Neural Conjugate Flows: A Physics-Informed Architecture with Flow Structure

Arthur Bizzi, Lucas Nissenbaum, João M. Pereira

Instituto de Matemática Pura e Aplicada (IMPA), Rio de Janeiro, Brazil
arthur.bizzi@impa.br, nissenbaum@impa.br, jpereira@impa.br

Abstract

We introduce Neural Conjugate Flows (NCF), a class of neural-network architectures equipped with exact flow structure. By leveraging topological conjugation, we prove that these networks are not only naturally isomorphic to a continuous group, but are also universal approximators for flows of ordinary differential equation (ODEs). Furthermore, topological properties of these flows can be enforced by the architecture in an interpretable manner. We demonstrate in numerical experiments how this topological group structure leads to concrete computational gains over other physics informed neural networks in estimating and extrapolating latent dynamics of ODEs, while training up to five times faster than other flow-based architectures.

Code — https://github.com/arthur-bizzi/Neural-Conjugate-Flows-AAAI

1 Introduction

The introduction of Physics-Informed Neural Networks (PINNs) (Raissi, Perdikaris, and Karniadakis 2019) has sparked interest in using neural networks to solve and discover differential equations. By encoding modeling into "Physics-informed" losses, networks $\mathcal{N}_\theta$ are trained to approximate *solutions* to differential equations:

$$\frac{d}{dt}\mathcal{N}_\theta(\mathbf{x}^0,t) = F\left(\mathcal{N}_\theta(\mathbf{x}^0,t)\right)$$
$$\iff \mathcal{L}(\theta) = \left|\left|\frac{d}{dt}\mathcal{N}_\theta(\mathbf{x}^0,t) - F\left(\mathcal{N}_\theta(\mathbf{x}^0,t)\right)\right|\right| = 0,$$

where $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a vector field, $\mathbf{x}^0 \in \mathbb{R}^n$ are initial conditions and $\mathcal{N}_\theta : \mathbb{R}^{n+1} \mapsto \mathbb{R}^n$ is a $\theta-$parameterized NN. $\mathcal{L}(\theta)$ denotes an equation loss relative to θ.

PINNs have since been applied to forward and inverse problems across a wide range of domains (De Ryck, Mishra, and Molinaro 2022; Patel et al. 2022; Mao, Jagtap, and Karniadakis 2020; Rao, Sun, and Liu 2021; Hasan et al. 2020).

Still, due to their black-box nature, PINNs struggle significantly with enforcing fundamental structural properties of the solutions. Thus, it is often necessary to encode constraints into PINNs through the use of additional Physics-based *losses*. These might include terms that reward proper conservation of Hamiltonians (Mattheakis et al. 2022), Lagrangians (Cranmer et al. 2020) and symmetries (Zhang et al. 2023).

PINNs are specially notorious for their issues with causal dependence on initial conditions, which manifest in the form of incorrect initial conditions or non-physical convergence to trivial solutions (see Fig. 1 for an example). Attempts have also been made to minimize these effects with modified losses, emulating causality (Wang, Sankaran, and Perdikaris 2022) or penalizing large gradients (Yu et al. 2022).

In contrast, few attempts have been made to tackle these issues with new physics-based *architectures*. Indeed, the vast majority of PINNs use general-purpose feed-forward architectures, such as the simple Multi-Layer Percetron (MLP) (Cuomo et al. 2022).

A remarkable solution to these causality issues may be found in the framework of Neural Ordinary Differential Equations (Neural ODEs) (Chen et al. 2018). This approach is instead based on modeling the *derivative* term of an ODE with a neural network:

$$\frac{d}{dt}\mathbf{x} = \mathcal{N}_\theta(\mathbf{x})$$

This ODE may then be solved with numerical solvers, using the adjoint method to calculate loss gradients.

This blend of neural and traditional methods leads to these networks having the structure of a *flow*, which we detail in Section 2. This leads to a solution that is automatically compliant to initial conditions and that implicitly satisfies causality and time reversibility, making Neural ODEs specially adequate for physics-informed contexts (Lai et al. 2021; O'Leary, Paulson, and Mesbah 2022).

However, the applicability of Neural ODEs is limited by the large computational overhead introduced by the sequential numerical solvers they are built upon. In particular, calculating gradients with the adjoint method leads to significant slowdowns (Kidger 2022). Regularizing and optimizing this process is an open problem (Finlay et al. 2020).

A computationally lighter alternative to Neural ODEs was proposed in (Biloš et al. 2021). However, this approach comes at the cost of the associative property of flows (see Section 2), leading to an incomplete group structure. In particular, although this approach may lead to correct initial conditions, it cannot enforce unicity or causality.

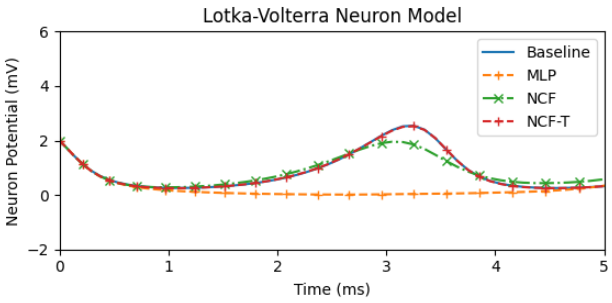


Figure 1: Spurious convergence of an MLP to equilibrium, denoting lack of uniqueness (see the Appendix for experimental details). The structure of NCFs makes this less likely.

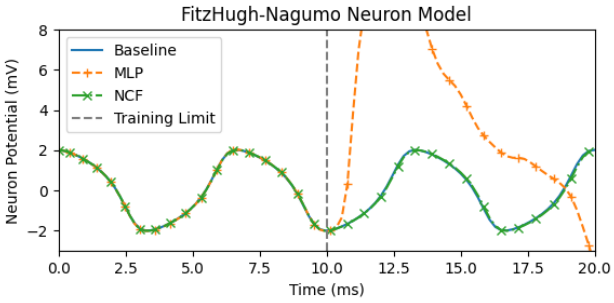


Figure 2: Extrapolation capacity of NCF versus an MLP (see Section 4 for details). The NCF is able to generalize beyond the training time of 10ms, the MLP is not.

1.1 Our Contributions

We propose a new flow-structured architecture, named *Neural Conjugate Flows* (NCFs). These models reproduce the structure of solution groups for differential equations exactly, by leveraging the dynamical-systems' framework of *topological flow conjugation*. To the best of our knowledge, the only other instance of conjugation being applied to machine learning is in the context of approximating Poincaré sections of flows (Bramburger, Brunton, and Kutz 2021), while Zhi et al. (2022) use it indirectly to warp trajectories of Neural ODEs .

In essence, NCFs attempt to topologically deform nonlinear vector fields into those associated with integrable systems. This deformation takes place as a conjugation mapping parameterized by an invertible neural network $\mathcal{H}_\theta$.

As a result, we have that:

- NCFs are flow operators, and as such automatically respect initial conditions and time causality.
- NCFs with conjugate affine flows are universal approximators for flow operators of autonomous ODEs.
- Several topological properties of the solution of the ODE may be enforced through properly choosing the inner flow operator.

In this sense, NCFs may be seen as topology-informed alternatives to Neural ODEs. Though both architectures share similar objectives, NCFs may be trained significantly faster. This is due to the fact that the simpler flows used here may be calculated in closed-form in a parallel fashion, as opposed to the intrinsically sequential calculation of numerical flows.

The paper is structured as follows. In Section 2, we revisit the notion of a flow operator, along with the notion of conjugation and affine flows. In Section 3 we introduce the general framework of Neural Conjugation and show that NCFs are universal approximators for flows of ODEs. Finally, in Section 4 we present numerical experiments and in Section 5 we present some limitations of the architecture, along with final remarks.

In the supplemental materials, we have included an Appendix with additional experiments, proofs and implementation details.

2 Flows and Conjugation

2.1 Flows

Consider the prototypical system of autonomous ordinary differential equations in $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$:

$$\frac{d}{dt}\mathbf{x} = F(\mathbf{x}), \quad \mathbf{x}(0) = \mathbf{x}^0 \tag{1}$$

Now let $\Phi : [0, T] \times \mathcal{X} \mapsto \mathcal{X}$ be the *flow operator* (or simply *flow*[1]) associated with the vector field F, defined as the map that propagates a initial state $\mathbf{x}$ through the ODE's dynamics for a given time interval t:

$$\Phi(t; \mathbf{x}^0) := \Phi^t \mathbf{x}^0 = \mathbf{x}(t) \tag{2}$$

It is a fundamental result in the theory of ODEs (Viana and Espinar 2021) that this operator should be not only continuous, but also behave as a *group*, as encoded in the following group properties:

I - Identity Element:

$$\Phi^0(\mathbf{x}) = \mathbf{x}, \quad \forall \mathbf{x} \in \mathcal{X} \tag{3}$$

II - Associativity:

$$\Phi^t \circ \Phi^\tau(\mathbf{x}) = \Phi^{t+\tau}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X},\ t, \tau \in \mathbb{R} \tag{4}$$

Together, **I** and **II** imply that flows are *invertible*:

"III" - Invertibility:

$$\Phi^{-t} \circ \Phi^t(\mathbf{x}) = \mathbf{x}, \quad \forall \mathbf{x} \in \mathcal{X},\ t \in \mathbb{R} \tag{5}$$

In many ways, these properties encode the notion of causality and well-posedness in physical systems:

- Systems with property **I** respect initial conditions;
- Systems with property **II** respect time causality and uniqueness of trajectories;
- Systems with property **III** enjoy time reversibility.

For MLP-PINNs, none of these properties are guaranteed; instead, they must be soft-enforced by means of different and often conflicting Physics-informed losses.

[1]Note that invertible 'flow' models, like Normalizing Flows (Papamakarios et al. 2021), are inspired by mathematical flows, but in general do not have true flow structure.

2.2 Topological Conjugation

Take two open subsets of $\mathbb{R}^n$, $\mathcal{X}_\Phi$ and $\mathcal{X}_\Psi$. Two flows Φ and Ψ are said to be (locally) *topologically conjugated* if there is a homeomorphism $H : \mathcal{X}_\Phi \to \mathcal{X}_\Psi$ such that:

$$\Phi^t(\mathbf{x}) = H^{-1} \circ \Psi^t \circ H(\mathbf{x}), \quad \forall \mathbf{x} \in \mathcal{X}_\Phi,\, t \in \mathbb{R} \tag{6}$$

where H^{-1} is the inverse of H. Intuitively, conjugated systems are related by a change of variables. Moreover, given a flow Ψ^t and a homeomorphism H, the conjugate operator $\Phi^t = H^{-1} \circ \Psi^t \circ H$ is also a flow:

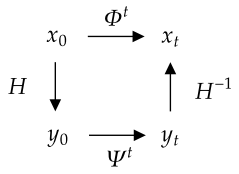
I - Identity Element:

$$\begin{aligned} \Phi^0(\mathbf{x}) &= H^{-1} \circ \Psi^0 \circ H(\mathbf{x}) \\ &= H^{-1} \circ H(\mathbf{x}) = \mathbf{x}. \end{aligned} \tag{7}$$

II - Associativity:

$$\begin{aligned} \Phi^t \circ \Phi^\tau(\mathbf{x}) &= H^{-1} \circ \Psi^t \circ H \circ H^{-1} \circ \Psi^\tau \circ H(\mathbf{x}) \\ &= H^{-1} \circ \Psi^t \circ \Psi^\tau \circ H(\mathbf{x}) \\ &= H^{-1} \circ \Psi^{t+\tau} \circ H(\mathbf{x}) = \Phi^{t+\tau}(\mathbf{x}). \end{aligned} \tag{8}$$

We may then use conjugation to construct new flows from known ones, as per the commutative diagram:



Fundamental theorems in the theory of dynamical systems leverage conjugation (Viana and Espinar 2021):

Flow-Box or **Tubular Flow Theorem:** Near a regular point p (one far from equilibria or cycles), a dynamical system is locally conjugated to a translation.

Hartman-Grobman Theorem: Near a hyperbolic equilibrium point p, a dynamical system is locally conjugated to its linearization around p (See Fig. A in the Appendix).

Underlying these theorems is the fact that conjugated vector fields are *topologically equivalent*: orbits in their phase spaces may be continuously deformed into one another. In particular, cycles and equilibria are preserved:

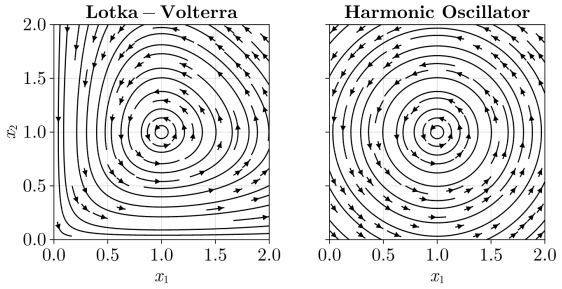


Figure 3: Neural Conjugates deform the orbits of the harmonic oscillator to match those of the Lotka-Volterra system.

We may leverage this by controlling the structure of the conjugate Ψ. In this work, we will use the fact that matrix exponentials are *Lie groups*, and thus have remarkable topologies (Humphreys 1972).

2.3 Affine Flows and Universality

We now restrict ourselves to affine systems. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. We have:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{b}, \;\; \mathbf{x}(0) = \mathbf{x}^0, \tag{9}$$

Their associated flows are well known, and may be given in terms of matrix exponentials[2]:

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x} + \int_0^t e^{\mathbf{A}\tau}\mathbf{b}\, d\tau \tag{10}$$

While this class of flows may at first seem too simple to be representative, we can leverage augmentation (or padding) to prove that any Lipschitz-continuous system of ODEs can be conjugated to an affine system.

To augment a system is to equip it with additional "dummy" dimensions, allowing wider, more representative models to be used (Dupont, Doucet, and Teh 2019). The increase in representation power enjoyed by systems in higher dimensions will be used both in our proofs and our implementation.

We will prove that any sufficiently well-behaved nonlinear system may be embedded in a larger dimensional manifold where it is essentially linear:

Theorem 1. Universality of Affine Conjugation. Let $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ be a Lipschitz-continuous vector field. Then for any positive integer m there exists an augmentation $\mathbf{a} \in \mathbb{R}^m$, a component $G : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^m$, a matrix $\hat{\mathbf{A}} \in \mathbb{R}^{(n+m)\times(n+m)}$ and a vector $\hat{\mathbf{b}} \in \mathbb{R}^{n+m}$ such that the following augmented system

$$\frac{d}{dt}\hat{\mathbf{x}} = \frac{d}{dt}\begin{bmatrix}\mathbf{x}\\ \mathbf{a}\end{bmatrix} = \begin{bmatrix}F(\mathbf{x})\\ G(\mathbf{x},\mathbf{a})\end{bmatrix} = \hat{F}(\hat{\mathbf{x}}) \tag{11}$$

is conjugated to the affine system:

$$\frac{d}{dt}\mathbf{y} = \hat{\mathbf{A}}\mathbf{y} + \hat{\mathbf{b}}. \tag{12}$$

In other words, we may extend nonlinear systems to a higher dimension in a way that unravels their orbits, so they can then be deformed to match the orbits of an affine system. This result can be thought as an extension of the Tubular-Flow Theorem: In essence, we may always choose an embedding that "destroys" the system's topology, making every point regular.

The proof, which we give in full in Appendix A, is constructive, using the flow operator of the ODE to construct the homeomorphism H.

This means that we may solve any sufficiently mild ODE as a part of a larger solution of the form:

$$\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{a}(t)\end{bmatrix} = H^{-1}\left(e^{\hat{A}t}H\left(\begin{bmatrix}\mathbf{x}^0\\ \mathbf{a}^0\end{bmatrix}\right) + \int_0^t e^{\hat{A}\tau}\hat{b}\, d\tau\right), \tag{13}$$

[2]In the Appendix, we give an alternate numerical formula that does not involve calculating integrals.

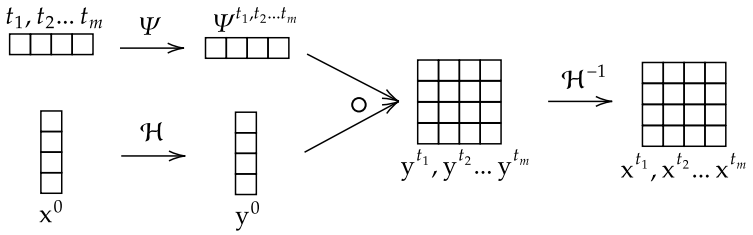


Figure 4: The NCF pipeline: Change variables to the conjugate manifold, iterate, then change back.

3 Neural Conjugate Flows

3.1 Neural Conjugation

Definition 1. Neural Conjugate Flows. A *Neural Conjugate Flow*[3] Φ^t is defined as:

$$\Phi^t = \mathcal{H}^{-1} \circ \Psi^t \circ \mathcal{H}, \tag{14}$$

where Ψ is a flow operator, $\mathcal{H}$ is an invertible neural network and $\mathcal{H}^{-1}$ is its inverse.

In principle, any invertible architecture could be used to approximate $\mathcal{H}$. For this paper, we have chosen the 'Coupling Layer' architecture, an Universal Approximator for homeomorphisms (see section 3.2).

Likewise, the flow Ψ could be either analytical or numerical in nature. For this paper, we will use the analytical affine flow, which is an universal approximator under conjugation, as discussed (see also section 3.3).

Inference in an NCF then takes place in three steps:

1. We feed the initial condition $\mathbf{x}^0$ to the network $\mathcal{H}$:
$$\mathbf{y}^0 = \mathcal{H}(\mathbf{x}^0).$$
2. We apply the flow Ψ evaluated at times $t_1, t_2, \ldots, t_m$:
$$\mathbf{y}^{t_i} = \Psi^{t_i}(\mathbf{y}^0), \quad i = 1, \ldots, m.$$
3. We feed the result to the inverse network $\mathcal{H}^{-1}$:
$$\mathbf{x}^{t_i} = \mathcal{H}^{-1}(\mathbf{y}^{t_i}), \quad i = 1, \ldots, m.$$

A schematic of this procedure is presented in Figure 4. A detailed view of our implementation may be found in Appendix B.

The combined universality of affine flows and Coupling Layers allows us to prove our main theorem: Affine Neural Conjugate Flows are universal approximators for flows of ODEs.

Theorem 2. Universality of Affine NCFs. Let Φ be a flow associated with a differentiable vector field. There are an augmentation $\hat{\Phi}$ and a Neural Conjugate Flow $\tilde{\Phi} = \mathcal{H}^{-1} \circ \Psi^t \circ \mathcal{H}$, with $\mathcal{H}$ a Coupling Layer ensemble and Ψ affine, such that $\tilde{\Phi}$ approximates $\hat{\Phi}$.

The proof is straightforward (see Appendix A). This result indicates that, given a sufficiently representative Coupling Layer, along with a properly crafted augmentation, NCFs can solve any differential equation where global existence is assured.

[3]Note that although time-one maps for NCFs could be used to implement normalizing flows, they are are two distinct concepts.

3.2 The Network $\mathcal{H}$

In principle, any invertible architecture could be used to represent $\mathcal{H}$. This includes approximately invertible ones, like AutoEncoders (Kingma and Welling 2022) and U-nets (Ronneberger, Fischer, and Brox 2015). In practice, however, we have not been able to make neural conjugates based on these models converge. We conjecture that having the group properties of flows *during* training (as opposed to after it) enhances convergence.

Our architecture of choice for $\mathcal{H}$ is a composition of Coupling Layers (Dinh, Sohl-Dickstein, and Bengio 2017) . These networks are universal approximators for diffeomorphisms (Teshima et al. 2020), whose inverse is easily computable.

Coupling Layers achieve perfect invertibility by applying neural networks to a split input, keeping half the dimensions constant at each layer (see Fig. 5). This comes at the cost of reduced representation power, as states are weakly coupled (Draxler et al. 2024).

To recover expressiveness, we again turn to augmentation. We concatenate a copy of $\mathbf{x}^0$ to the input, widening our model by a factor of 2 and allowing the Coupling Layers at each step to capture the entirety of the n-dimensional state $\mathbf{x}$ (see Fig. 6). We then solve a duplicated ODE[4]:

$$\frac{d}{dt}\begin{bmatrix}\tilde{\mathbf{x}}\\ \hat{\mathbf{x}}\end{bmatrix} = \begin{bmatrix}F(\tilde{\mathbf{x}})\\ F(\hat{\mathbf{x}})\end{bmatrix} \tag{15}$$

and take the output to be the average $\mathbf{x} = (\tilde{\mathbf{x}} + \hat{\mathbf{x}})/2$.

In addition to using more expressive invertible layers, augmentation allows us to use affine flows in $2n$ dimensions, leading to substantially more expressive conjugates Ψ. Moreover, by projecting onto a higher-dimensional space we aim to trigger the conditions for universality (see Theorem 1).

This comes at the cost of managing the additional 'twin' dimensions, as conjugation is necessarily dimension-preserving. Note that both $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$ must be trained as legitimate solutions to the problem; otherwise, if one of them is allowed to drift freely (as is done in e.g. the ANODEs framework (Dupont, Doucet, and Teh 2019)), the averaging projection would lead to trajectory crossing and the flow structure would be lost.

Empirically, this scheme can be interpreted as follows: although our flows are taking place in a $2n-$dimensional space, during training we attempt to restrict our dynamics to the codimension-n 'diagonal' manifold $(\mathbf{x}, \mathbf{x})$, where flow structure is preserved.

[4]See Fig (7) for a visual representation.

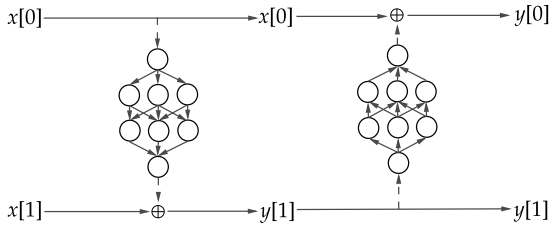


Figure 5: A standard coupling layer ensemble in $\mathbb{R}^2$. Inverse evaluation may be done by reversing the arrows and replacing $+$ for $-$. Notice that each MLP only 'sees' half the input, leading to reduced representation power.

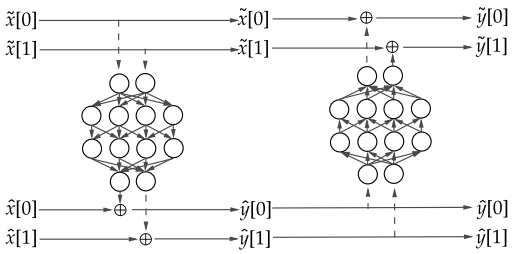


Figure 6: Our coupling layer ensemble. By twinning the input, we ensure each MLP sees both positions of the $\mathbf{x}$ vector, $\mathbf{x}[0]$ and $\mathbf{x}[1]$.

3.3 The Flow Ψ

Our architecture is best suited for systems for which some qualitative description is known. Theoretical or numerical results may provide insight into a system's topology, which can then be leveraged to choose a suitable conjugate. Explicit conjugates, though rare, should make for excellent flows Ψ.

Motivated by Universality theorem 1, we have set our conjugate flows Ψ to be affine flows. We then let the parameters $\mathbf{A}$ and $\mathbf{b}$ of Ψ be trained along with the parameters of $\mathcal{H}$. Although the proof (see Appendix A) seems to imply that purely translational flows ($\mathbf{A} = 0$) suffice for universality, allowing for a learnable matrix $\mathbf{A}$ seems to perform better in practice.

Affine flows may be calculated very quickly, at least for systems in low dimensions: The exponentials $\exp(\mathbf{A}t_i)$ may be calculated entirely in parallel using batching/broadcasting, in contrast to numerical solvers.

Additionally, we may enforce topological properties of the flow by appropriately projecting $\mathbf{A}$ onto a suitable matrix algebra, defining the topology of the associated Lie group through the exponential map. For example, if we desire oscillatory behavior, we may enforce the topology of the orthogonal group by making $\mathbf{A}$ the skew-symmetric component of a trainable parameter $\mathbf{M}$:

$$\mathbf{A} = (\mathbf{M} - \mathbf{M}^T)/2. \tag{16}$$

We interpret our architecture as approximating a flow by that of an *integrable* system. As a result, Neural Conjugates of affine flows should perform poorly when emulating strongly nonlinear features, such as convergence towards limit cycles and strange attractors. We show that this is indeed the case in Appendix C.

Moreover, this indicates that careful initialization of the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ are necessary to achieve convergence. We have observed that a poor initialization leads to a tug-of-war between Ψ and $\mathcal{H}$. To avoid this, we initialize $\mathcal{H}$ near the identity and Ψ as a suitable affine approximation to the system dynamics around $\mathbf{x}^0$ (see Appendix B).

$$\mathbf{A}_0\mathbf{x}^0 + \mathbf{b}_0 \approx F(\mathbf{x}^0). \tag{17}$$

4 Numerical Experiments

We will validate our models by solving forward and inverse problems in the realm of mathematical neuroscience. We aim to evaluate how well they can learn the firing/spiking dynamics of different biological neuron models, with and without data.

We evaluate three metrics. The first metric is accuracy, measured in terms of the mean-squared error between the estimate obtained by each method and the ground truth baseline, obtained using a high-accuracy numerical solver. Our second metric is total training time. Our third metric is the models' extrapolation capacity, through which we aim to establish if the model has learned to generalize the system's dynamics. To measure this, we compute the model beyond the training interval $[0, T]$, computing its accuracy relative to the ground truth over the interval $[0, 2T]$.

The baseline MLP-PINNs were built according to current best practices (Wang et al. 2023), including a Gaussian Fourier feature layer (Tancik et al. 2020) with $\sigma = 2$. We use the initial condition trick in (Biloš et al. 2021), Xavier initialization (Kumar 2017) and tanh activations.

For the Neural ODEs, we used the Pytorch implementation given by the TorchDyn library (Poli et al. 2021). This library offers fully optimized implementations for the networks and the associated numerical solvers. A second-order, semi-implicit (midpoint) solver with fixed time-step was used to prevent the slowdowns of up to one order of magnitude imposed by higher-order methods. The vector fields were represented by MLPs with tanh activations and Xavier initialization.

Finally, we evaluate two implementations of Neural Conjugate Flows: vanilla, denoted NCF, and topologically enforced, denoted NCF-T. The homeomorphisms $\mathcal{H}$ are composed of two MLP-based Coupling Layers.

Each model was trained for 2000 epochs, full-batch, and optimized with ADAM (Kingma and Ba 2014). Each experiment was run five times, after which the average and standard deviation of each metric were evaluated. They were executed on the same machine, equipped with an AMD Ryzen 9 5900HX processor, an RTX 3060 GPU and 16GB of RAM. Further specifications may be found in Appendix C.

4.1 Forward Problem: FitzHugh-Nagumo

The FitzHugh-Nagumo (FH) system (FitzHugh 1961) is a minimal description of the activation dynamics of a spiking neuron. It is a prime example of a relaxation oscillator and exhibits rich dynamics, including limit cycles, excitation

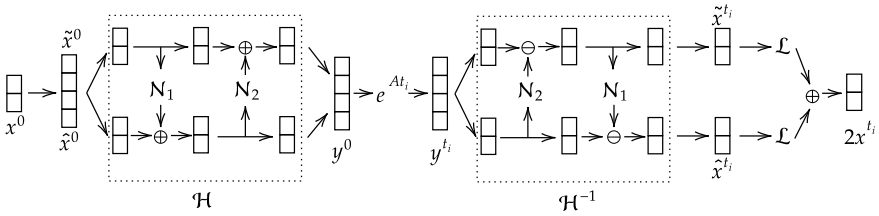


Figure 7: Our augmented Affine Neural Conjugate, in full. The following operations are applied in order, from left to right: input duplication, the homeomorphism $\mathcal{H}$ (comprised of two coupling layers), the affine flow Ψ and the inverse homeomorphism $\mathcal{H}^{-1}$. The outputs $\tilde{\mathbf{x}}^t, \hat{\mathbf{x}}^t$ are each trained according to the Physics-informed loss $\mathcal{L}$ then finally averaged.

blocks and anodal breaks. Additionally, it becomes stiff for small ϵ. The equations read:

$$\begin{aligned} \frac{dV_m}{dt} &= V_m - \frac{V_m^3}{3} - r + I \\ \frac{dr}{dt} &= \epsilon(V_m + a - br) \end{aligned} \tag{18}$$

where V_m is the neuron membrane potential, r is a recovery variable associated with sion channels, I is an external current, and ϵ, a, and b are parameters.

We compare three models: MLP, NCF and NCF-T.As in this case we have perfect knowledge of the differential equation, making a comparison to Neural ODEs would be unfair: one might as well use a traditional numerical solver in their place.

For this experiment, we compare how well MLP-PINNs and NCFs can solve the FH system with *no data*, using only the associated Physics-informed loss. For training, we subdivided the time-interval uniformly in $N = 100$ samples t_i. A single Physics-informed loss was used, implemented as an L_2 penalty for the ODE residue at t_i:

$$\mathcal{L}_{\text{PINN}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left|\left| \frac{d}{dt}\mathcal{N}_\theta(\mathbf{x}^0, t_i) - F(\mathcal{N}_\theta(\mathbf{x}^0, t_i)) \right|\right|_2^2, \tag{19}$$

where $\mathcal{N}$ stands for the model and F stands for the right-hand side of (18) and (19). For evaluation, we measured the the accuracy and generalization losses as:

$$\mathcal{L}_{\text{acc,extrap}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left|\left|\mathcal{N}_\theta(\mathbf{x}^0, t_i) - \mathbf{x}^{t_i}\right|\right|_2^2, \tag{20}$$

where for measuring accuracy we take $0 \leq t_i \leq 10$, while for generalization we take $0 \leq t_i \leq 20$. The results are listed in Table 1. Experimental details may be found in Appendix C.

4.2 Inverse Problem: Hodgkin-Huxley

The Hodgkin Huxley model (Hodgkin and Huxley 1952) is the canonical neuron model used in mathematical biology, based on the dynamics of the giant squid axon. The model is a fourth-order, nonlinear differential equation that displays remarkably intricate phenomena, including chaos (Guckenheimer and Oliva 2002). The equation for the neuronal voltage is given by:

$$\begin{aligned} C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} =& - I + \bar{g}_{\text{K}} n^4 (V_m - V_K) \\ &+ \bar{g}_{\text{Na}} m^3 h (V_m - V_{Na}) + \bar{g}_l (V_m - V_l) \end{aligned} \tag{21}$$

where Vm is the membrane potential, C_m is the membrane capacitance, I is the stimulus current, V_K, V_{Na}, V_l are the base voltages for sodium, potassium, and leak currents, $\bar{g}_{\text{Na}}, \bar{g}_{\text{K}}, \bar{g}_{\text{L}}$ are the respective maximal conductances and $E_{\text{Na}}, E_{\text{K}}, E_{\text{L}}$ are the respective reversal potentials.

m, h, n are gating variables for sodium activation, sodium inactivation, and potassium activation, which follow their own set of nonlinear equations, determined experimentally.

For this experiment, we measure the networks' capacity to use both data *and* physics to reproduce the dynamics of the neuron. We attempt to emulate the experimental setup for finding surrogate models for n, m, h from measurements (Johnston and Wu 1994) using a simulation as ground-truth.

We compare four models: MLP, NCF, NCF-T and NODE. We sub-divide the time-interval $[0, 14]$ into $N = 100$ samples and train full-batch, now using two losses.

First, a Physics-informed loss associated *only* to the principled model for V_m given in eq. (22):

$$\mathcal{L}_{\text{PINN}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left|\left| \frac{d}{dt}\mathcal{N}_\theta(\mathbf{x}^0, t_i)[0] - F(\mathcal{N}_\theta(\mathbf{x}^0, t_i))[0] \right|\right|_2^2 \tag{22}$$

Second, a data-driven loss based on noisy samples $\tilde{\mathcal{X}}$ of the remaining three variables n, m, h:

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left|\left|\mathcal{N}_\theta(\mathbf{x}^0, t_i)[1:3] - \tilde{\mathbf{x}}^{t_i}[1:3]\right|\right|_2^2, \tag{23}$$

The evaluation metrics are the same as for Experiment 1. Our objective is to observe if the models can combine heterogeneous information into a solid model for the spiking dynamics. The result are in Table 2. Details may be found in Appendix C.

4.3 Analysis and Additional Results

We observe that in general, neural conjugates are slower than classical feed-forward architectures roughly by a factor of two, while being faster than Neural ODEs roughly by a factor of five.

Our belief is that the speed deficit compared to MLPs can be mostly attributed to the fact that the input must flow

Model	Layers	Params	$\mathcal{L}_{\text{acc}}$	$\mathcal{L}_{\text{extrap}}$	Time(s)
MLP	$3 \to 32 \to 32 \to 32 \to 2$	2.3K	$4.9{\times}10^{-4} \pm 2.7{\times}10^{-5}$	$2.8{\times}10^{1} \pm 5.4{\times}10^{0}$	35.7 ± 0.9
NCF	$2 \times (2 \to 32 \to 32 \to 2)$	2.5K	$6.7{\times}10^{-2} \pm 1.0{\times}10^{-2}$	$2.5{\times}10^{-1} \pm 3.0{\times}10^{-2}$	74.7 ± 0.5
NCF-T	$2 \times (2 \to 32 \to 32 \to 2)$	2.5K	$3.2{\times}10^{-4} \pm 2.6{\times}10^{-5}$	$1.9{\times}10^{-3} \pm 4.6{\times}10^{-4}$	75.8 ± 2.2

Table 1: Experiment 1: FitzHugh-Nagumo model

Model	Layers	Params	$\mathcal{L}_{\text{acc}}$	$\mathcal{L}_{\text{extrap}}$	Time(s)
MLP	$5 \to 128 \to 128 \to 4$	17.0K	$3.2{\times}10^{-5} \pm 3.5{\times}10^{-5}$	$7.7{\times}10^{0} \pm 1.7{\times}10^{0}$	26.5 ± 0.8
NCF	$2 \times (4 \to 90 \to 90 \to 4)$	18.1K	$1.8{\times}10^{-2} \pm 4.4{\times}10^{-3}$	$3.2{\times}10^{-2} \pm 3.2{\times}10^{-1}$	44.4 ± 1.9
NCF-T	$2 \times (4 \to 90 \to 90 \to 4)$	18.1K	$1.6{\times}10^{-3} \pm 9.7{\times}10^{-4}$	$1.6{\times}10^{-3} \pm 9.8{\times}10^{-4}$	45.6 ± 2.4
NODE	$4 \to 128 \to 128 \to 4$	17.7K	$3.5{\times}10^{-2} \pm 9.0{\times}10^{-3}$	$3.8{\times}10^{-2} \pm 9.1{\times}10^{-3}$	241.1 ± 11.2

Table 2: Experiment 2: Hodgkin-Huxley model

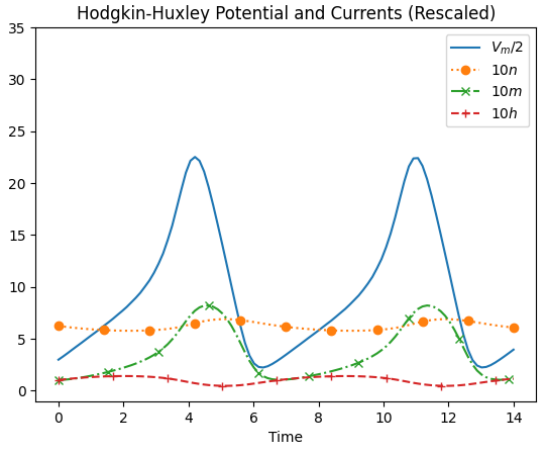


Figure 8: Reference values for potentials and currents.

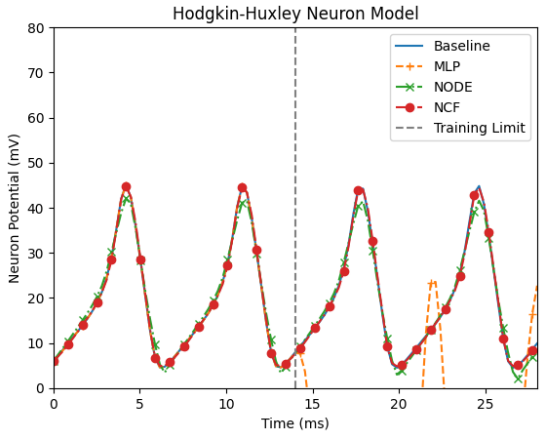


Figure 9: Reconstructed and extrapolated potentials.

through $\mathcal{H}$ twice (one for $\mathcal{H}$ and one for the inverse $\mathcal{H}^{-1}$). On the other hand, the speed advantage in comparison to Neural ODEs comes from the fact that the affine flows Ψ may be calculated in an entirely parallel manner, as opposed to the sequential nature of the solvers built into Neural ODEs.

We observe that although MLP-PINNs can interpolate solutions quickly and accurately, they do not display any generalization power. On the other hand, Neural ODEs and Neural Conjugates can be said to be truly learning the latent neuron dynamics, as illustrated by their extrapolation capabilities. We have also observed that NCFs both interpolate and extrapolate solutions better than Neural ODEs, at significantly lower computational cost.

The results show that NCFs perform particularly well when equipped with a well-chosen topology. There are situations for which we expect NCFs to perform worse than the other models, however. This is most evident when dealing with strongly nonlinear phenomena, such as chaotic behavior or complex limit cycles. We demonstrate this in Appendix C.

5 Final Remarks

We have introduced Neural Conjugate Flows, a novel Physics-Informed architecture with the structure of flows, achieved via the composition of invertible neural networks and affine systems. Through the use of topological conjugation, these networks have the properties of continuous groups by construction, leading to automatic compliance with initial conditions and enhanced causality. It is also topology-informed.

Numerical experiments demonstrate that, despite the reduced representation power of invertible networks, our architecture consistently outperforms both Multi-layer Perceptrons and Neural Ordinary Differential equations at extrapolating the dynamics of latent dynamical systems. Moreover, they are up to five times faster than Neural ODEs, while being about twice as slow as conventional PINNs.

However, NCFs as implemented here are strongly limited by the representation capacity of the central flow Ψ. If it is chosen to be just affine, Ψ deals poorly with systems with strongly nonlinear behavior.

A Universal Approximation

To prove that NCFs are Universal Approximators for solutions of ODEs, we need only find a class of flows Ψ that are conjugated to any ODE and a class of invertible networks that approximate any homeomorphism H.

For H, the class of coupling layers suffices, as discussed. As for Ψ, we may choose the class of affine flows, as we have established. We'll prove this next.

Proof of Theorem 1. We take the minimal augmentation in $m = 1$ additional dimensions, with $a \in \mathbb{R}$ and $G(a, \mathbf{x}) := 1$. this leads to the augmented system:

$$\frac{d}{dt}\hat{\mathbf{x}} = \frac{d}{dt}\begin{bmatrix}\mathbf{x}\\ a\end{bmatrix} = \begin{bmatrix}F(\mathbf{x})\\ 1\end{bmatrix} = \hat{F}(\hat{\mathbf{x}}) \tag{24}$$

Obviously, if F is Lipschitz continuous, so is $\hat{F}$. Furthermore, the variables a and $\mathbf{x}$ are independent, which implies that the flow $\hat{\Phi} : \mathbb{R} \times \mathbb{R}^{n+1} \mapsto \mathbb{R}^{n+1}$, associated with $\hat{F}$, has a simple expression in terms of the flow Φ associated with F:

$$\hat{\Phi}^t\left(\begin{bmatrix}\mathbf{x}\\ a\end{bmatrix}\right) = \begin{bmatrix}\Phi^t x\\ a + t\end{bmatrix} \tag{25}$$

We will prove that the flow $\hat{\Phi}$ is conjugated to the affine system with $\hat{A} = 0$ (the all-zero $n + 1 \times n + 1$ matrix) and set $b = [0_n, 1]^T$, (the vector with n zeros and a one in its last position). The flow associated with this affine system is given by

$$\hat{\Psi}^t\left(\begin{bmatrix}\mathbf{x}\\ a\end{bmatrix}\right) = \left(\begin{bmatrix}\mathbf{x}\\ a + t\end{bmatrix}\right). \tag{26}$$

We show the two flows are conjugated by providing an explicit formula for the conjugation $\mathcal{H}$ and its inverse:

$$\mathcal{H}\left(\begin{bmatrix}\mathbf{x}\\ a\end{bmatrix}\right) = \begin{bmatrix}\Phi^{-a}\mathbf{x}\\ a\end{bmatrix} \qquad \mathcal{H}^{-1}\left(\begin{bmatrix}\mathbf{x}\\ a\end{bmatrix}\right) = \begin{bmatrix}\Phi^{a}\mathbf{x}\\ a\end{bmatrix}. \tag{27}$$

We note that H and its inverse are indeed well-defined in $\mathbb{R}^n$. Indeed, Picard's Existence Theorem establishes that since $\hat{F}$ is globally Lipschitz, we have that the flow Φ^a and its time-reversal Φ^{-a} are well-defined everywhere, which implies that $\mathcal{H}$ is well defined and a homeomorphism. Finally, the proof follows by algebraic manipulation:

$$\begin{aligned}\mathcal{H}^{-1} \circ \hat{\Psi}^t \circ \mathcal{H}\left(\begin{bmatrix}\mathbf{x}\\ a\end{bmatrix}\right) &= \mathcal{H}^{-1}\left(\begin{bmatrix}\Phi^{-a}\mathbf{x}\\ a + t\end{bmatrix}\right)\\ &= \begin{bmatrix}\Phi^{t+a}\Phi^{-a}\mathbf{x}\\ a + t\end{bmatrix} = \hat{\Phi}^t\left(\begin{bmatrix}\mathbf{x}\\ a\end{bmatrix}\right).\end{aligned} \tag{28}$$

□

B Implementation Details

B.1 Evaluating Affine Flows

A well-known trick to evaluate an affine flow without calculating integrals may be given as follows. We again augment the system, adding a dummy state with no dynamics so that the resulting purely linear system behaves as an affine system:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}\\ a\end{bmatrix} = \begin{bmatrix}\mathbf{A} & \mathbf{b}\\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ a\end{bmatrix} \tag{29}$$

If we now set $a^0 = 1$, we get that the dynamics of $\mathbf{x}$ under this flow is now exactly that of an affine flow:

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x} + \mathbf{b}.$$

B.2 Initialization

Affine NCFs fundamentally try to approximate the target dynamics by that of a higher-dimensional affine system. Given an initial position $\mathbf{x}^0$, a natural first attempt would be to try and fit a linear system passing by $\mathbf{x}^0$ such that its dynamics are as close as possible to those of the target system. We do this in three steps:

1. Set $\mathcal{H}$ to be close to the identity: This can be done by multiplying the output of the neural networks used in the coupling layer ensemble by a small value.

2. Map equilibria to the origin: Locate an equilibrium close to $\mathbf{x}^0$, say $\bar{\mathbf{x}}$, and change variables $\mathbf{x} \mapsto \mathbf{x} - \bar{\mathbf{x}}$. If there are none, we just assume it to be located at $\mathbf{x} = 0$.

3. Define $\mathbf{A}$: Compute the matrix $\mathbf{A}$ such that $\mathbf{A}\mathbf{x}$ best approximates the dynamics at $\mathbf{x}^0$. We claim $\mathbf{A}$ is the best possible linear approximation at $\mathbf{x}^0$ if $\mathbf{A}\mathbf{x}^0$ exactly matches the first derivative and achieves the minimum possible error for the second. This may be expressed as:

$$\min_A ||A - \mathbf{J}_0|| \text{ subject to } A\mathbf{x}^0 = F(\mathbf{x}^0) \tag{30}$$

where $\mathbf{J}_0$ stands for the jacobian of f at $\mathbf{x}^0$. There is a closed form solution for this problem:

$$\mathbf{A} = \mathbf{J}_0 + \frac{1}{\|\mathbf{x}^0\|^2}\left(F(\mathbf{x}^0) - \mathbf{J}_0\mathbf{x}^0\right)(\mathbf{x}^0)^T.$$

This is, in some sense, the best possible initialization: If f is linear, this initialization implies that our network starts out as the analytical solution to the problem. Still, it is possible that $\mathbf{J}_0$ contains eigenvalues with positive real part: in this case, this initialization will lead to a solution that blows up exponentially in time, which is rarely a good first guess.

In these cases, we adapt the approach above. We decompose the matrix $\mathbf{A}$ computed above into its symmetric and skew-symmetric components. We then keep only the skew-symmetric component, which contains only eigenvalues with zero real part: $\tilde{\mathbf{A}} = (\mathbf{A} - \mathbf{A}^T)/2$. We then supplement the remaining terms with a constant vector field $\mathbf{b}$ such that:

$$F(\mathbf{x}^0) = \tilde{\mathbf{A}}\mathbf{x}^0 + \mathbf{b}. \tag{31}$$

C Experiments

Due to space restrictions, we include here the remaining details regarding the experiments in the main paper, as well as additional experiments.

We found that using automatic differentiation lead to conflict with the numerical solvers underlying the Neural ODEs in Experiment 2. For this reason, all time derivatives were calculated using a second-order, centered finite-difference approximation with small (10^{-3}) step size.

Likewise, we found that using the adjoint method for gradient calculations lead to substantial slowdowns. For this reason, gradients were in fact calculated using usual back-propagation (discretize-then-optimize), as per the specifications of TorchDyn.

C.1 Experimental Details - FitzHugh-Nagumo

For simplicity, the parameters were chosen as $a = b = I = 0$ and $\epsilon = 1$. This leads to no significant changes in the

Model	Params	$\mathcal{L}_{\text{acc}}$	$\mathcal{L}_{\text{extrap}}$	Time(s)
MLP	2.3K	$1.1\times10^{0} \pm 1.1\times10^{-1}$	$3.3\times10^{1} \pm 7.6\times10^{0}$	34.1 ± 0.6
NCF	2.5K	$9.2\times10^{-2} \pm 2.3\times10^{-2}$	$3.6\times10^{-1} \pm 4.8\times10^{-1}$	81.8 ± 2.7
NCF-T	2.5K	$2.6\times10^{-2} \pm 4.0\times10^{-2}$	$1.1\times10^{-1} \pm 1.8\times10^{-1}$	82.9 ± 1.5

Table A. Experiment 3: Lotka-Volterra model

Model	Params	$\mathcal{L}_{\text{acc}}$	$\mathcal{L}_{\text{extrap}}$	Time(s)
MLP	2.3K	$4.4\times10^{-5} \pm 2.1\times10^{-5}$	$5.1\times10^{0} \pm 1.8\times10^{0}$	33.1 ± 0.6
NCF	2.5K	$1.4\times10^{0} \pm 8.8\times10^{-3}$	$1.7\times10^{0} \pm 1.9\times10^{-2}$	80.7 ± 1.6
NCF-T	2.5K	$1.4\times10^{0} \pm 8.8\times10^{-3}$	$2.3\times10^{0} \pm 8.9\times10^{-2}$	80.2 ± 1.2

Table B. Experiment 4: FitzHugh-Nagumo model

system's qualitative behavior. For the initial conditions, we chose $\mathbf{x}^0 = (2, -2/3)$ to begin near the limit cycle. The optimizer was set up with Learning rate $\alpha = 1\times10^{-3}$ and decay parameters $\beta = (0.9, 0.99)$.

As discussed, the data for the second experiment is synthetic, generated from numerical integration of Hodgkin and Huxley's original model. Their model for the gating variables n, m, h was as follows:

$$\frac{di}{dt} = \alpha_i(V_m)(1 - i) - \beta_i(V_m)i, \tag{32}$$

where i stands for either n, m or h. Likewise, the nonlinear functions α and β are the same as in the original paper:

$$\alpha_n(V_m) = \frac{0.01(10 - V)}{\exp\left(\frac{10-V}{10}\right) - 1}, \alpha_m(V_m) = \frac{0.1(25 - V)}{\exp\left(\frac{25-V}{10}\right) - 1},$$

$$\alpha_h(V_m) = 0.07\exp\left(-\frac{V}{20}\right), \beta_n(V_m) = 0.125\exp\left(-\frac{V}{80}\right),$$

$$\beta_m(V_m) = 4\exp\left(-\frac{V}{18}\right), \beta_h(V_m) = \frac{1}{\exp\left(\frac{30-V}{10}\right) + 1}$$

For the real system, the magnitude of these variables varies widely, leading to some currents being much larger than others. To minimize this multi-scale effect, and to avoid saturating input neurons, we rescale V, n, m, h by factors of $0.1, 10, 10, 10$ respectively. We again initialize the system close to the limit cycle.

The optimizer was set up with learning rate $\alpha = 2.5\times10^{-3}$ and decay parameters $\beta = (0.9, 0.95)$.

C.2 Additional Experiment: Causality

We illustrate the difficulties PINNs have with causality and spurious convergence to trivial solutions. The Lotka-Volterra system, built to model ecological competition, has also found applications as a model for spiking neurons (Noonburg 1989):

$$\frac{dx}{dt} = \alpha x - \beta xy, \quad \frac{dy}{dt} = -\gamma y + \delta xy, \tag{33}$$

This simple oscillator nevertheless poses a remarkable challenge for PINNs, due to the fact that its orbits veer close to the equilibrium lines $x = 0$ and $y = 0$. As a result, regular PINNs often find it easier to gravitate towards these equilibria. For this experiment, we measure how often each architecture converges to an incorrect solution, using the same setup as experiment 1. We take $\alpha = \beta = \gamma = \delta = 1$ for simplicity.

The results, given in Table A, clearly indicate that regular PINNs were entirely unable to approximate the solution to this problem, while NCFs converged to the correct, physically meaningful solutions (see Figure 1).

C.3 Additional Experiment: Nonlinearity

To illustrate the difficulties of Affine Neural Conjugates with strongly nonlinear phenomena, we run a second instance of Experiment 1, now with an initial condition closer to the origin. The orbit starts as an oscillator increasing in amplitude, until it converges to a limit cycle (see Figure A).

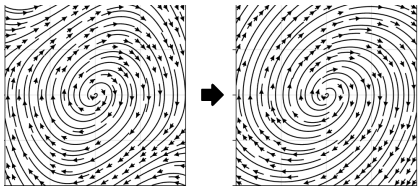


Figure A: The FitzHugh-Nagumo model and its linearization.

This poses a significant challenge to affine NCFs because the system behaves qualitatively similar to two distinct linear systems at different points in time: At $t = 0$, it should behave as a hyperbolic linear system, spiralling outwards. Later, when it converges to the limit cycle, the system behaves as an harmonic oscillator, with purely imaginary eigenvalues.

Because the flow Ψ is fixed in time, it cannot adapt to these different realities. As a result, the network can't choose between the two distinct topologies and fails to converge.

We emphasize that this is due to the topological regularization imposed by affine flows, which may be desirable in some contexts. More representative Ψ could lessen this effect.

Acknowledgements

This work was supported by Petrobras. We thank profs. Sergei Tikhomirov and Marcelo Viana for their suggestions.

References

Biloš, M.; Sommer, J.; Rangapuram, S. S.; Januschowski, T.; and Günnemann, S. 2021. Neural flows: Efficient alternative to neural ODEs. *Advances in neural information processing systems*, 34: 21325–21337.

Bramburger, J. J.; Brunton, S. L.; and Kutz, J. N. 2021. Deep learning of conjugate mappings. *Physica D: Nonlinear Phenomena*, 427: 133008.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 6572–6583. Red Hook, NY, USA: Curran Associates Inc.

Cranmer, M.; Greydanus, S.; Hoyer, S.; Battaglia, P.; Spergel, D.; and Ho, S. 2020. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*.

Cuomo, S.; Di Cola, V. S.; Giampaolo, F.; Rozza, G.; Raissi, M.; and Piccialli, F. 2022. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3): 88.

De Ryck, T.; Mishra, S.; and Molinaro, R. 2022. Weak physics informed neural networks for approximating entropy solutions of hyperbolic conservation laws. In *Seminar für Angewandte Mathematik, Eidgenössische Technische Hochschule, Zürich, Switzerland, Rep*, volume 35, 2022.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. In *International Conference on Learning Representations*.

Draxler, F.; Wahl, S.; Schnörr, C.; and Köthe, U. 2024. On the universality of coupling-based normalizing flows. *arXiv preprint arXiv:2402.06578*.

Dupont, E.; Doucet, A.; and Teh, Y. W. 2019. Augmented Neural ODEs. arXiv:1904.01681.

Finlay, C.; Jacobsen, J.-H.; Nurbekyan, L.; and Oberman, A. 2020. How to Train Your Neural ODE: the World of Jacobian and Kinetic Regularization. In *Proceedings of the 37th International Conference on Machine Learning*, 3154–3164. PMLR.

FitzHugh, R. 1961. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Biophysical Journal*, 1(6): 445–466.

Guckenheimer, J.; and Oliva, R. A. 2002. Chaos in the Hodgkin–Huxley model. *SIAM Journal on Applied Dynamical Systems*, 1(1): 105–114.

Hasan, A.; Pereira, J. M.; Ravier, R.; Farsiu, S.; and Tarokh, V. 2020. Learning Partial Differential Equations From Data Using Neural Networks. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3962–3966.

Hodgkin, A. L.; and Huxley, A. F. 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117: 500–544.

Humphreys, J. E. 1972. *Introduction to Lie Algebras and Representation Theory*. Springer New York. ISBN 9781461263982.

Johnston, D.; and Wu, S. M.-S. 1994. *Foundations of cellular neurophysiology*. MIT press.

Kidger, P. 2022. On Neural Differential Equations. arXiv:2202.02435.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P.; and Welling, M. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114.

Kumar, S. K. 2017. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*.

Lai, Z.; Mylonas, C.; Nagarajaiah, S.; and Chatzi, E. 2021. Structural identification with physics-informed neural ordinary differential equations. *Journal of Sound and Vibration*, 508: 116196.

Mao, Z.; Jagtap, A. D.; and Karniadakis, G. E. 2020. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360: 112789.

Mattheakis, M.; Sondak, D.; Dogra, A. S.; and Protopapas, P. 2022. Hamiltonian neural networks for solving equations of motion. *Physical Review E*, 105(6): 065305.

O'Leary, J.; Paulson, J. A.; and Mesbah, A. 2022. Stochastic physics-informed neural ordinary differential equations. *Journal of Computational Physics*, 468: 111466.

Papamakarios, G.; Nalisnick, E.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2021. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57): 1–64.

Patel, R. G.; Manickam, I.; Trask, N. A.; Wood, M. A.; Lee, M.; Tomas, I.; and Cyr, E. C. 2022. Thermodynamically consistent physics-informed neural networks for hyperbolic systems. *Journal of Computational Physics*, 449: 110754.

Poli, M.; Massaroli, S.; Yamashita, A.; Asama, H.; Park, J.; and Ermon, S. 2021. TorchDyn: implicit models and neural numerical methods in PyTorch. In *Neural Information Processing Systems, Workshop on Physical Reasoning and Inductive Biases for the Real World*, volume 2.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. E. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378: 686–707.

Rao, C.; Sun, H.; and Liu, Y. 2021. Physics-informed deep learning for computational elastodynamics without labeled data. *Journal of Engineering Mechanics*, 147(8): 04021043.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference*, 234–241. Springer.

Tancik, M.; Srinivasan, P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.; and Ng, R. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33: 7537–7547.

Teshima, T.; Ishikawa, I.; Tojo, K.; Oono, K.; Ikeda, M.; and Sugiyama, M. 2020. Coupling-based invertible neural networks are universal diffeomorphism approximators. *Advances in Neural Information Processing Systems*, 33: 3362–3373.

Viana, M.; and Espinar, J. M. 2021. *Differential equations: a dynamical systems approach to theory and practice*, volume 212. American Mathematical Society.

Wang, S.; Sankaran, S.; and Perdikaris, P. 2022. Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*.

Wang, S.; Sankaran, S.; Wang, H.; and Perdikaris, P. 2023. An expert's guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*.

Yu, J.; Lu, L.; Meng, X.; and Karniadakis, G. E. 2022. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393: 114823.

Zhang, Z.-Y.; Zhang, H.; Zhang, L.-S.; and Guo, L.-L. 2023. Enforcing continuous symmetries in physics-informed neural network for solving forward and inverse problems of partial differential equations. *Journal of Computational Physics*, 492: 112415.

Zhi, W.; Lai, T.; Ott, L.; Bonilla, E. V.; and Ramos, F. 2022. Learning Efficient and Robust Ordinary Differential Equations via Invertible Neural Networks. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 27060–27074. PMLR.