HyperKKL: Enabling Non-Autonomous State Estimation through Dynamic Weight Conditioning

Yahia Salaheldin Shaaban[1] Salem Lahlou[1] Abdelrahman Sayed Sayed[2]

[1] MBZUAI, Abu Dhabi, UAE [2] Univ Gustave Eiffel, France

{Yahia.Shaaban, Salem.Lahlou}@mbzuai.ac.ae | abdelrahman.ibrahim@univ-eiffel.fr | Code: github.com/yehias21/HyperKKL

Problem & Motivation

KKL observers lift a nonlinear system into a quasi-linear latent space via an injective map $\mathcal{T}$, where state estimation reduces to a stable linear filter inverted by a left inverse $\mathcal{T}^*$. Existing learning-based KKL methods assume autonomous dynamics and fail under exogenous inputs u(t).

Key Questions

Q1. Is input conditioning necessary for KKL observers on controlled systems? Which is more effective augmenting the latent dynamics or adapting the maps?

Q2. Why are hypernetworks the right architectural mechanism for input conditioning?

System & KKL Framework

Controlled nonlinear system:

$$\dot{x}(t) = f(x(t), u(t)), \qquad y(t) = h(x(t))$$

A KKL observer lifts it into a stable latent system $z = \mathcal{T}(x, t)$ evolving as:

$$\dot{z}(t) = A\,z(t) + B\,h(x(t))$$

with A Hurwitz, (A, B) controllable, and $\mathcal{T}$ satisfying the PDE:

$$\frac{\partial \mathcal{T}}{\partial x} f(x, u) + \frac{\partial \mathcal{T}}{\partial t} = A\,\mathcal{T}(x, t) + B\,h(x)$$

$\mathcal{T}$ must be time-varying to absorb u(t). We learn $\mathcal{T}$ and $\mathcal{T}^*$ with neural networks.

Geometric Picture

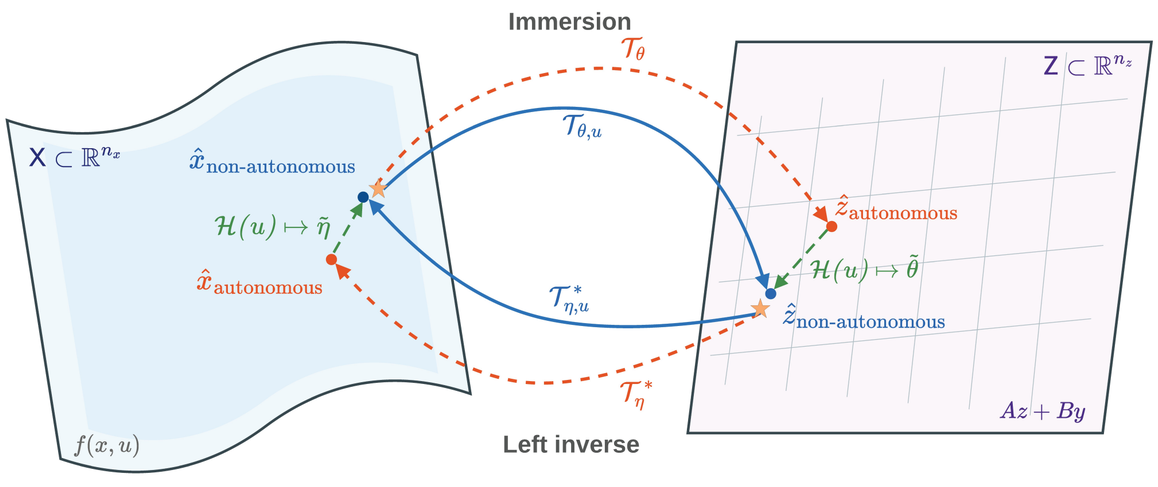

Figure 1. Autonomous (dashed) vs. non-autonomous (solid) transformations. Input u(t) shifts the latent image; time-varying maps $\mathcal{T}$, $\mathcal{T}^*$ are required.

Experimental Validation

Four benchmarks × four input regimes {zero, constant, sinusoid, square}. Metric: SMAPE over 100 random initial conditions with Gaussian process & measurement noise ($\sigma^2 = 0.01$).

Method	Duffing Zero	Duffing Const	Duffing Sin	Duffing Sqr	Van der Pol Zero	Van der Pol Const	Van der Pol Sin	Van der Pol Sqr
Autonomous	19.3	74.6	37.1	40.7	17.7	32.6	20.2	22.2
Curriculum	33.0	80.5	67.3	71.9	51.4	76.3	81.2	81.4
HyperKKL$_{obs}$	**5.6**↓	**21.1**↓	**22.7**↓	**21.3**↓	5.3↓	**26.0**↓	**15.5**↓	**17.9**↓
HyperKKL$_{dyn}$	8.2↓	57.3↓	29.7↓	30.7↓	**5.0**↓	32.7	18.8↓	20.8↓

Method	Rössler Zero	Rössler Const	Rössler Sin	Rössler Sqr	FitzHugh–Nagumo Zero	FitzHugh–Nagumo Const	FitzHugh–Nagumo Sin	FitzHugh–Nagumo Sqr
Autonomous	64.9	66.9	64.8	64.9	**9.8**	61.4	41.3	45.0
Curriculum	98	103.1	103.0	103.3	59.6	67.0	55.6	54.9
HyperKKL$_{obs}$	64.4↓	68.0	66.5	68.8	9.9	**42.1**↓	**30.9**↓	**32.5**↓
HyperKKL$_{dyn}$	**52.7**↓	**55.4**↓	**53.5**↓	**54.0**↓	17.0	54.6↓	40.8↓	44.5↓

Key Findings

✓ Average 29% SMAPE reduction across non-zero input regimes,

✓ HyperKKL**dyn** dominates the chaotic Rössler (−17% over autonomous)

✗ The limitation is architectural, not data.

HyperKKL Framework

We approximate $\mathcal{T}(x, t)$ and $\mathcal{T}^*(z, t)$ with neural networks whose weights are conditioned on the input history u[t−ω, t]. Two complementary strategies:

HyperKKL$_{obs}$

Augmented Observer

- Static $\mathcal{T}$, $\mathcal{T}^*$ (frozen after pretraining)
- Input correction injected into latent dynamics via Φ_ϖ
- GRU → latent embedding ℓ → injection MLP
- ~10K added parameters

HyperKKL$_{dyn}$

Dynamic Transformation Maps

- Time-varying $\mathcal{T}(\cdot; \theta(t))$, $\mathcal{T}^*(\cdot; \eta(t))$
- Hypernetwork $\mathscr{H}$ generates low-rank weight deltas from u[t−ω, t]
- No injection term ($\Phi \equiv 0$)
- ~45K added parameters

Training architecture (Dynamical hyperkkl)

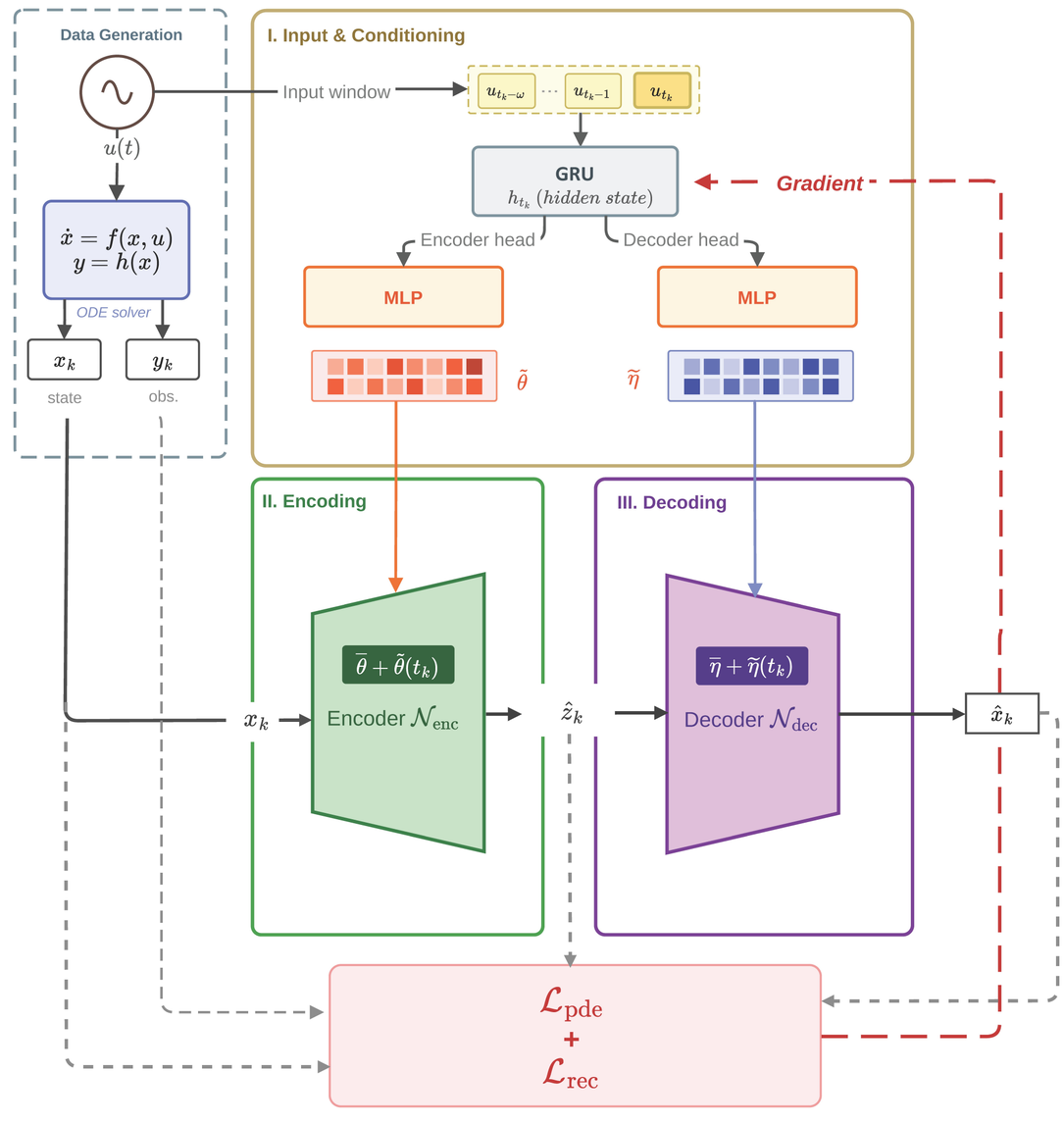

Figure 2. HyperKKL_dyn training. A shared GRU encodes u[t−ω, t] into a context h_t; MLP heads predict weight perturbations $\tilde{\theta}(t)$, $\tilde{\eta}(t)$ added to frozen base weights $\bar{\theta}$, $\bar{\eta}$. Losses L_pde + L_rec propagate only through the hypernetwork.

Qualitative Result: Duffing, Square Input

HyperKKL variants track the ground truth across abrupt input transitions; the autonomous baseline accumulates persistent phase drift.