Deep Operator Networks in Control Theory: concepts and applications

Umberto Biccari
Chair of Computational Mathematics, Universidad de Deusto, Bilbao, Spain
umberto.biccari@deusto.es
cmc.deusto.es

Joint work with **Jun Chen** and **Kaijing Lyu** - Beijing Institute of technology

The Mathematics of Scientific Machine Learning and Digital Twins
Erice, November 19-25 2025

Operator Learning and DeepONet

Operator Learning: from functions to operators

Machine Learning learns functions $f : \mathbb{R}^n \to \mathbb{R}^m$. **Operator Learning** aims to learn mappings between functions, i.e. operators $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$, with

- $\mathcal{X}$ a space of **input functions** $u : D \to \mathbb{R}$.
- $\mathcal{Y}$ a space of **output functions** $\mathcal{G}[u] : U \to \mathbb{R}$.

Why it matters?

- Many problems in science and engineering are governed by function-to-function laws.
- Classical numerical methods can approximate one particular case; OL generalizes across families of solutions, enabling **real-time computations** without having to solve the equation every time.

$\mathcal{G}[a](x) = z(x)$, where

$$\begin{cases} \mathcal{L}_a z = f & x \in \Omega \subset \mathbb{R}^d \\ z|_{\partial\Omega} = g \end{cases}$$

maps coefficients $a(x)$ to solutions $z(x)$.

$\mathcal{G}[x_0](t) = u(t)$, where

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) & t \in (0, T) \\ x(0) = x_0,\ x(T) = x_T \end{cases}$$

maps input data x_0 to controls $u(t)$.

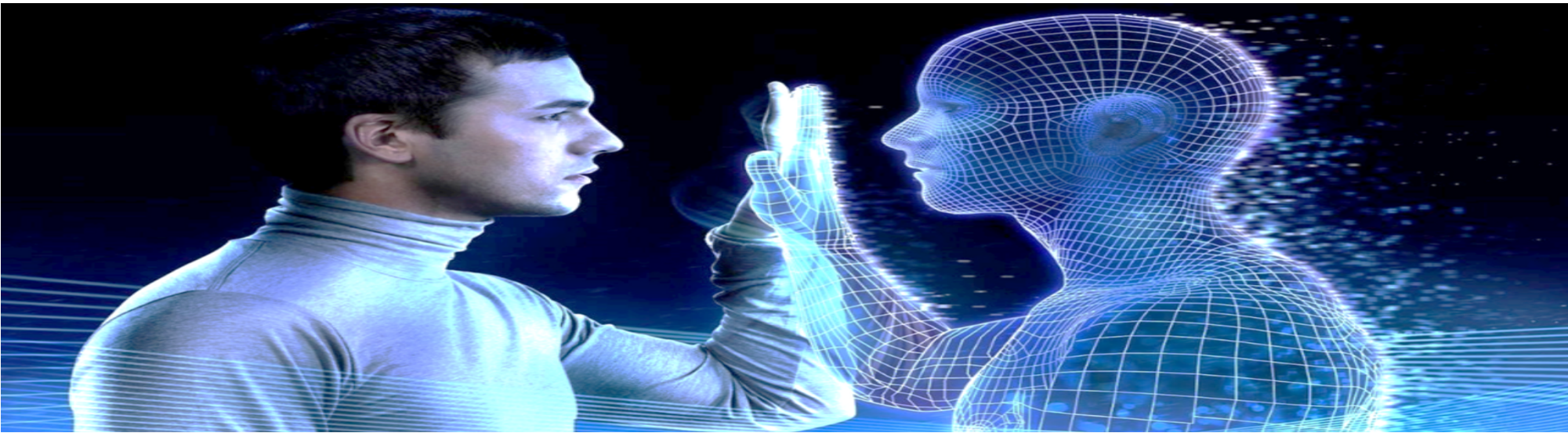
Operator Learning for Digital Twins

A Digital Twin needs a fast and reliable way to predict how a system will evolve under changing conditions. OL can learn the entire mapping from inputs to outputs of a physical system, allowing the twin to update and respond in real time without re-solving a PDE or a large-scale model.

Example: in laser powder-bed fusion additive manufacturing, a DT can use a neural-operator surrogate to map laser parameters to thermal/melt-pool fields, enabling real-time optimization of the process and defect minimization.

DEEP NEURAL OPERATOR ENABLED DIGITAL TWIN MODELING FOR ADDITIVE MANUFACTURING

Ning Liu[1*], Xuxiao Li[1], Manoj R. Rajanna[1], Edward W. Reutzel[2], Brady Sawyer[2], Prahalada Rao[3], Jim Lua[1], Nam Phan[4], and Yue Yu[5*]



Operator Learning approaches

Method	Core Idea	Bibliography
Fourier Neural Operator (FNO)	Learns the integral kernel of the operator in Fourier space.	Li, Kovachki, Azizzadenesheli, Bhattacharya, Stuart, & Anandkumar, 2021. LINK
Physics-Informed Neural Operator (PINO)	Incorporates PDE residuals into the loss function.	Li, Zheng, Kovachki, Jin, Chen, Liu, Azizzadenesheli, & Anandkumar, 2023. LINK
General Neural Operator Transformer (GNOT)	Uses self-attention to capture long-range dependencies.	Hao, Wang, Su, Ying, Dong, Liu, Cheng, Song, & Zhu, 2023. LINK
Deep Operator Network (DeepONet)	Learns operators via a separable branch–trunk architecture that maps input functions to outputs at any evaluation point.	Lu, Jin & Karniadakis, 2019. LINK
Deep Neural ODE Operator Network (NODE-ONet)	Embeds neural ODE dynamics within operator learning.	Li, Liu, Song, Yue, & Zuazua, 2025. LINK

Deep Operator Network (DeepONet)

$$\mathcal{G} \sim \boldsymbol{\beta}(\boldsymbol{\theta}, \boldsymbol{u}) \cdot \boldsymbol{\tau}(\boldsymbol{\theta}, y)$$

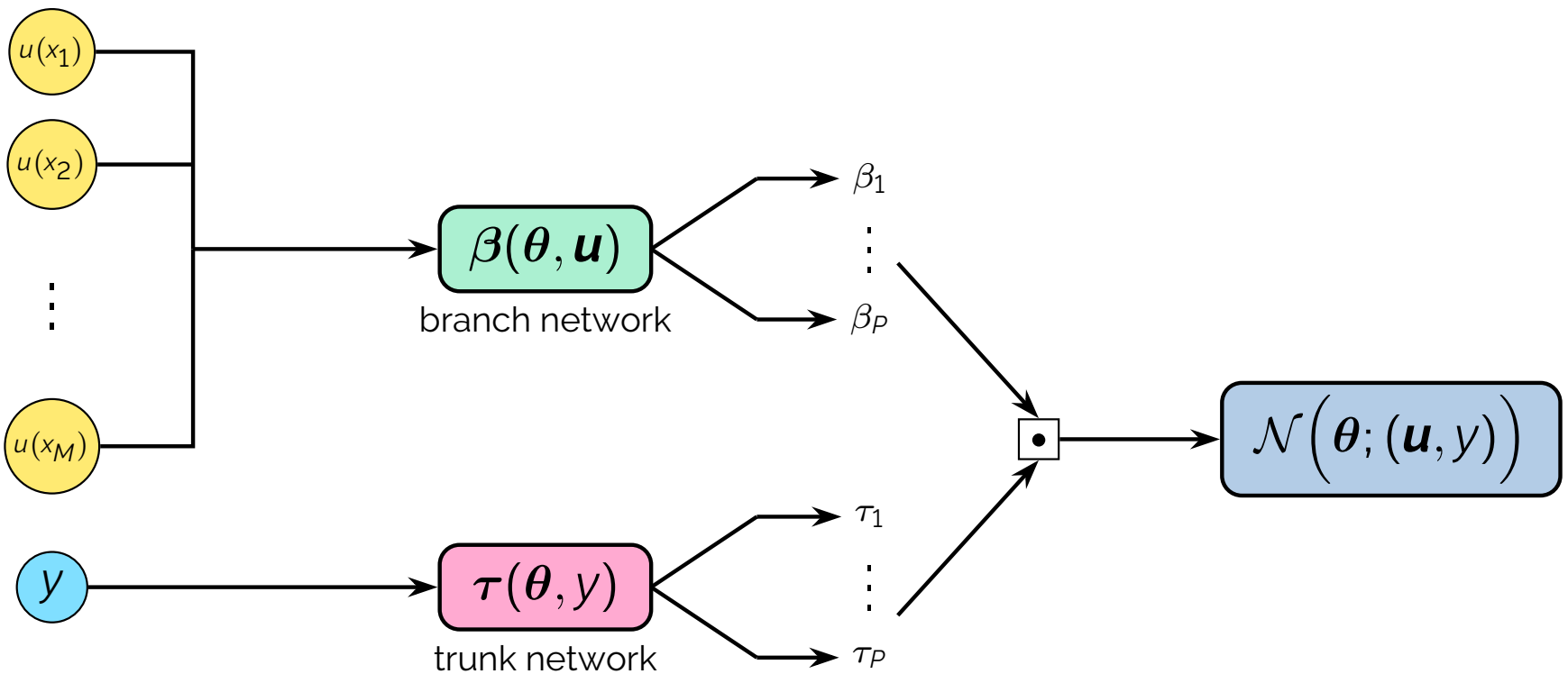
$$\boldsymbol{\beta}(\boldsymbol{\theta}, \boldsymbol{u}) := \left(\sum_{i=1}^{N} c_i^k \sigma \left(\sum_{j=1}^{M} \xi_{ij}^k u(x_j) + \omega_i^k \right) \right)_{k=1}^{M}$$

$$\boldsymbol{\tau}(\boldsymbol{\theta}, y) := \left(\sigma \left(w_k \cdot y + \eta_k \right) \right)_{k=1}^{M}$$

$$\begin{array}{ccc} \mathcal{X} & \xrightarrow{\mathcal{G}} & \mathcal{Y} \\ \downarrow \mathcal{E} & & \uparrow \mathcal{R} \\ \mathbb{R}^M & \xrightarrow{\mathcal{A}} & \mathbb{R}^P \end{array}$$

$$\mathcal{G} \sim \mathcal{R} \circ \mathcal{A} \circ \mathcal{E}$$

- $\mathcal{E} : u \mapsto \boldsymbol{u}$ encoder
- $\mathcal{A} : \boldsymbol{u} \mapsto \boldsymbol{\beta}(\boldsymbol{\theta}, \boldsymbol{u})$ approximation
- $\mathcal{R} : \boldsymbol{\beta}(\boldsymbol{\theta}, \boldsymbol{u}) \mapsto \boldsymbol{\beta}(\boldsymbol{\theta}, \boldsymbol{u}) \cdot \boldsymbol{\tau}(\boldsymbol{\theta}, y)$ reconstruction



Where does this architecture come from?

Compared with other existing operator learning methodologies, **DeepONet fulfills a proven Universal Approximation (UA) principle**.

Lu, Jin, Pang, Zhang & Karniadakis, *Learning nonlinear operators via deeponet based on the universal approximation theorem of operators*, Nat. Mach. Intell., 2021.

Lanthaler, Mishra & Karniadakis, *Error estimates for DeepONets: a deep learning framework in infinite dimensions*, Trans. Math. Appl., 2022.

Method	Universal Appr.	Comments
DeepONet	**Yes**	Proven universal approximation theorem for continuous nonlinear operators $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$.
NODE-ONet	**Yes**	Improvement of DeepONet
FNO	**Partial**	No full universality result; preliminary analyses show convergence and stability under certain regularity assumptions.
PINO	**No**	No known universal approximation result
GNOT	**No**	No known universal approximation result

Where does this architecture come from?

If $\mathcal{G}$ is an operator between two function spaces, from the UA principle for functions we get

$$\mathcal{G}[u](y) \sim \sum_{k=1}^{N} c_k\left(\mathcal{G}[u]\right)\sigma(w_k \cdot y + \eta_k).$$

Cybenko, *Approximation by superpositions of a sigmoidal function*, Math. Control Signals Syst., 1989.

Leshno, Lin, Pinkus & Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural Netw., 1993.

Theorem (Universal Approximation principle for operators)

Suppose that $\sigma \in C^\infty(\mathbb{R})$ is not a polynomial, $\mathcal{X}$ is a Banach space, $K_1 \subset \mathcal{X}$, $K_2 \subset \mathbb{R}^d$ and $V \subset C(K_1)$ are compact sets, and $\mathcal{G} : V \to C(K_2)$ is a continuous operator. Then, for any $\varepsilon > 0$ there exist positive integers N, M and P, sensor points $\{x_j\}_{j=1}^M \subset K_1$ and parameters $\{(c_i^k, \xi_{ij}^k, \omega_i^k, w_k, \eta_k)\} \subset \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}$ for all $i \in [\![N]\!], j \in [\![M]\!]$ and $k \in [\![P]\!]$ such that

$$\left|\mathcal{G}[u](y) - \sum_{k=1}^{P}\sum_{i=1}^{N} c_i^k \sigma\left(\sum_{j=1}^{M} \xi_{ij}^k u(x_j) + \omega_i^k\right)\sigma\left(w_k \cdot y + \eta_k\right)\right| < \varepsilon, \qquad \begin{array}{l}\text{for all } u \in V \\ \text{and } y \in K_2\end{array}.$$

Chen & Chen, *Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems*, IEEE Trans. Neural Netw., 1995.

Error analysis

Error taxonomy

The total error E_{tot} between $\mathcal{G}$ and its DeepONet approximation $\mathcal{N}(\boldsymbol{\theta}; (\boldsymbol{u}, y))$ can be decomposed as

$$E_{\text{tot}} = E_{\text{approx}} + E_{\text{optim}} + E_{\text{gener}}$$

Component	Description	Controlled by
E_{approx}	**Approximation error:** intrinsic limitation of the architecture	Universal Approximation Theorem
E_{optim}	**Optimization error:** imperfect minimization of the empirical loss	Training algorithm convergence
E_{gener}	**Generalization error:** coming from learning the operator from a finite sample instead of knowing it everywhere	Data sampling measure μ; dataset size N

Error analysis - generalization

Definition (Learning setting for DeepONet)

Let $\mathcal{X} \hookrightarrow L^2(D)$ and $\mathcal{Y} \hookrightarrow L^2(U)$ be Banach spaces of real-valued functions defined on sets D and U. A **learning setting for DeepONet** is a pair $(\mu, \mathcal{G})$, where μ specifies how input functions are sampled and $\mathcal{G}$ defines the corresponding outputs, such that

1. **Sampling measure:** $\mu \in \mathcal{P}_2(\mathcal{X})$ is a probability measure on $\mathcal{X}$ with finite second moments;
2. **Regularity of support:** there exists a Borel set $A \subset \mathcal{X}$ of continuous functions such that $\mu(A) = 1$;
3. **Target operator:** $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$ is a Borel measurable operator with finite energy: $\|\mathcal{G}\|_{L^2(\mu)} < \infty$

Sampling measure: input functions are drawn from a well-defined probability law with finite energy, ensuring the learning problem is statistically meaningful.

Regularity of support: guarantees that inputs can be represented by their sensor values, making the operator learnable from discrete data.

Target operator: ensures the outputs have finite variance, so the expected loss and the generalization error are well-defined in $L^2(\mathcal{X} \times U; d\mu)$.

In this framework, the generalization error scales as $E_{\text{gener}} \sim N^{-1/4}$

Lanthaler, Mishra & Karniadakis, *Error estimates for DeepONets: a deep learning framework in infinite dimensions*, Trans. Math. Appl., 2022.

DeepONet in control theory

Applications of DeepONet in control theory

Representative studies illustrating how DeepONet has been deployed for different classes of control problems

Control framework	DeepONet role	Bibliography
ODE and PDE control	Learn initial-datum-to-control maps	García-Cervera, Kessler, Pedregal & Periago, 2025. LINK
	Learn backstepping kernels or boundary-to-state maps	Qi, Zhang & Krstic, 2024. LINK
Adaptive control	Learn adaptive gain or parameter-to-control operators	Bhan, Shi & Krstic, 2023. LINK
Model Predictive Control	Replace dynamics in MPC with learned operators	de Jong, Shukla & Lazar, 2025. LINK
Optimal control and inverse problems	Learn HJB or inverse operators	Lee & Kim, 2025. LINK

LQR stabilization via Riccati

Control problem

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) & t > 0 \\ x(0) = x_0 \end{cases}$$

$x, x_0 \in \mathbb{R}^N \quad A \in \mathbb{R}^{N \times N}$

$B \in \mathbb{R}^{N \times M} \quad u \in \mathbb{R}^M$

Asymptotic stabilization: $x(t) \to 0$ as $t \to +\infty$

LQR: $J(u) = \int_0^\infty \left(x^\top Q x + u^\top R u \right) dt$

Optimal control: $u(t) = -R^{-1} B^\top P x(t)$

Riccati: $A^\top P + PA - PBR^{-1}B^\top P + Q = 0$

$0 \preceq Q \in \mathbb{R}^{N \times N}$ and $0 \prec P \in \mathbb{R}^{N \times M}$

Required: (A, B) stabilizable

$(A, Q^{1/2})$ detectable

Goal

Use DeepONet to learn the continuous operator

$$\begin{array}{rccc} \mathcal{G}: & \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times M} \times \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times M} & \to & \mathbb{R}^{N \times N} \\ & (A, B, Q, R) & \mapsto & P \end{array}$$

Chen & Biccari, *Learning Algebraic Riccati Equation solutions via DeepONet*, in preparation.

LQR stabilization via Riccati

Theorem

Assume the standard LQR hypotheses (A, B) stabilizable and $(A, Q^{1/2})$ detectable, so the true Riccati solution P exists and yields the Hurwitz matrix $A_{\text{stab}} = A - BR^{-1}B^{\top}P$. Let $\widehat{P} = \mathcal{G}(A, B)$ be the output of DeepONet. If

$$\|\widehat{P} - P\| < \frac{\lambda_{\min}\left(Q + PBR^{-1}B^{\top}P\right)}{2\|PBR^{-1}B^{\top}\|}, \tag{1}$$

then $\widehat{A}_{\text{stab}} = A - BR^{-1}B^{\top}\widehat{P}$ is Hurwitz.

LQR stabilization via Riccati

Brunovsky canonical form

If (A, B) is controllable, there exists $T \in \mathbb{R}^{N\times N}$ nonsingular s.t. $(A_c, B_c) = (T^{-1}AT, T^{-1}B)$ has **block-diagonal structure**

$$A_c = \mathrm{diag}(A_1, \ldots, A_S), \quad B_c = \mathrm{diag}(B_1, \ldots, B_S)$$

$$A_i = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{iN_i} & \end{pmatrix}, \quad B_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

$$N_1 + N_2 + \ldots + N_S = N$$

Known facts

- Each pair (A_i, B_i) controls a chain of N_i state variables.
- The Brunovsky form is unique (up to reordering of blocks) and represents all possible controllable dynamics of dimension N.

Numerical experiments in dimension $N = 3$

$$A_c = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ a_1 & a_2 & a_3 \end{pmatrix} \quad B_c = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$A_c = \begin{pmatrix} 0 & 1 & 0 \\ a_{11} & a_{12} & 0 \\ 0 & 0 & a_{21} \end{pmatrix} \quad B_c = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$A_c = \begin{pmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{pmatrix} \quad B_c = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

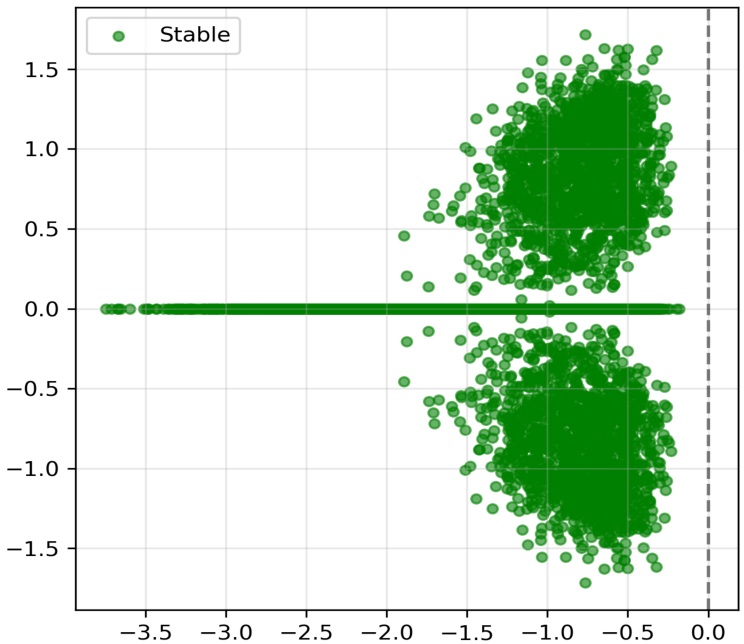
The dataset is built sampling matrices A_c in the three possible Brunovsky configurations, so to cover in each case the following possible situations

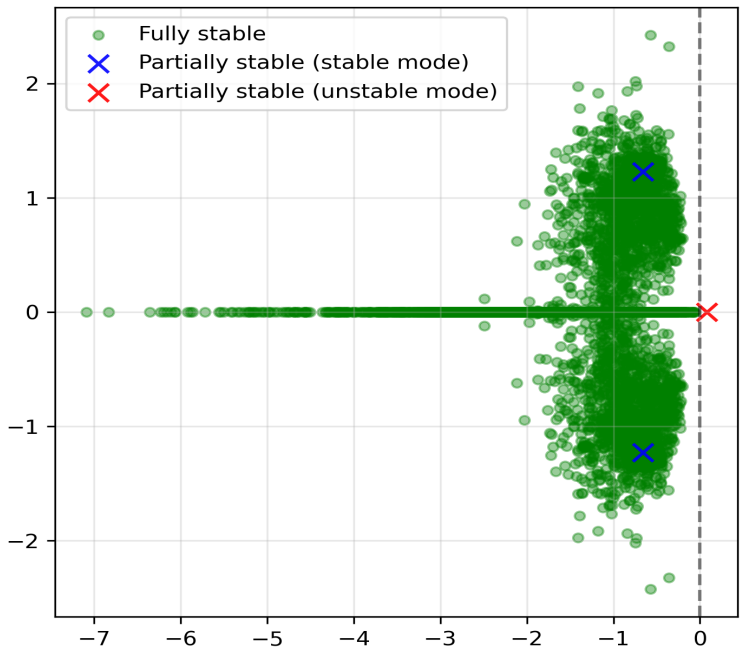
1. A_c **fully stable**: all the eigenvalues with negative real part.
2. A_c **unstable**: at least one eigenvalue with positive real part.
3. A_c **marginal**: some but not all the eigenvalues with zero real part.
4. A_c **fully periodic**: all the eigenvalues with zero real part.

When A_c is fully stable, the dynamics does not need a control u to decay to zero. But since the Riccati operator $\mathcal{G}(A, B) \mapsto P$ is defined and continuous for **all stabilizable systems**, including fully stable matrices in the dataset is necessary. Learning the operator's global behavior requires sampling all possible dynamics to capture the full smooth domain of the mapping.

Numerical experiments in dimension $N = 3$

Riccati solution P (left) anf $\hat{P}$ (right) computed for 2800 controllable **test** pairs (A, B).





The true solution P stabilizes all the 2800 dynamics in the test dataset. The NO solution $\hat{P}$ stabilizes 99.89% of the dynamics in the test dataset.

P CPU time	$\hat{P}$ CPU time	Speedup ↑
11.3*ms*	0.76*ms*	14.88x

Stabilization of stochastic PDE-ODE systems

$$
\begin{aligned}
&\dot{X}(t) = AX(t) + Bz(0,t) && t \in \mathbb{R}^+ \\
&X(0) = X_0 \\
&\partial_t w(x,t) = -\Lambda^+(t)\partial_x w(x,t) + \Sigma^{++}(t)w(x,t) + \Sigma^{+-}(t)z(x,t) && (x,t) \in (0,1) \times \mathbb{R}^+ \\
&\partial_t z(x,t) = \Lambda^-(t)\partial_x z(x,t) + \Sigma^{-+}(t)w(x,t) + \Sigma^{--}(t)z(x,t) && (x,t) \in (0,1) \times \mathbb{R}^+ \\
&w(0,t) = Q(t)z(0,t) + CX(t) && t \in \mathbb{R}^+ \\
&z(1,t) = R(t)w(0,t) + U(t) && t \in \mathbb{R}^+ \\
&w(x,0) = w_0,\ z(x,0) = z_0 && x \in (0,1)
\end{aligned}
$$

States

$X \in \mathbb{R}^2$
$(w,z) \in \mathbb{R}^3 \times \mathbb{R}$

Stochastic parameters

$\Lambda^+ \in \mathbb{R}^{3\times 3}$ $\quad \Lambda^- \in \mathbb{R}$ $\quad \Sigma^{++} \in \mathbb{R}^{3\times 3}$
$\Sigma^{+-} \in \mathbb{R}^{3\times 1}$ $\quad \Sigma^{-+} \in \mathbb{R}^{1\times 3}$ $\quad \Sigma^{--} \in \mathbb{R}$
$Q \in \mathbb{R}^{3\times 1}$ $\quad R \in \mathbb{R}^{1\times 3}$

$$\mathscr{S} = \left\{\Lambda^+, \Lambda^-, \Sigma^{++}, \Sigma^{+-}, \Sigma^{-+}, \Sigma^{--}, Q, R\right\}$$

Control

$U \in \mathbb{R}$

Goal

Compute $U(t) : \mathbb{R}^+ \to \mathbb{R}$ stabilizing the dynamics:

$$\mathbb{E}\left[p(t)\right] \leq c_1 e^{-c_2 t} p(0), \quad p(t) = |X(t)|^2 + \int_0^1 \left(\|w(x,t)\|^2 + |z(x,t)|^2\right) dx$$

Lyu, Biccari, Wang, *Robust stabilization of hyperbolic PDE-ODE systems via Neural Operator-approximated gain kernels*, Preprint, 2025.

Backstepping

For the **nominal** system with parameters $\mathscr{S}_0 = \left\{\Lambda_0^+, \Lambda_0^-, \Sigma_0^{++}, \Sigma_0^{+-}, \Sigma_0^{-+}, \Sigma_0^{--}, Q_0, R_0\right\}$

$$U(t) = -R_0 w(1,t) + \int_0^1 \left(K(1,\xi)w(\xi,t) + N(1,\xi)z(\xi,t)\right) d\xi + \gamma(1)X(t)$$

$$\begin{aligned}
&\Lambda_0^- K_x(x,\xi) = K_\xi(x,\xi)\Lambda_0^+ + N(x,\xi)\Sigma_0^{-+} + K(x,\xi)\left(\Sigma_0^{++} - \Sigma_0^{--}\mathbf{I}_3\right) && x \in (0,1), \xi \in (0,x)\\
&\Lambda_0^- N_x(x,\xi) + \Lambda_0^- N_\xi(x,\xi) = K(x,\xi)\Sigma_0^{+-} && x \in (0,1), \xi \in (0,x)\\
&K(x,x)\left(\Lambda_0^-\mathbf{I}_3 + \Lambda_0^+\right) = -\Sigma_0^{-+} && x \in (0,1)\\
&\Lambda_0^- N(x,0) - K(x,0)\Lambda_0^+ Q_0 = \gamma(x)B && x \in (0,1)\\
&\Lambda_0^- \gamma'(x) = \gamma(x)A - \Sigma_0^{--}\gamma(x) - K(x,0)\Lambda_0^+ C && x \in (0,1)
\end{aligned}$$

Theorem

Assume that (A, B) are stabilizable. Then the control U built by backstepping stabilizes the nominal dynamics.

Backstepping with DeepONet

Solving numerically the backstepping kernel equations is most often computationally expensive, especially over highly-refined meshes. To overcome this, we can leverage DeepONet to learn the mapping $\mathcal{G} : \mathscr{S}_0 \mapsto (K, N, \gamma)$ from system parameters to the kernels.

Theorem

Let $\widehat{\mathcal{G}}$ be the DeepONet approximation of the backstepping kernels map $\mathcal{G}$, and $(\widehat{K}, \widehat{N}, \widehat{\gamma})$ the associated backstepping kernels. Then, if

$$\mathbb{E}\Big[\|S(t) - S_0\|\Big] \leq \varepsilon, \qquad \text{for all } t > 0, S \in \mathscr{S} \text{ and } S_0 \in \mathscr{S}_0$$

the control

$$\widehat{U}(t) = -R_0 w(1,t) + \int_0^1 \Big(\widehat{K}(1,\xi) w(\xi,t) + \widehat{N}(1,\xi) z(\xi,t)\Big)\, d\xi + \widehat{\gamma}(1) X(t)$$

stabilizes the stochastic dynamics.

Numerical experiments - simulations' configuration

ODE dynamics: $A = \begin{bmatrix} 0 & 2 \\ -2 & 0 \end{bmatrix}$, $B = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$

Deterministic parameters:

$$\Lambda_0^+ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1.01 & 0 \\ 0 & 0 & 0.98 \end{bmatrix} \qquad \Sigma_0^{++} = \begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.3 & 0 \\ 0 & 0 & 0.3 \end{bmatrix} \qquad \Sigma_0^{+-} = \begin{bmatrix} 0.5 \\ -0.1 \\ 0.2 \end{bmatrix}$$

$$\Sigma_0^{-+} = \begin{bmatrix} 0.3 & -0.2 & 0.1 \end{bmatrix} \qquad \Sigma_0^{--} = 0.3, \qquad Q_0 = \begin{bmatrix} 1 \\ 1.05 \\ 1 \end{bmatrix}$$

$$R_0 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

Stochastic parameter Λ^-, with nominal value $\Lambda_0^- = 1$ and five possible values

$$\Lambda_1^- = 0.8, \quad \Lambda_2^- = 1, \quad \Lambda_3^- = 1.1, \quad \Lambda_4^- = 1.2, \quad \Lambda_5^- = 1.5,$$

with transition probabilities $P_{ij}(\varrho, t)$ computed by solving the Kolmogorov equation

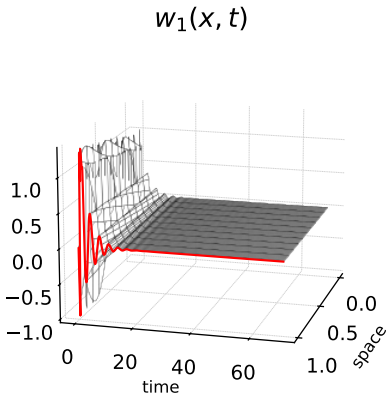
$$\partial_t P_{ij} = -c_j(t) P_{ij} + \sum_{k=1}^{r} P_{ik} \tau_{kj}(t),$$

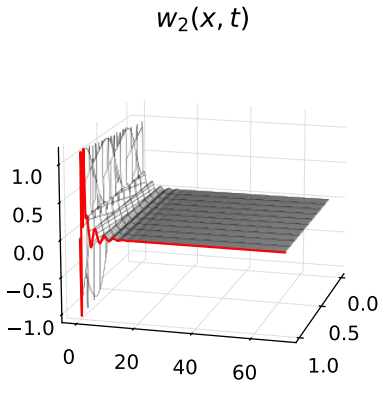
$$P_{ii}(\varrho, \varrho) = 1, \quad P_{ij}(\varrho, \varrho) = 0 \text{ for } i \neq j$$

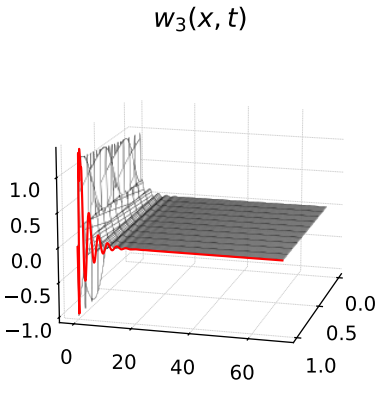
$$c_j(t) = \sum_{k \neq j = 1}^{r} \tau_{jk}(t)$$

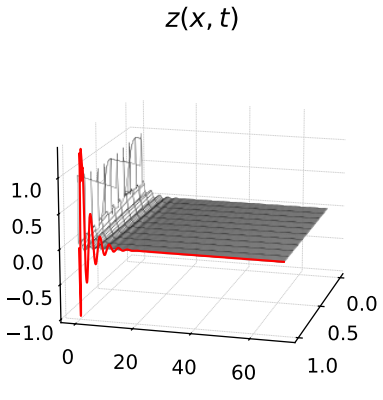
$$\tau_{ij}(t) = \begin{cases} 0 & \text{if } i = j \\ 20 & \text{if } i \in \{1, 5\} \\ 1 & \text{if } i \in \{2, 3, 4\}, j \in \{1, 5\} \\ 10\left(1 + 2\cos\left(10^{-3}(i + 5j)t\right)\right)^2 & \text{if } i \neq j \in \{2, 3, 4\} \end{cases}$$

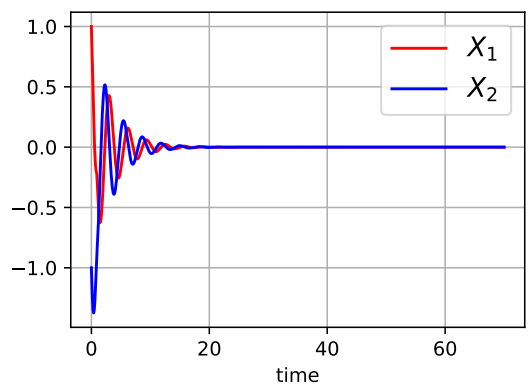
Numerical experiments - simulation results











dx	Analytical kernel CPU time	NO Kernel CPU time	Speedup ↑
0.1	0.038	0.021	1.81x
0.01	3.104	0.023	135x
0.005	11.805	0.025	472x
0.001	261.5	0.036	7263x

Conclusions

Critical conclusions on DeepONet

Advantages

Rigorous operator approximation framework. DeepONet is based on UA for operators, offering a solid theoretical foundation absent in most other OL architectures.

Fast inference. Once trained, DeepONet replaces iterative or PDE-based solvers with instantaneous inference, which is valuable in control and Digital Twins.

Limitations

Data-hungry. DeepONet requires large, rich datasets due to its $N^{-1/4}$ generalization rate.

High training cost. Despite efficient inference, training remains computationally demanding.

Scalability. Once we have a trained model for d-dimensional problems, can we adapt it easily to dimension $d + 1$?

Take-home message

DeepONet offers a rigorous UA guarantee, but its rigid architecture can limit flexibility. Other architectures, such as GNOT, may have an enhanced flexibility but at the cost of higher computational and memory demands.

How can we integrate these complementary strengths (rigorous theory, physics, scalability,...) instead of relying on a single architecture for all tasks?

THANKS FOR YOUR ATTENTION!!

The **CoDeFeL** project (ERC-2022-ADG) has received funding from the European Union's Horizon ERC Grants programme under grant agreement No. 101096251. The views and opinions expressed are solely those of the author(s) and do not necessarily reflect those of the European Research Council Executive Agency (ERCEA), the European Union or the granting authority who cannot be held responsible for them.

Additional funding

- PID2023-146872OB-I00-**DyCMaMod** (MINECO)

European Research Council

Established by the European Commission