2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Building Hybrid B-Spline And Neural Network Operators

Raffaele Romagnoli, Jasmine Ratchford, and Mark H. Klein

Abstract— **Control systems are critical in ensuring the safety of cyber-physical systems (CPS) across domains like airplanes and missiles. Safeguarding CPS necessitates runtime methodologies that continuously monitor safety-critical conditions and respond in a verifiably safe manner. Many real-time safety approaches require predicting the future behavior of systems. However, achieving this requires accurate models that can operate in real time. Inspired by DeepONets, we propose a novel approach that combines B-splines' inductive bias with data-driven neural networks (NNs). Our hybrid B-spline neural operator serves as a universal approximator, validated on a 6-DOF quadrotor.**

I. Introduction

Control systems are crucial for ensuring the safety of cyber-physical systems (CPS) like airplanes and missiles [1]. Because dynamic interactions with the environment make it impossible to foresee all hazards, runtime approaches[2] are required to continuously monitor safety conditions and react safely. Many runtime approaches require accurate models of system behavior in real time. This is exceedingly challenging for complex systems. The notable success of machine learning (ML) techniques has drawn researchers' attention towards scientific ML (SciML) methods, which can learn from governing equations and data [3][4] and provide real-time solutions [5][6].

We are inspired by DeepONets [4], a NN architecture that can be used to learn generalized non-linear operators. For an autonomous system a NN could learn the operator that maps initial conditions (IC) to unique trajectory that describes the autonomous systems' path to a desired end state.

DeepONets learn basis functions for approximated solutions. While advantageous for complex systems, this approach introduces uncertainty due to stochastic models in both the trunk net (where basis functions are learned) and the branch net (where coefficients are learned). In contrast, established methods like splines provide a well-researched, continuous basis with an inductive bias. Our study combines neural operator theory with B-splines as basis functions [7].

The key parameters that need to be learned when using B-splines to approximate continuous functions are control points. We train a NN to learn the mapping from a set of differential equations' initial conditions (ICs) to the control points of a B-spline approximation. The general approach of using B-spline functions to approximate the solutions of PDEs in the field of Finite Element Methods (FEMs) is referred to isogeometric analysis (IGA). A similar solution has been proposed in [8] where a NN generates the B-spline coefficients for a parameterized family of vector fields to reduce computational costs.

This paper's contribution is an analysis of this approach within the framework of deep neural operators, demonstrating its property as a universal approximator, and providing bounds on the approximation error. These results are developed for a general nonlinear autonomous system, and the approach is tested on a controlled 6-degree-of-freedom (DOF) quadrotor with a state space of 12 dimensions. Additionally, a comparison between different network architectures, namely fully connected neural networks (FCNN) and recurrent neural networks (RNN), is provided. We also highlight the benefits of neural operators in reducing computational time compared to classical methods for solving ODEs.

Proving uncertainty bounds is a challenge ML-based approaches. The method proposed opens new opportunities to explore uncertainty quantification, which is fundamental in all aspects of CPSs, including safety. We distinguish our research focus from research that focuses on incorporating physics [6] for improved predictive performance and other physics-based guarantees. The choice of B-splines is motivated by several factors: B-splines are fully defined by a finite number of control points, which can be used for safety inference due to the convex-hull property (See section II.D) [9]. In addition, splines are easy to extend to higher dimensions, enabling a simple description of a high-order smooth function to be derived from the NN[1]. Finally, the relationship between B-splines and Gaussian Process will facilitate inclusion of errors and uncertainty associated with the underlying method of interpolation [10] providing an avenue for future work to explore and refine these connections.

The paper is organized as follows: Section II presents the preliminaries needed to formulate the problem statement (Section II.E). Section III presents the main theoretical results. Section IV details the experimental results, and Section V offers the conclusions..

II. Preliminaries

This section is divided in the following parts: nonlinear systems [11], neural networks [12], the universal approximation theorem for nonlinear operators [13], B-splines functions [9] and finally the problem statement. For further details on the discussed topics, please refer to the aforementioned references.

R. Romagnoli is with the School of Science and Engineering, Duquesne University, Pittsburgh, PA 15213, USA: romagnolir@duq.edu

J. Ratchford and M. H. Klein are with the Software Engineering Institute, Carnegie Mellon university, Pittsburgh, PA 15235, USA: jratchford@sei.cmu.edu mk@sei.cmu.edu

[1]This may enable us to more easily evaluate the dynamics of the resulting solutions and compare to physics-based constraints

A. Nonlinear Systems

Consider a nonlinear system expressed in the state-space form:

$$\dot{x} = f(t, x) \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector with n components, and the function $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz on a ball around the IC $x_0 \in \mathcal{B}_r = \{x \in \mathbb{R}^n : \|x - x_0\| \leq r\}$ and for the time interval $[t_0, t_1]$: $\|f(x) - f(y)\| \leq L\|x - y\|$ for all $x, y \in \mathcal{B}_r$. $\|\cdot\|$ is the Euclidean norm. From the local Cauchy's theorem, there exists a unique solution on the time interval $[t_0, t_0 + \delta]$ with a positive $\delta \in \mathbb{R}$. According to Peano's theorem, the solution can be written as:

$$x(t) = x_0 + \int_0^t f(s, x(s))ds \tag{2}$$

which is a continuous function $x : \mathbb{R} \to \mathbb{R}^n$. Let us consider the space of continuous functions over the time interval $[a, b]$, defined as $\mathcal{X} = C([a, b]; \mathbb{R}^n)$, which is a Banach space with norm $\|x\|_C = \max_{t\in[0,\delta]} \|x(t)\|$. Considering $\mathcal{S} = \{x \in \mathcal{X} : \|x - x_0\|_C \leq r\}$ it is possible to show that the right-hand side of (2) is a mapping $\mathcal{P} : \mathcal{S} \to \mathcal{S}$, and moreover, this mapping is a contraction:

$$x(t) = \mathcal{P}(x)(t) \tag{3}$$

where the function $x(t)$ is a fixed point.

We assume that the solution can be extended over the time interval $[0, T]$ and the Lipschitz condition holds for a compact set $W \subset \mathbb{R}^n$. If the solution belongs to W for any $t > 0$ then T can be arbitrarily large. Equation (1) is the general representation of a non-autonomous system, where the explicit dependence on time t can be due to an external input $u \in \mathbb{R}^p$:

$$\dot{x} = f(x, u)$$

If the input signal is a state feedback control law of the form $u = \gamma(x)$, where $\gamma : \mathbb{R}^n \to \mathbb{R}^p$, the resulting closed-loop system can be represented as an autonomous system of the form

$$\dot{x} = f(x) \tag{4}$$

Without loss of generality, we assume that $x = 0$ is an equilibrium point (i.e., $f(0) = 0$) for (4), hence the origin is contained in W.

B. Neural Networks

The fundamental element of a neural network is the standard neuron, which takes n inputs $\{u_1, u_2, \ldots, u_n\}$, where $u_i \in \mathbb{R}$, and transforms them into a scalar value $y \in \mathbb{R}$. The activation signal for the i-th neuron is computed as:

$$s = u_1\theta_{i,1} + u_2\theta_{i,2} + \cdots + u_n\theta_{i,n} + \theta_{i,0} \tag{5}$$

where $\theta_{i,j} \in \mathbb{R}$ with $j = 1, \ldots, n$ are the weights that need to be estimated, and $\theta_{i,0}$ is called the bias. The activation signal (5) serves as the input to the activation function $\sigma : \mathbb{R} \to \mathbb{R}$, which is typically nonlinear (e.g., sigmoidal, ReLU, etc.). Assuming we have a single-layer network with N parallel neurons that use the same activation function, the output can be expressed as:

$$\begin{aligned} y &= \sum_{i=1}^{N} w_i \sigma \left(\sum_{j=1}^{n} \theta_{i,j} u_j + \theta_{i,0} \right) \\ &= \sum_{i=1}^{N} w_i \sigma \left(\theta_i^T u + \theta_{i,0} \right) \end{aligned} \tag{6}$$

where $\theta_i = [\theta_{i,1}, \ldots, \theta_{i,n}]^T$, $u = [u_1, \ldots, u_n]^T$. If the output layer has a dimension greater than one, (6) describes the m-th component of the output vector:

$$y_m = \sum_{i=1}^{N} w_{i,m} \sigma \left({\theta_{i,m}}^T u + \theta_{i,0}^m \right) \tag{7}$$

In the case of multi-layer networks known as feedforward neural networks (FNNs), the output of one layer becomes the input of the next one:

$$y_m^{[l+1]} = \sum_{i=1}^{N} w_{i,m} \sigma \left({\theta_{i,m}^{[l+1]}}^T y^{[l]} + \theta_{i,0}^{m^{[l+1]}} \right) \tag{8}$$

where $[l + 1]$ indicates the $(l + 1)$-th layer, and $y^{[l]}$ is the corresponding input vector, which is the output of the previous layer. Equations (7) and (8) illustrate how a single-layer network with scalar output can be extended to handle vector outputs and multi-layer FNNs. Consequently, the theory of universal approximation of NNs primarily focuses on the case described in (6). In the following section, we provide basic definitions and results on the universal approximation of nonlinear operators with NNs.

C. The Universal Approximation Theorem for Nonlinear Operators

By considering a specific class of activation functions, Tauber-Wiener (TW) functions, we present three main theorems from [13]. Theorems 1 and 2 deal with the approximation of functions and functionals with NNs, respectively. Finally, Theorem 3 combines the two previous results to approximate nonlinear continuous operators.

Definition 1: A function $\sigma : \mathbb{R} \to \mathbb{R}$ is called a Tauber-Wiener (TW) function if all linear combinations $\sum_{i=1}^{N} c_i \sigma(\lambda_i z + \theta_i)$, $\lambda_i, \theta_i, c_i \in \mathbb{R}$, $i = 1, 2, ..., N$ are dense in every $C([a, b]; \mathbb{R})$.

Let $g : K \to \mathbb{R}$, where $K \subset \mathbb{R}$ is a compact set, $\mathcal{U} \subset C(K; \mathbb{R})$ is a compact set, and $\sigma \in (TW)$ is an activation function.

Theorem 1: For any $\epsilon > 0$, there exists a positive integer N, $\theta_0^i \in \mathbb{R}$, $\theta_i \in \mathbb{R}^n$, $i = 1, ..., N$ independent of $g \in C(K; \mathbb{R})$, and constants $c_i(g)$, $i = 1, ..., N$ depending on g such that

$$\left| g(z) - \sum_{i=1}^{N} c_i(g) \sigma(\theta_i^T z + \theta_0^i) \right| < \epsilon \tag{9}$$

holds for all $z \in K$ and $g \in \mathcal{U}$. Moreover, each $c_i(g)$ is a linear continuous functional defined on $\mathcal{U}$.

The last statement is due to the fact that in the proof, $c_j(g)$ represents the Fourier coefficients of g. If $g : \mathbb{R} \to \mathbb{R}$ and $K = [a, b]$, it can be associated with a specific time interval. Note that $\mathbb{R}^n$ with $n \geq 1 \in \mathbb{N}$ is a Banach space.

Now, let $\mathcal{X}$ be a Banach space, $\mathcal{K} \subseteq \mathcal{X}$, $\mathcal{V}$ a compact set in $C(\mathcal{K}; \mathbb{R})$, and g a continuous functional defined on $\mathcal{V}$.

Theorem 2: For any $\epsilon > 0$, there exist a positive integer N, m points $z_1, ..., z_m \in \mathcal{K}$, and real constants c_j, ζ_0^j, $\xi_{i,j}$, $i = 1, ..., N$, $j = 1, ..., m$ such that

$$\left| g(u) - \sum_{i=1}^{N} c_i \sigma \left(\sum_{j=1}^{m} \xi_{i,j} u(z_j) + \zeta_0^j \right) \right| < \epsilon \tag{10}$$

holds for all $u \in \mathcal{V}$.

Now, let $\mathcal{K}_1 \subseteq \mathcal{X}$, $K_2 \subseteq \mathbb{R}^n$ be two compact sets, $\mathcal{V}$ a compact set in $C(\mathcal{K}_1; \mathbb{R})$, and $\mathcal{G}$ a nonlinear continuous operator mapping $\mathcal{V}$ into $C(K_2; \mathbb{R})$.

Theorem 3: For any $\epsilon > 0$, there exist positive integers M, N, m, constants c_i^k, ζ_0^{jk}, $\xi_{i,j}^k \in \mathbb{R}$, points $\theta_k \in \mathbb{R}^n$, $z_j \in \mathcal{K}_1$, $i = 1, ..., M$, $k = 1, ..., N$, $j = 1, ..., m$ such that

$$\left| \mathcal{G}(u)(y) - \sum_{k=1}^{N} \sum_{i=1}^{M} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{i,j}^k u(z_j) + \zeta_0^{jk} \right) \cdot \sigma(\theta_k^T y + \theta_0^k) \right| < \epsilon \tag{11}$$

holds for all $u \in \mathcal{V}$ and $y \in K_2$.

D. B-splines

B-splines are piece-wise polynomial functions derived from slight adjustments of Bezier curves, aimed at obtaining polynomial curves that tie together smoothly. Here, we are interested not in representing geometric curves, but functions as in [9]. Therefore, we consider a parameter $t \in K = [a, b] \subseteq \mathbb{R}$, and $(c_i)_{i=1}^{\ell} \in \mathbb{R}$ as a set of ℓ control points for a spline curve $s(t)$ of degree d, with non-decreasing knots $(\hat{t}_i)_{i=1}^{\ell+d+1}$.

$$s(t) = \sum_{i=1}^{\ell} c_i B_{i,d}(t) \quad \text{for} \quad t \in K \subseteq \mathbb{R}, \tag{12}$$

where $B_{i,d}(t)$, $d > 1$, is given by the Cox-de Boor recursion formula [7]:

$$B_{i,d}(t) = \frac{t - \hat{t}_i}{\hat{t}_{i+d} - \hat{t}_i} B_{i,d-1}(t) + \frac{\hat{t}_{i+d+1} - t}{\hat{t}_{i+d+1} - \hat{t}_{i+1}} B_{i+1,d-1}(t), \tag{13}$$

and

$$B_{i,0}(t) = \begin{cases} 1, & \hat{t}_i \leq t < \hat{t}_{i+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

Property 1: Any value assumed by $s(t)$, $\forall t \in K$, lies in the convex hull of its ℓ control points $(c_i)_{i=1}^{\ell}$.

Property 2: Suppose that the number $\hat{t}_{i+1}$ occurs m times among the knots $(\hat{t}_j)_{j=i-d}^{m+d}$ with m an integer bounded by $1 \leq m \leq d + 1$, e.g., $\hat{t}_i < \hat{t}_{i+1} = \cdots = \hat{t}_{i+m} < \hat{t}_{i+m+1}$, then the spline function $s(t)$ has a continuous derivative up to order $d - m$ at knot $\hat{t}_{i+1}$.

This property implies that the smoothness of the spline can be adjusted using multiple knot points. A common choice is to set $m = d+1$ multiple knot points for the initial and final knot points. This way, Equation (12) assumes the first and final control points as initial and final values.

Property 3: the B-spline basis functions are continuous in t, $B_{i,d}(t) \in C(K; \mathbb{R})$, and bounded [14].

By defining the vectors

$$\begin{aligned} c &\triangleq [c_1\, c_2\, \ldots\, c_\ell]^T \\ B_d(t) &\triangleq [B_{1,d}(t)\, B_{2,d}(t)\, \ldots\, B_{\ell,d}(t)] \end{aligned} \tag{15}$$

(12) can be rewritten as $s(t) = B_d(t)c$.

B-splines are generally used to represent curves. However, our problem is a function of time. This can be conceptualized as a 2D curve. Therefore, each control point should be a 2D point. To ensure that $s(t)$ is indeed a function of time, we associate the parameter t with the time variable that varies within the interval $[a, b]$. We enforce that the position of each control point along the axis representing time t remains fixed by partitioning the interval $[a, b]$ into $\ell - 1$ equispaced sub-intervals. By doing so, we obtain the formulation presented in (15), where each control point is associated with only a scalar coefficient.

In this paper we are interested in the B-spline representation for the approximation of the solution of (4) where $x \in \mathbb{R}^n$ consists of a set of n scalar B-splines, one for each component of the state vector. In general, this B-spline is represented as

$$\underline{s}(t) = \underline{B}_d(t)\underline{c} \tag{16}$$

where $\underline{s} : \mathbb{R} \to \mathbb{R}^n$, and

$$\underline{B}_d(t) \triangleq \operatorname{diag}[B_d(t)] \tag{17}$$

$$\underline{c} \triangleq \left[\underline{c}_1^T, \cdots, \underline{c}_n^T\right]^T \tag{18}$$

$$\underline{c}_i \triangleq [c_{i1}, \cdots, c_{i\ell}]^T. \tag{19}$$

The index i indicates the i-th components of the multidimensional B-spline $\underline{s}$.

E. Problem Statement

Our focus is on providing future state estimation of nonlinear closed-loop systems of the form (4) by approximating the operator $\mathcal{P}(x)(t)$ (3) via a deep operator network that combines deep NNs and B-spline functions. By momentarily assuming that (4) is scalar, our goal is to approximate $\mathcal{P}(x)(t)$ as follows:

$$x(t) = \mathcal{P}(x)(t) \approx \hat{\mathcal{P}}(x)(t) = \sum_{i=1}^{\ell} c_i(x_0) B_{i,d}(t) \tag{20}$$

for $t \in [0, T]$, where the coefficients $c_i(x_0)$ represent the B-spline's control points along the time interval $[0, T]$, determining the shape of $x(t)$ depending on the IC x_0. The $c_i(x_0)$ is a functional as depending of the IC x_0 which is approximated by a deep NN. Building upon the theoretical framework established in [13], we extend the

theory to encompass (20). Subsequently, we delve into the multidimensional scenario, where $x \in \mathbb{R}^n$, particularly when addressing error bounds.

III. B-spline-Based Deep Neural Operator

Let us consider the same conditions of Theorem 3, modified for the problem of finding an approximating operator (20) for the scalar case. We consider $\mathcal{X} = C([0,T];\mathbb{R})$ as a Banach space with the norm $\|x\|_C = \max_{t\in[0,T]} \|x(t)\|$. Let $\mathcal{S} \subseteq \mathcal{X}$ be a compact set, and the operator $\mathcal{P}(x)(t)$ maps an element of $\mathcal{S}$ into itself. The scalar IC $x_0 = x(0) \in \mathcal{S}$.

Theorem 4: For any $\epsilon > 0$, there exist positive integers M, ℓ, constants c_i^k, ζ_0^{jk}, $\xi_i^k \in \mathbb{R}$, functions $B_{k,d}(t) \in C([0,T],\mathbb{R})$, and $x_0 \in \mathcal{S}$ such that

$$\left| \mathcal{P}(x)(t) - \sum_{k=1}^{\ell} \sum_{i=1}^{M} c_i^k \sigma\left(\xi_i^k x_0 + \xi_0^{ik}\right) \cdot B_{k,d}(t) \right| \leq \epsilon \quad (21)$$

holds for all $x_0 \in \mathcal{S}$ and $t \in K_2$.

Proof: To prove this theorem, we adopt a high-level approach similar to that used in [4], omitting a detailed proof for brevity. In this case, the branch network $\sum_{i=1}^{M} c_i^k \sigma\left(\xi_i^k x_0 + \xi_0^{ik}\right)$ is expressed in the same form as in Theorem 3, except the trunk network is replaced by the B-spline basis functions $B_{k,d}(t)$. Recalling the proof of Theorem 3 (Theorem 5 in [13]), we need to show that we can write

$$\left| \mathcal{P}(x)(t) - \sum_{k=1}^{\ell} c_k(\mathcal{P}(x)) B_{k,d}(t) \right| \leq \epsilon/2 \quad (22)$$

Theorem 1 cannot be used since $B_{k,d}(t)$ is a polynomial function. Instead, we use the Weierstrass approximation theorem [15]. This theorem is proven by using a linear combination of Bernstein polynomials used as basis functions. This linear combination of polynomials is also used to generate Bézier curves which are a particular subtype of B-spline functions [16]. In particular, it is possible to transform a Bézier curve into a B-spline and vice-versa [14], therefore we can switch from a representation with B-spline basis function $B_{k,d}(t)$ to Bernstein polynomials and vice-versa. This implies that we can use B-splines as universal approximators of functions, and for this reason, (22) holds. ∎

For the case of $x \in \mathbb{R}^n$, it is possible to redefine Theorem 4 as Theorem 2 in [4]. Specifically, we need to define one B-spline for each component of the state, and the input to the network is a vector $x_0 \in \mathbb{R}^n$, where for each neuron i, we have $\sigma\left(\sum_{j=1}^{n} \xi_{i,j}^k x_0^j + \zeta_0^{ik}\right)$.

A. IC and control points mapping

Let us assume that for a specific IC $x_0 \in W \subseteq \mathbb{R}$ we compute the particular solution $x(t)$ for $t \in [0,T]$. We can take N samples of the solution $x(t)$ by properly choosing a sampling step h and we can form a vector of measurements $y_m = [x(0), x(h), x(2h), ..., x((N-1)h)]^T$ where in this case the values of $x(h)$ are scalar values. From (15), we can write

$$y_m = C_m c_{x_0}; \qquad C_m = \begin{bmatrix} B_d(0) \\ B_d(h) \\ \vdots \\ B_d((N-1)h) \end{bmatrix} \in \mathbb{R}^{N\times\ell}. \quad (23)$$

where ℓ is the number of control points. The vector of control points defined in (15) is renamed c_{x_0} to indicate the solution to a specific trajectory $x(t)$ generated from x_0. From the B-spline properties, the rank of C_m is equal to ℓ. It is possible to find the vector of control points c_{x_0} by using the pseudo inverse $c_{x_0} = (C_m^T C_m)^{-1} C_m^T y_m$ which minimizes $\|y_m - C_m c_{x_0}\|^2$ (see [17]). As a consequence of the Weierstrass approximation theorem, it follows that by appropriately choosing values for ℓ and h, the error can be reduced to a desired level.

$$|x(t) - B_d(t) c_{x_0}| < \epsilon.$$

For any given IC $x_0 \in W$ we find a unique sequence of control points. In fact, the first control point of the sequence corresponds with the IC itself. This means that there is a mapping $\mathcal{M}$ that associates to each IC $x_0 \in W$ a sequence of ℓ control points.

From Theorem 4 it follows that

$$\mathcal{M}(x_0) \approx \hat{\mathcal{M}}(x_0) = \begin{bmatrix} \sum_{i=1}^{M} c_i^1 \sigma(\xi_i^1 x_0 + \zeta_0^{i1}) \\ \vdots \\ \sum_{i=1}^{M} c_i^\ell \sigma(\xi_i^\ell x_0 + \zeta_0^{i\ell}) \end{bmatrix} \quad (24)$$

where the NN is used to approximate the mapping $\mathcal{M}$. In this way, thanks to the B-spline representation, an infinite dimensional problem has been reduced to a finite one. The use of B-splines guarantees that the approximation of the solution of $x(t)$ is a continuous function defined for any $t \in [0,T]$. In the next subsection we define the problem for the multidimensional case $x \in \mathbb{R}^n$, formalize the existence of the mapping $\mathcal{M}$, and provide some error bounds.

B. Multidimensional Solution and Error Bounds

For the general problem where $\hat{\mathcal{P}}(x)$ approximates the solution of n dimensional ODEs such as (4), (20) can be written as

$$\hat{\mathcal{P}}(x)(t) = \underline{B}_d(t) \hat{\mathcal{M}}(x_0) \quad (25)$$

where $\underline{B}_d(t) \in \mathbb{R}^{n\times(n\cdot\ell)}$ accordingly with (16), and the network $\hat{\mathcal{M}} : \mathbb{R}^n \to \mathbb{R}^{n\cdot\ell}$ has n inputs and $n \cdot \ell$ outputs.

Lemma 1: For the autonomous system described by (4), where x_0 denotes the IC in $W \subset \mathbb{R}^n$, and $x(t)$ denotes a solution for the time interval $[0,T]$, with $h > 0$ representing a sampling time and $\ell > 0$ indicating the number of B-spline control points for each component of the state x, there exists a mapping $\mathcal{M} : \mathbb{R}^n \to \mathbb{R}^{n\cdot\ell}$ and a positive scalar ϵ such that

$$\|x(t) - \underline{B}_d(t) \mathcal{M}(x_0)\| \leq \epsilon \quad (26)$$

where, $\underline{B}_d(t) \in \mathbb{R}^{n\times(n\cdot\ell)}$.

Proof: For each i-th component of a solution $x(t)$ obtained for a specific IC $x_0 \in W$, we can formulate the least squares problem (23), where $y_{m,i} = C_m c_{i,x_0}$. Here, we define:

$$\underline{y}_m = \left[y_{m,1}^T, \cdots, y_{m,n}^T\right], \ \underline{C}_m = \text{diag}[C_m]$$

and $\underline{c}_{x_0} = \left[c_{1,x_0}^T, \cdots, c_{n,x_0}^T\right]^T$. This problem assumes the least squares form $\underline{y}_m = \underline{C}_m \underline{c}_{x_0}$, which can be solved using the pseudo-inverse. As $\underline{C}_m$ is a block diagonal matrix where each block has ℓ independent columns, a unique sequence of control points is generated by the least squares. Consequently, there exists a mapping $\mathcal{M}$ that maps any x_0 into the B-spline control points domain, ensuring (26) holds. ∎

Similarly to the approach in [18], where a FNN is employed to approximate the one-step evolution function, we utilize a generalization of Theorem 1 in [19] to establish the following proposition:

Proposition 1: Considering the conditions of Lemma 1 and a mapping $\mathcal{M}$ satisfying (26), there exists an FNN $\hat{\mathcal{M}}$ such that

$$\|\mathcal{M}(\cdot) - \hat{\mathcal{M}}(\cdot)\| \leq \gamma \tag{27}$$

for any $x_0 \in W$.

We define $\hat{x}(t) \triangleq \underline{B}_d(t)\hat{\mathcal{M}}(x_0)$ and $\tilde{x}(t) \triangleq \underline{B}_d(t)\mathcal{M}(x_0)$, where $\mathcal{M}$ and $\hat{\mathcal{M}}$ retain their meanings from Proposition 1.

Lemma 2: Under the conditions of Lemma 1, for the problem of finding $\hat{\mathcal{P}}(x)$, there exists a finite $M > 0$ such that

$$\|x(t) - \hat{x}(t)\| \leq \epsilon + M\gamma \tag{28}$$

for any $t \in [0, T]$, where ϵ and γ are the bounds related to the least squares and FNN approximation, respectively.

Proof: For any $t \in [0, T]$, we can write

$$\begin{aligned} \|x(t) - \hat{x}(t)\| &= \|x(t) - \hat{x}(t) + \tilde{x}(t) - \tilde{x}(t)\| \\ &\leq \|x(t) - \tilde{x}(t)\| + \|\tilde{x}(t) - \hat{x}(t)\| \end{aligned} \tag{29}$$

From (26) of Lemma 1, we have $\|x(t) - \tilde{x}(t)\| \leq \epsilon$, whereas

$$\begin{aligned} \|\tilde{x}(t) - \hat{x}(t)\| &= \|\underline{B}_d(t)(\mathcal{M}(x_0) - \hat{\mathcal{M}}(x_0))\| \\ &\leq \|\underline{B}_d(t)\|\|\mathcal{M}(x_0) - \hat{\mathcal{M}}(x_0)\| \leq M\gamma \end{aligned} \tag{30}$$

where γ results from Proposition 1, and M results from the boundedness of the B-spline basis functions [14]. ∎

IV. Experiments

We demonstrate the value of this technique through several experiments with a quadrotor with nonlinear dynamics and controlled with a linear quadratic regulator (LQR) controller [20]. The input in every experiment was the 12-dimensional IC, consisting of the 3-D position, velocity, angular orientation, and angular velocity. The output is a set of control points for a known sampling interval and time horizon.

We randomly selected ICs within a 12-dimensional ball: 2 meters in each spatial dimension, 2 meters per second in velocity, $\frac{\pi}{4}$ radians in angular dimensions, and 5 radians per second in angular velocity. These conditions defined trajectories assuming an equilibrium point at $\vec{0}$ (referred to as x_{eq}). We calculated control points for each trajectory using least squares fitting and a 3rd-order B-spline ($d = 3$). Our training dataset included 5000 ICs. To enhance rotational equivariance, we rotated randomly generated ICs by π, $-\frac{\pi}{2}$, $-\frac{\pi}{4}$, $-\frac{\pi}{6}$, $\frac{\pi}{4}$, $\frac{\pi}{3}$, and $\frac{\pi}{2}$ radians around the Z-axis. In our initial experiment, we used 50 control points with a 2.5-second time horizon. Our network was a simple fully-connected neural network (FCNN) with 12 layers, each containing 120 neurons. In order to examine model variance, training was reinitialized 10 times and training was stopped after 2 hours or when the loss had reached about 10^{-5}, which condition was satisfied first. One model out of the 10 attempts did not converge within this loss condition and, thus, was not used in the subsequent analysis.

A FCNN does not exploit the physical properties of the system. In our second experiment we combined a gated recurrent unit (GRU) in combination with a FCNN in a recurrent neural network (RNN) architecture. Both the GRU and the subsequent fully connected layers had a width of 120. The fully connected layers had a depth of 3. This resulted in a lower validation loss (mean squared error) for fewer than half as many parameters, as can be seen in Table I.

V. Results

We used the trajectories and control points found using least squares fitting via singular value decomposition in training the NNs. In order to evaluate and compare across experiments, we evaluated against a test set containing 1000 new ICs. In Table I we compare the timing between different experiments and the ODE Solver We find that the FCNNs is 5 times faster than least squares fitting and the RNN is 5 times slower than SciPy ODE solver. The increase in evaluation time between the FCNN and the RNN is a consequence of a for-loop required to calculate the entire trajectory. Notably, the ODE solver used is a python wrapper around a ODEPACK library whereas the NNs were built using PyTorch. Therefore, we advise caution when comparing these evaluation times.

In both cases, the error is correlated with the IC, parameterized by the 'radius' in the 12-D ball of the IC, as seen in Figure 1. This result is aligned with several factors: 1) Since x_{eq} is asymptotically stable, the radius of the IC in the unit ball serves as a measure of the distance the quadrotor is from it; 2) Our ICs were randomly sampled within the cube of the radius to balance comprehensive sampling throughout the volume and sampling at different radii. Consequently, the training points density decreases with IC radius.

Summary statistics can only provide a partial understanding of prediction accuracy. We chose four specific ICs to analyze qualitatively by varying the IC in a single dimension. These are shown in Figure 2. The ICs we considered were (with all unspecified values set to zero): $x_0^y = (0.8\,m, 1.2\,m)$ - Red (solid and dotted) lines indicate the least-squares ("ground truth") trajectories and the pink shading indicates the range of the respective neural-network approximated trajectories.; $x_0^y = 1.0\,m, x_0^{dy} = (0.8\ \ m/s, 1.2\,m/s)$ - Blue lines and shaded regions; $x_0^y = -1.0\,m, x_0^{d\theta} =$

TABLE I: Time comparisons for different methods.

Method	T[sec]	ℓ	# Parameters	Mean Final MSE (Loss)	# Models	Evaluation Time [sec]
ODE solver	2.5	50	-	-	-	0.0049 ± 0.0015
FCNN	2.5	50	219360	$1.4 \pm 0.2 \times 10^{-5}$	9	0.00097 ± 0.00023
RNN	2.5	50	78732	$2.9 \pm 0.6 \times 10^{-6}$	10	0.028 ± 0.003

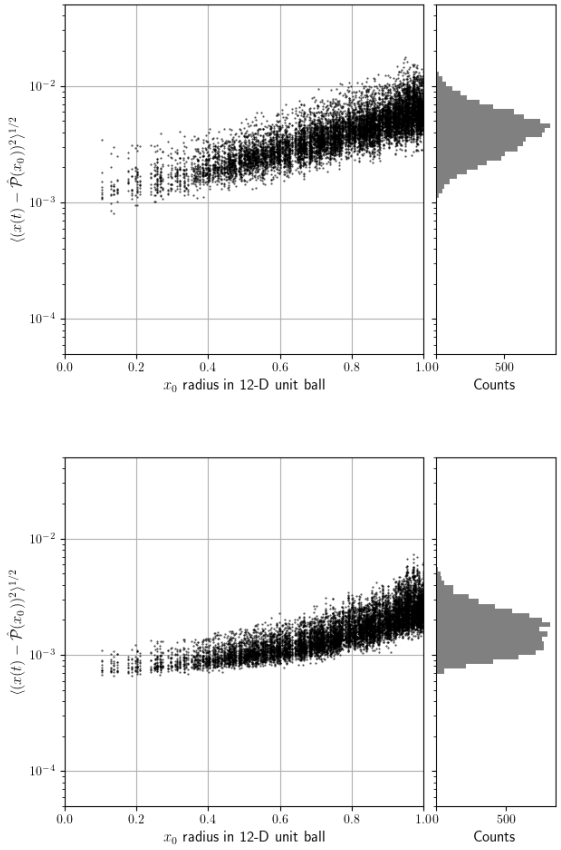

Fig. 1: (top) FCNN and (bottom) RNN root mean squared error versus IC radius. The error is positively correlated with distance. The profile histogram (right) shows the overall error.

$(4\,rad/s, 6\,rad/s)$ - Green lines and shaded regions; $x_0^y = -1.0\,m$, $x_0^{d\theta_y} = 5\,rad/s, x_0^{\theta_y} = \pi/16 \cdot (3, 5)$ - Orange lines and shaded regions; $x_0 = R/2$ where R is the maximum value in any dimension, except for $x_z = -R/2$. Black lines and gray shaded regions. In this case, all directions were varied slightly.

This study examines per-trajectory fit, comparing variation due to network architecture and training with variation in ICs. Both NN approximations (shaded regions) and least squares fits of spline control points (lines) show the impact of IC variation. However, the shaded regions also account for model stochasticity. Notably, IC variance dominates, except in the final case where all dimensions are far from zero. Additionally, the least squares trajectory consistently falls within the range found by the NN, although only four of twelve dimensions are shown.

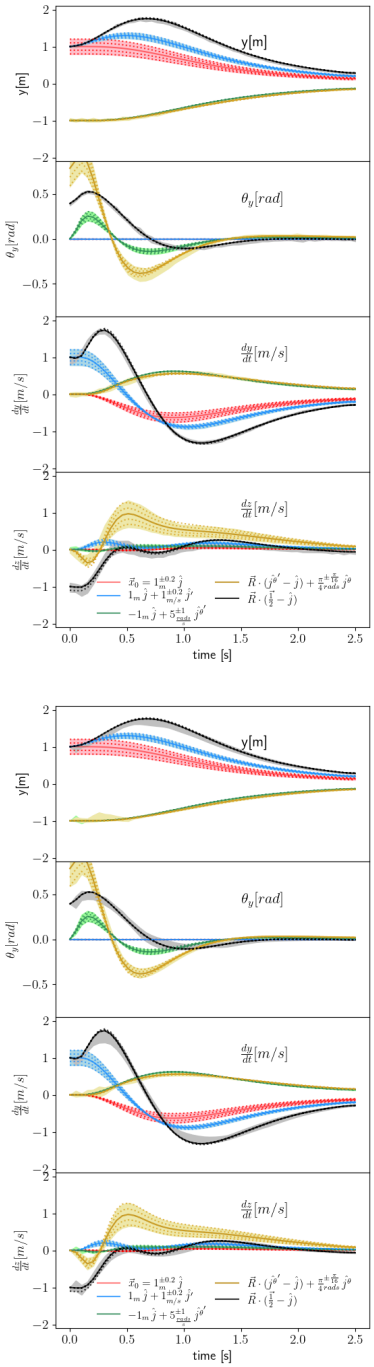

Fig. 2: (top) FCNN and (bottom) RNN predicted and least-squares fitted trajectories for sets of IC, varied about a central value as indicated on the legend. Only four dimensions are shown to conserve space.

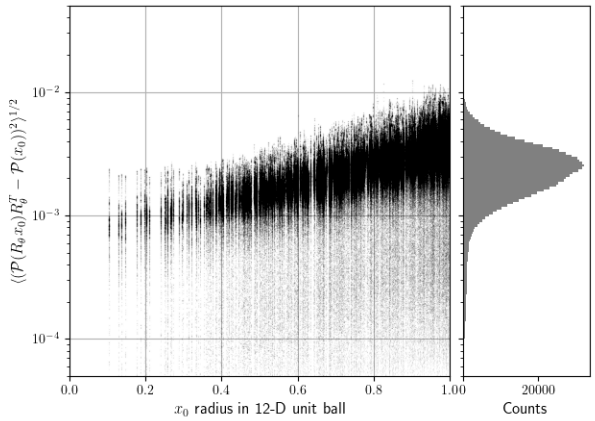

Fig. 3: ICs are sampled at 85 angles about Z axis from $\pi/8$ to $15\pi/8$. The neural operators (here, FCNN, the RNN behaves similarly) should display axial symmetry. The root mean squared trajectory difference vs. IC radius is shown.

We consider rotations of the ICs and the symmetries that should be present. Figure 3, shows the root mean squared difference between our original testing dataset and the test dataset rotated to 85 evenly spaced angles between $\pi/8$ and $15\pi/8$ and the resulting trajectory, rotated back. An ideal equivariant operator has the property $R_\theta^T \mathcal{P}(R_\theta x_0) = R_\theta^T R_\theta \mathcal{P}(x_0) = \mathcal{P}(x_0)$ and the root mean squared difference, $\langle (R_\theta^T \mathcal{P}(R_\theta x_0) - \mathcal{P}(x_0))^2 \rangle^{1/2} = 0$. We find that this deviation is comparable to the overall error exhibited by the respective NNs.

VI. Conclusions and Future Work

We proposed a theoretical framework that combines NN and B-splines as universal approximates of operators. The B-splines basis was motivated by the ability to exploit control points for fast safety assurance and its relation to Gaussian Process Regression, which we believe may provide mechanisms of quantifying uncertainty. As a result of this work, we will explore the connection between uncertainty quantification and control points in order to take advantage of the convex-hull property of the B-splines to assess safety. We will also explore using this approach with physics informed training methods to influence training with an informed bias. Additionally, exploring adjustments to the network architecture and training could enhance performance. In addition, follow-on work involves developing a framework to estimate and fine-tune the NN errors by leveraging the B-spline convex hull property. Ensuring that the actual state trajectory resides within the convex hull of the control points enables real-time evaluations of safety violations. Finally, we wish to extend this framework to non-autonomous systems to advance the development of NN-based controllers.

VII. Acknowledgements

Copyright 2024 Carnegie Mellon University and Raffaele Romagnoli. This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution. DM24-0319

References

[1] A. Platzer, *Logical Foundations of Cyber-Physical Systems*. Springer, 2018.

[2] D. de Niz, B. Andersson, M. Klein, J. Lehoczky, A. Vasudevan, H. Kim, and G. Moreno, "Mixed-trust computing for real-time systems," in *2019 IEEE 25th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 1–11, IEEE, 2019.

[3] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, *et al.*, "Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence," tech. rep., USDOE Office of Science (SC), Washington, DC (United States), 2019.

[4] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," *Nature machine intelligence*, vol. 3, no. 3, pp. 218–229, 2021.

[5] T. X. Nghiem, J. Drgoňa, C. Jones, Z. Nagy, R. Schwan, B. Dey, A. Chakrabarty, S. Di Cairano, J. A. Paulson, A. Carron, *et al.*, "Physics-informed machine learning for modeling and control of dynamical systems," in *2023 American Control Conference (ACC)*, pp. 3735–3750, IEEE, 2023.

[6] A. Saviolo, G. Li, and G. Loianno, "Physics-inspired temporal learning of quadrotor dynamics for accurate model predictive trajectory tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10256–10263, 2022.

[7] C. De Boor and C. De Boor, *A practical guide to splines*, vol. 27. springer-verlag New York, 1978.

[8] K. Doległo, A. Paszyńska, M. Paszyński, and L. Demkowicz, "Deep neural networks for smooth approximation of physics with higher order and continuity b-spline base functions," *arXiv preprint arXiv:2201.00904*, 2022.

[9] L. Jetto, V. Orsini, and R. Romagnoli, "B-splines and pseudo-inversion as tools for handling saturation constraints in the optimal set-point regulation," in *2017 American Control Conference (ACC)*, pp. 1041–1048, IEEE, 2017.

[10] X. Bay, L. Grammont, and H. Maatouk, "Generalization of the kimeldorf-wahba correspondence for constrained interpolation," 2016.

[11] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002.

[12] S. Bittanti and G. Picci, *Identification, adaptation, learning: the science of learning models from data*, vol. 153. Springer Science & Business Media, 1996.

[13] T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," *IEEE transactions on neural networks*, vol. 6, no. 4, pp. 911–917, 1995.

[14] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline techniques*, vol. 6. Springer, 2002.

[15] K. R. Davidson and A. P. Donsig, *Real Analysis and Applications: Theory in Practice*. Springer New York, NY, 2010.

[16] F. Chudy and P. Woźny, "Linear-time algorithm for computing the bernstein–bézier coefficients of b-spline basis functions," *Computer-Aided Design*, vol. 154, p. 103434, 2023.

[17] R. Romagnoli and E. Garone, "A general framework for approximated model stable inversion," *Automatica*, vol. 101, pp. 182–189, 2019.

[18] T. Qin, Z. Chen, J. D. Jakeman, and D. Xiu, "Data-driven learning of nonautonomous systems," *SIAM Journal on Scientific Computing*, vol. 43, no. 3, pp. A1607–A1624, 2021.

[19] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta numerica*, vol. 8, pp. 143–195, 1999.

[20] R. Romagnoli, B. H. Krogh, D. de Niz, A. D. Hristozov, and B. Sinopoli, "Software rejuvenation for safe operation of cyber–physical systems in the presence of run-time cyberattacks," *IEEE Transactions on Control Systems Technology*, 2023.