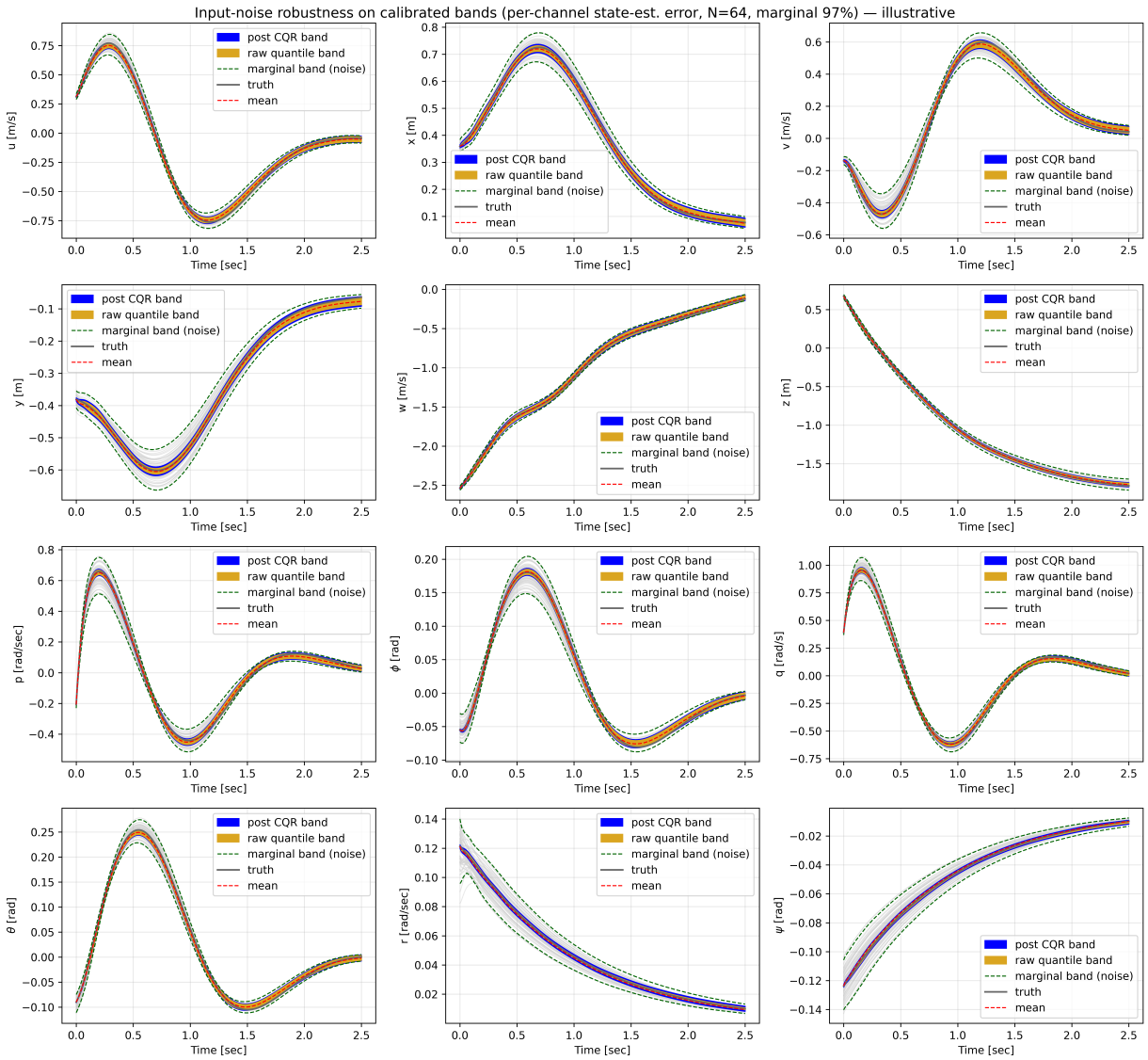

Figure 8: Input-noise propagation for one initial condition with a small amount of noise in each dimension. Black is truth; red dashed is the mean prediction; gold is the raw quantile band; blue is the per-dim conformal-calibrated band; green dashed is the marginal band from perturbing the IC with representative sensor-noise magnitudes and taking mixture quantiles across the perturbation set. This does *not* provide coverage guarantees in the sense of Tibshirani et al. (2019) and we intend a full study with a justified model for the distribution shift.