Physica D 470 (2024) 134364

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

NINNs: Nudging induced neural networks☆

Harbir Antil [a], Rainald Löhner [b], Randy Price [c,*]

[a] Department of Mathematical Sciences and The Center for Mathematics and Artificial Intelligence, George Mason University, Fairfax, VA 22030, United States of America

[b] The Center for Computational Fluid Dynamics, George Mason University, Fairfax, VA 22030, United States of America

[c] The Center for Mathematics and Artificial Intelligence and The Center for Computational Fluid Dynamics, George Mason University, Fairfax, VA 22030, United States of America

ARTICLE INFO

Communicated by Boumediene Hamzi

MSC:
93C20
93C15
68T07
80A32
76B75

Keywords:
Nudging informed neural networks
NINNs
Deep neural networks
Convergence analysis
Data assimilation
Chemically reacting flows

ABSTRACT

Nudging induced neural networks (NINNs) algorithms are introduced to control and improve the accuracy of deep neural networks (DNNs). The NINNs framework can be applied to almost all pre-existing DNNs, with forward propagation, with costs comparable to existing DNNs. NINNs work by adding a feedback control term to the forward propagation of the network. The feedback term nudges the neural network towards a desired quantity of interest. NINNs offer multiple advantages, for instance, they lead to higher accuracy when compared with existing data assimilation algorithms such as nudging. Rigorous convergence analysis is established for NINNs. The algorithmic and theoretical findings are illustrated on examples from data assimilation and chemically reacting flows.

1. Introduction

To illustrate the proposed ideas, let us recall that Residual neural networks (ResNets) are an established way to do supervised machine learning and multiple authors have made the connection between ResNets and ODEs which has helped prove their stability [1–3]. Next, we introduce a general ResNet with input y_0, layers

$$y_{\ell+1} = y_\ell + \tau F(\theta_\ell, y_\ell) \quad \text{for} \quad \ell = 0, \dots, L-1, \tag{1.1}$$

where θ_ℓ are the weight and bias parameters for the ℓth layer, and τ is a positive parameter. The function F takes the form $W(\theta_\ell)y_\ell + b(\theta_\ell)$ where the matrix W is determined by the weight parameters and the vector b is determined by the bias parameters. The corresponding continuous dynamical system is given by

$$d_t y = F(\theta(t), y), \qquad y(0) = y_0. \tag{1.2}$$

The ResNet (1.1) can be seen as the forward Euler discretization of the initial value problem (1.2) [3–6]. In this paper, we leverage the connection between ResNets and ODEs to nudge a ResNet towards a given quantity of interest (QoI), by introducing a feedback law. This approach is hereby termed as Nudging Induced Neural Networks (NINNs). NINNs are applied to data assimilation and realistic chemically reacting flow problems. Also a rigorous convergence analysis is established for NINNs.

Data assimilation techniques are used to improve our knowledge about the state by combining the model with the given observations. The standard nudging algorithm is widely used in data assimilation. In the past, nudging was applied to finite-dimensional dynamical systems governed by ordinary differential equations with applications in meteorology [7–12]. As nudging has matured, it has been extended to more general situations, including partial differential equations [13–21]. Given a continuous dynamical system

$$\partial_t u = f(u(t)), \tag{1.3}$$

☆ This work is partially supported by the Defense Threat Reduction Agency (DTRA) under contract HDTRA1-15-1-0068. Jacqueline Bell served as the technical monitor. Also, it is partially supported by NSF grants DMS-2110263, DMS-1913004, the Air Force Office of Scientific Research under Award NOs: FA9550-19-1-0036 and FA9550-22-1-0248.

* Corresponding author.
E-mail addresses: hantil@gmu.edu (H. Antil), rlohner@gmu.edu (R. Löhner), rprice25@gmu.edu (R. Price).

https://doi.org/10.1016/j.physd.2024.134364
Received 3 March 2023; Received in revised form 26 March 2024; Accepted 5 September 2024
Available online 12 September 2024
0167-2789/Published by Elsevier B.V.

with unknown initial conditions, the nudging algorithm entails solving

$$\partial_t w = f(w) - \mu(I_M w - w^{QoI}), \qquad w(0) = w_0 \text{ (arbitrary)}, \tag{1.4}$$

where I_M is a linear operator called the interpolant operator and w^{QoI} is a quantity of interest (QoI). Here $\mu > 0$ is the nudging parameter. In the case $w^{QoI} = I_M(u)$, it is possible to establish approximation error estimates between u and w solving (1.3) and (1.4), respectively. In the case $w^{QoI} \neq I_M(u)$, w^{QoI} is chosen by the user to nudge the solution towards some desired behavior, i.e. 0 in a particular region of the domain. Recently in [22], the authors replace the discrete version (2.1) of (1.4) by a ResNet (1.1). This provides a new and cheaper alternative to nudging as no expensive simulations are needed to generate the nudging solution. Error estimates have also been derived. The present work builds upon and improves the method presented in the article.

Motivated by nudging (1.4), this work presents a completely new class of algorithms called NINNs which are meant to directly control DNNs, such as ResNets (1.1), by appropriately applying nudging (feedback). One example of NINNs is

$$y_{\ell+1} := y_\ell + \tau F(\theta_\ell, y_\ell) - \tau\mu g_{\ell+1}(y^{QoI}, y_\ell), \quad \ell = 1, \dots, L-2. \tag{1.5}$$

The NINNs feedback term g_ℓ is a function that depends on the current layer y_ℓ and the current quantity of interest y^{QoI}. Notice that, the NINNs feedback term is applied to the forward propagation (1.1) after the training has been carried out. Furthermore, the NINNs framework is a type of feedback control for DNNs. This framework offers multiple advantages, for example, it leads to new data assimilation algorithms with similar accuracy in comparison to the nudging data assimilation algorithm for ODEs. One drawback with NINNs is the requirement of training data for the neural network.

Performance of NINNs is illustrated using two types of examples. In the first example, we use NINNs as a data assimilation algorithm. In the second example, we consider a realistic chemically reacting flow problem to illustrate uses for NINNs in simulating stiff ODEs. In case of stiff or chaotic systems [23], a proven technique is to learn the update map $u^{n-1} \to u^n$, i.e., one time step solution. The neural net takes as input the state at time t_{n-1} and the output is the approximate state at time t_n. This is the approach we will take for the ResNets in the numerical section, more details are given in Section 5.1.

Outline: Section 2 contains preliminary material which is necessary to introduce NINNs in Section 3. Section 4 contains error analysis of NINNs and Section 5 covers preliminary material for the experiments. Next we provide experimental results from using NINNs as a data assimilation algorithm in Section 6. In Section 7, we provide experimental results showcasing NINNs ability to improve previously trained networks for chemically reacting flows.

2. Objectives and definitions

The goal of this section is to explain the nudging algorithm in more detail and to describe the ResNet architecture so that in Section 3 we can smoothly transition into NINNs. The contents of the next three Sections 2.1–2.3 are by now well-known, see for instance [6,22–24].

2.1. Nudging

Consider a dynamical system given by a differential equation (1.3) with $u(t) \in X$. Here X is either an infinite dimensional space (in case of PDEs) or a finite dimensional space $X = \mathbb{R}^d$ in case of vector ODEs. The continuous nudging algorithm with a continuous in time quantity of interest (QoI) w^{QoI} is given by (1.4). More realistic is the case with discrete QoI at times $\{t_n\}$ [20]:

$$\partial_t w = f(w(t)) - \mu(I_M(w(t)) - w^{QoI}(t_n)), \quad \forall t \in [t_n, t_{n+1}], \quad w(0) = w_0. \tag{2.1}$$

Notice that, in both cases we have QoI of the form $\{w^{QoI}(t)\}_{t\in\mathbb{R}^+}$ or $\{w^{QoI}(t_n)\}_{n=1}^{\infty}$ which are an input to the algorithm. Moreover, $I_M : X \mapsto X$ is an interpolant operator acting on X. In the case of ODEs, an example of I_M is the orthogonal projection operator onto a subset K of X. In this case $X = \mathbb{R}^d$ and $K = \text{span}\{\phi_i\}_{i=1}^{d_K}$ with $\phi_i = \mathbf{e}_i$ the standard basis elements in $\mathbb{R}^d$ and $d \geq d_K$.

The goal of nudging is to nudge the solution of (1.4) or (2.1) towards the given quantity of interest. A typical QoI is given by partial observations of a particular solution of (1.3), i.e., $w^{QoI} = I_M(u)$, namely we are nudging w towards a particular solution u. This is useful for data assimilation where the initial condition is an unknown and the user is given partial observations of u. In this scenario we desire $\|w - u\| \to 0$, where w is the nudging solution, u is the reference solution, and $\|\cdot\|$ is an appropriate norm.

2.2. DNN with bias ordering

The next two subsections cover the ResNet details. We are particularly interested in ResNets that approximate a dynamical system (1.3). A proven technique is to teach the ResNet the update map $u(t_n) \to u(t_{n+1})$ which we denote by S below. First introduced in [24], consider the following optimization problem which abstractly represents the training of the DNNs

$$\min_{\{W_\ell\}_{\ell=0}^{L-1}, \{b_\ell\}_{\ell=0}^{L-2}} J(\{(y_L^i, S(u^i))\}_i, \{W_\ell\}_\ell, \{b_\ell\}_\ell), \tag{2.2a}$$

$$\text{subject to } y_L^i = \mathcal{F}(u^i; (\{W_\ell\}, \{b_\ell\})), \quad i = 1, \dots, N_s, \tag{2.2b}$$

$$b_\ell^j \leq b_\ell^{j+1}, \quad j = 1, \dots, n_{\ell+1} - 1, \quad \ell = 0, \dots, L-2. \tag{2.2c}$$

The input–output pairs used in training are represented by $\{u^i\}_{i=1}^{N_s}$ and $\{S(u^i)\}_{i=1}^{N_s}$, with N_s denoting the number of samples. The goal is to minimize the difference between the DNN output (DNN is represented by $\mathcal{F}$ in (2.2b)) y_L^i and the true output $S(u^i)$ using the loss function J from (2.2a). Bias ordering is enforced in each layer by (2.2c) and its purpose is to decrease the parameter search space, see [24] for more details. The weight matrix is $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell+1}}$, and the bias vector is $b_\ell \in \mathbb{R}^{n_{\ell+1}}$ where the ℓth layer has n_ℓ neurons. The next subsection describes the exact layout of the DNNs we consider in this article.

In our numerical experiments the loss function J in (2.2a) will be

$$J := \frac{1}{2N}\sum_{i=1}^{N} \|y_L^i - S(u^i)\|_2^2 + \frac{\lambda}{2}\sum_{\ell=0}^{L-1}(\|W_\ell\|_1 + \|b_\ell\|_1 + \|W_\ell\|_2^2 + \|b_\ell\|_2^2), \tag{2.3}$$

where $\lambda \geq 0$ is the regularization parameter. The second summation regularizes the weights and biases. Following [24], and motivated by Moreau-Yosida regularization, the bias ordering (2.2c) is implemented as an additional penalty term in J

$$J_\gamma := J + \frac{\gamma}{2}\sum_{\ell=0}^{L-2}\sum_{j=1}^{n_{\ell+1}-1} \|\min\{b_\ell^{j+1} - b_\ell^j, 0\}\|_2^2. \tag{2.4}$$

Here γ is a penalization parameter. For convergence results as $\gamma \to \infty$, see [24].

2.3. DNN structure

To introduce NINNs, we will use ResNets with the following form with input $y_0 \in \mathbb{R}^d$, inner layer feature vectors $y_\ell \in \mathbb{R}^{n_\ell}$, and output feature vector $y_L \in \mathbb{R}^{d^*}$

$$\begin{aligned} y_1 &:= \sigma(W_0 y_0 + b_0), \\ y_{\ell+1} &:= y_\ell + \tau\sigma(W_\ell y_\ell + b_\ell), \quad \ell = 1, \dots, L-2, \\ y_L &:= W_{L-1} y_{L-1}. \end{aligned} \tag{2.5}$$

The scalar $\tau > 0$ and the activation function σ are user defined. For the purpose of this work, we have chosen a smooth quadratic approximation of the ReLU function,

$$\sigma(x) = \begin{cases} \max\{0, x\} & |x| > \epsilon, \\ \frac{1}{4\epsilon}x^2 + \frac{1}{2}x + \frac{\epsilon}{4} & |x| \leq \epsilon. \end{cases}$$

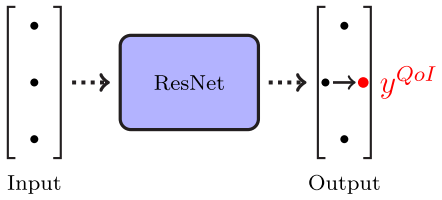

Fig. 1. The figure depicts a ResNet with input/output in $\mathbb{R}^3$. The goal is to nudge the ResNet output towards the given y^{QoI} (second component).

The ReLU function is given by taking $\epsilon = 0$ in the above definition. The connection between ResNets and ODEs described in Eqs. (1.1) and (1.2) motivates the use of ResNets in this work. The nudging algorithm is known to be an effective tool for control of ODEs and PDEs. The objective of this work is to show that the ResNet (2.5) can be effectively nudged towards a given QoI in the same manner the nudging algorithm nudges ODEs and PDEs as described in Section 2.1.

3. NINNs

The user provides a trained ResNet of the form (2.5) and a QoI y^{QoI}. Then a general NINN introduces a feedback law into the ResNet:

$$\begin{aligned} y_1 &:= \sigma(W_0(y_0 - \mu g_0(y^{QoI}, y_0)) + b_0) - \tau\mu g_1(y^{QoI}, y_1), \\ y_{\ell+1} &:= y_\ell + \tau\sigma(W_\ell y_\ell + b_\ell) - \tau\mu g_{\ell+1}(y^{QoI}, y_\ell), \quad \ell = 1, \dots, L-2, \\ y_L &:= W_{L-1}y_{L-1} - \tau\mu g_L(y^{QoI}, y_{L-1}). \end{aligned} \tag{3.1}$$

The goal is to choose functions $\{g_\ell\}_{\ell=0}^{L}$ and parameter $\mu \in \mathbb{R}$ such that the output y_L of (3.1) is nudged towards y^{QoI}, i.e. the quantity $|y_L - y^{QoI}| \in \mathbb{R}$ is minimized, see Fig. 1. Notice, g_0 controls the input, g_L controls the output, and the remaining g_ℓ control the inner layers. Next we present choices for the functions g_ℓ and the details in the following subsections (see Fig. 1).

3.1. Methods

Let $f_{\ell+1}(x) := y_\ell + \tau\sigma(W_\ell x + b_\ell)$ for $\ell = 1, \dots, L-2$ and $f_L(x) := W_{L-1}x$. Next, we consider multiple cases: (a) $y_L, y^{QoI} \in \mathbb{R}$ and the layer width $n_1 = n_2 = \cdots = n_{L-1}$ is constant and (b) general ResNet.

Case 1 ($y_L, y^{QoI} \in \mathbb{R}$, $n_1 = n_2 = \cdots = n_{L-1}$):

$$g_\ell := (f_L \circ \dots \circ f_{\ell+1}(y_\ell) - y^{QoI}) \frac{1}{\|W_{L-1}\|_1} W_{L-1} \quad \text{for } \ell = 1, \dots, L-1. \tag{NINN \#1}$$

(NINN #1) is designed to control y_ℓ during the forward propagation of the ResNet such that the quantity $|y_L - y^{QoI}| \in \mathbb{R}$ is minimized. The difference $f_L \circ \dots \circ f_{\ell+1}(y_\ell) - y^{QoI} \in \mathbb{R}$ is a measure of the closeness of the current layer y_ℓ to the target QoI y^{QoI}. The n_ℓ vector $\frac{1}{\|W_{L-1}\|_1} W_{L-1} \in \mathbb{R}^{n_\ell}$ serves to match the current layer width and to ensure $|W_{L-1}y_{L-1} - y^{QoI}|$ is being minimized. This method comes with two restrictions on the ResNet. First, the neural network output $y_L \in \mathbb{R}$ and second, constant layer width. By training one of the aforementioned ResNets per output component, see Fig. 2, we can build ResNet systems to handle general output $y_L \in \mathbb{R}^n$. One downside of (NINN #1) is the extra computational cost associated with calculating $f_L \circ \dots \circ f_{\ell+1}(y_\ell)$ for $\ell = 1, \dots, L-1$.

Case 2 (General ResNet):

$$g_0 := y_0 - y^{QoI}_{input}, \quad g_L := W_{L-1}y_{L-1} - y^{QoI}_{output}$$
$$\text{and } g_\ell = 0 \text{ for } \ell = 1, \dots, L-1. \tag{NINN \#2}$$

(NINN #2) modifies only the input and output to the neural network. This method can be applied to any neural network as we are controlling the input and output only. Note, the introduction of y^{QoI}_{input} and y^{QoI}_{output} is to handle the case of different input and output dimensions. For example, when learning ODEs in Section 6, the dimension of y^{QoI}_{input} agrees with the full state dimension and the dimension of y^{QoI}_{output} is one. This is a product of using ResNet systems pictured in Fig. 2.

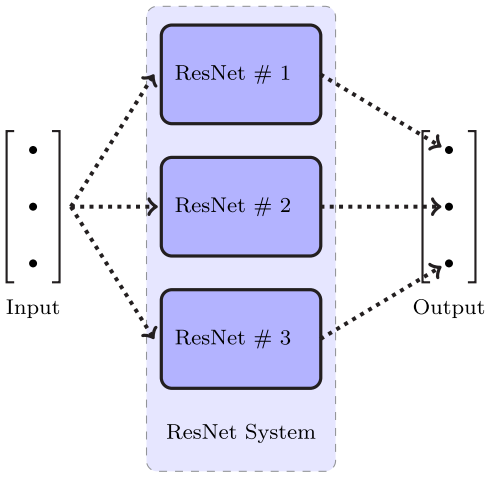

Fig. 2. A ResNet system consisting of 3 ResNets. Each ResNet is responsible for one component of the output. Nudging a component of the output towards a QoI requires the user to apply NINNs to the corresponding ResNet.

3.2. System of ResNets

In some cases, is it desired to use a system of ResNets instead of one ResNet. Each ResNet is then responsible for some of the output components. The benefits include increased surrogate accuracy and parallelization of the training, see Section 5 where we utilize the structure in our numerical examples and previous work [22–25]. To apply the NINNs framework (3.1) to a system of ResNets, such as shown in Fig. 2, we simply require applying the framework to each ResNet in the system the user desires to control. To control the first component of the output pictured in Fig. 2 requires applying the NINNs framework to ResNet #1 only.

4. Error analysis

In this section we provide error analysis resulting from applying NINNs to ResNets trained to learn a dynamical system,

$$\partial_t u = f(u(t)), \quad u(0) = u_0. \tag{4.1}$$

We will assume u is finite dimensional, either by construction or discretization. Given partial/incomplete QoI (or observations) taken from a dynamical system (4.1) solution, we will show NINNs can recover the solution under certain assumptions. First, we introduce the necessary definitions and assumptions.

Definition 4.1 (*Resnet State Space*). Consider a ResNet with inner layers of size n_ℓ with $\ell = 1, \dots, L-2$. The ResNet state space is given by $\mathbb{R}^{n_\ell}$. If the ResNet is a collection of N_R ResNets, then we define $n_\ell = \sum_{i=1}^{N_R} n_{\ell_i}$, where n_{ℓ_i} is the size of the ℓth inner layer for the ith ResNet.

Fig. 3 shows a ResNet system where each ResNet contains two layers with three neurons (also known as components) each. The ResNet state space is $\mathbb{R}^9$. The components of the ResNet state space ordering is shown in the figure.

We introduce $\chi_{\Omega_{QoI}} : \mathbb{R}^{n_\ell} \mapsto \mathbb{R}^{n_\ell}$, as an indicator type function that sets components without any NINNs feedback law to zero. Let $x \in \mathbb{R}^{n_\ell}$,

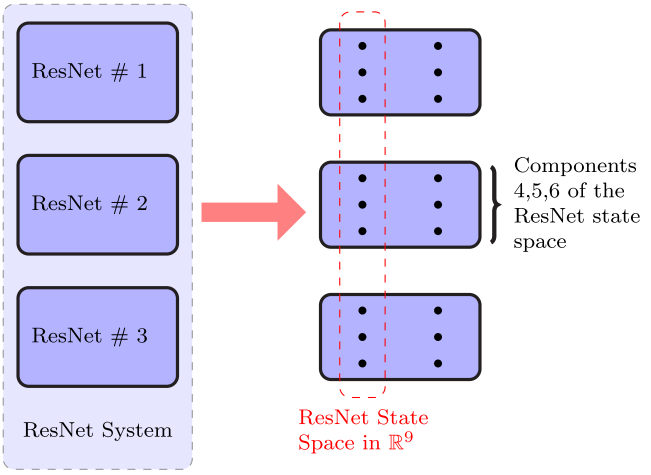

Fig. 3. A ResNet system consisting of 3 ResNets. Each ResNet contains two layers with three neurons in each layer. The ResNet state space is $\mathbb{R}^9$. Components of the ResNet state space are labeled from top to bottom.

then the ith component of $\chi_{\Omega_{QoI}}(x)$ is defined as:

$$[\chi_{\Omega_{QoI}}(x)]_i = \begin{cases} 0 & \text{if } i \notin \Omega_{QoI}, \\ [x]_i & \text{if } i \in \Omega_{QoI}, \end{cases} \tag{4.2}$$

where

$$\Omega_{QoI} = \{i \in \{1, 2, \dots, n_\ell\} | \\ \times \ i\text{th component of ResNet state space is nudged.}\}. \tag{4.3}$$

The notation $[x]_i$ signifies the ith component of a vector. It is easy to see $\chi_{\Omega_{QoI}}$ has the following property,

$$|\chi_{\Omega_{QoI}}(x)| \le |x|, \quad \forall x \in \mathbb{R}^{n_\ell}. \tag{4.4}$$

$|\cdot|$ represents the 2-norm in this paper. We re-introduce the interpolant operator $I_M : \mathbb{R}^d \mapsto \mathbb{R}^d$ where d accounts for the dimension of u in (4.1), recall (1.4).

Assumption 4.2 (*Stability of I_M*)**.** For all x in $\mathbb{R}^d$, there exists a constant $c_M \ge 0$ such that $|I_M(x) - x| \le c_M |x|$.

The partial/incomplete QoI (or observations) taken from (4.1) will be in the form $\{I_M(u(k\Delta t))\}_{k=0}^{\infty}$ where $\Delta t \in \mathbb{R}^+$ is a positive real number. Next we introduce the concept of a *continuous ResNet*, i.e., a ResNet with an infinite amount of layers. The forward propagation through the continuous ResNet is determined by $W(t) : \mathbb{R} \mapsto \mathbb{R}^{n_\ell \times n_\ell}$, $b(t) : \mathbb{R} \mapsto \mathbb{R}^{n_\ell}$ and activation function $\sigma : \mathbb{R}^{n_\ell} \mapsto \mathbb{R}^{n_\ell}$. We combine these functions into $f_{NN}(x, t) = \sigma(W(t)x + b(t))$. We represent the forward propagation of the whole continuous ResNet by the function $h(t) : [0, \infty) \mapsto \mathbb{R}^{n_\ell}$ satisfying

$$\begin{aligned} &\partial_t h = f_{NN}(h(t), t), \quad h(0) = x_0. \\ &h(t) = \int_0^t f_{NN}(h(s), s) ds + x_0. \end{aligned} \tag{4.5}$$

If we discretize (4.5) using Forward Euler then we arrive at (2.5).

Assumption 4.3 (*Continuous ResNet*)**.** There exists a continuous ResNet that, given $u(t)$ at time $t \ge 0$, can replicate the behavior of $u(t + \Delta t)$ (solution to (4.1)) with $\Delta t > 0$.

(i) The ResNet input and output lies in $\mathbb{R}^d$.[1]

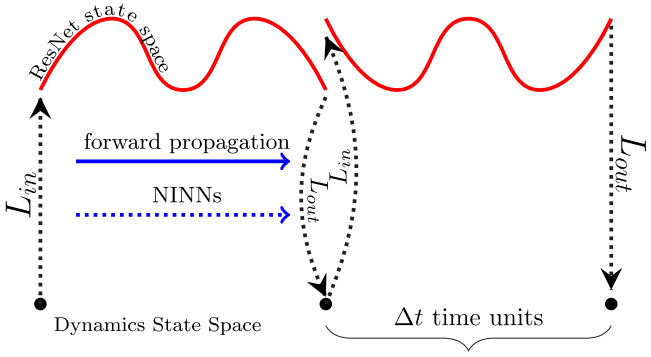

Fig. 4. The figure depicts the evolution of an arbitrary initial condition through the trained ResNet.

(ii) The layer width n_ℓ is the same except for the input and output.
(iii) The forward propagation is given by (4.5) with f_{NN} being Lipschitz continuous with Lipschitz constant $\widetilde{K}$.

Definition 4.4 (*Input–Output Transformation*)**.** Define, $L := L_{in} \circ L_{out}$. $L_{out} : \mathbb{R}^{n_\ell} \mapsto \mathbb{R}^d$ is a linear transformation with Lipschitz constant K_{out} that maps objects from the ResNet state space ($\mathbb{R}^{n_\ell}$) into the ResNet output space ($\mathbb{R}^d$). $L_{in} : \mathbb{R}^d \mapsto \mathbb{R}^{n_\ell}$ is a transformation, with Lipschitz constant K_{in}, that maps objects from the ResNet input space ($\mathbb{R}^d$) into the ResNet state space ($\mathbb{R}^{n_\ell}$). L has Lipschitz constant $K_L = K_{in} K_{out}$.

In the notation of (2.5), the transformation L is the action of W_{L-1} (=: L_{out}) and $\sigma(W_0 y_0 + b_0)$ (=: L_{in}). It is reasonable to assume that L is Lipschitz with Lipschitz constant $K_L = K_{in} K_{out}$. For instance, the action of W_{L-1} is linear and when σ is ReLU, L_{in} is Lipschitz. These transformations are shown in Fig. 4 where the evolution of an arbitrary initial condition through the ResNet state space is depicted.

Next, we rewrite the continuous ResNet (4.5) with input coming from (4.1). This will be crucial to compare NINNs with u solving (4.1).

$$\begin{aligned} &\partial_t v = f_{NN}(v(t)), \qquad \forall t \in (k\Delta t, (k+1)\Delta t), \\ &v(k\Delta t) = L_{in}(u(k\Delta t)), \\ &v(0) = u_0, \end{aligned} \tag{4.6}$$

with $v \in \mathbb{R}^{n_\ell}$ and L_{in} given in Definition 4.4. This is the best approximation of the dynamics (4.1) in the ResNet state space. It will be useful for the rest of the section to introduce the notation z^- to indicate a limit from the left. The following ResNet error term quantifies how good the approximation v is in comparison to the dynamics (4.1).

Definition 4.5 (*ResNet Error ϵ_{NN}*)**.** The ResNet error term ϵ_{NN} is the maximum value of $|L_{out}(v^-(k\Delta t)) - u(k\Delta t)|^2$ across all $k \in \mathbb{Z}^+$, where u and v are respectively given by (4.1) and (4.6).

We define the map $F_{NN} : \mathbb{R}^d \times [0, \Delta t] \to \mathbb{R}^{n_\ell}$ as follows. F_{NN} takes the input $x_0 \in \mathbb{R}^d$ and applies the trained continuous ResNet (Assumption 4.3) to produce the evolution in time. In other words,

$$F_{NN}(x_0, t) := h(t) = \int_0^t f_{NN}(h(s), s) ds + L_{in}(x_0). \tag{4.7}$$

This map is Lipschitz in the first component with Lipschitz constant K, we establish this next.

Theorem 4.6. *Assume that Assumption 4.3 holds. Recall f_{NN} and L_{in} are Lipschitz with Lipschitz constants $\widetilde{K}$ and K_{in} from Assumption 4.3 (iii) and*

[1] In this section we are considering ResNets that replicate dynamical systems. In this setting is it natural for the input and output dimensions to be equal. Our analysis can be easily extended to handle the case of additional input parameters.

Definition 4.4 respectively. Then the map $(x,t) \mapsto F_{NN}(x,t) := h$ *(cf. (4.7)) with* $x \in \mathbb{R}^d$ *denoting the input is Lipschitz in the first component with Lipschitz constant* $K = e^{\widetilde{K}\Delta t}K_{in}$.

Proof. Let $x_1, x_2 \in \mathbb{R}^d$. Lifting x_1, x_2 to $\mathbb{R}^{n_\ell}$ with L_{in} and denoting the two separate evolution's in (4.6) by h_1, h_2 with $\widetilde{h} = h_1 - h_2$, we obtain that

$$\partial_t \widetilde{h} = f_{NN}(h_1) - f_{NN}(h_2).$$

Multiply both sides by $\widetilde{h}$ and using the Lipschitz property of f_{NN},

$$\frac{1}{2}\partial_t|\widetilde{h}|^2 - \widetilde{K}|\widetilde{h}|^2 \le 0.$$

After integrating,

$$|\widetilde{h}(t)|^2 \le e^{2\widetilde{K}\Delta t}|\widetilde{h}(0)| = e^{2\widetilde{K}\Delta t}|L_{in}(x_1) - L_{in}(x_2)| \le e^{2\widetilde{K}\Delta t}K_{in}|x_1 - x_2|.$$

The proof is complete. □

We remind the reader that the analysis in this section takes into account a system of ResNets, such as in Fig. 2. To account for this, we will rewrite the ith (NINN #1) feedback control term as

$$g^i_\ell := (f_L \circ \dots \circ f_{\ell+1}(y_\ell) - y^{QoI})\frac{1}{\|W_{L-1}\|_1}W_{L-1} := \mathcal{N}_i z_{\ell_i},$$

with $\mathcal{N}_i := (f_L \circ \dots \circ f_{\ell+1}(y_\ell) - y^{QoI}) \in \mathbb{R}$ and $z_{\ell_i} := \frac{1}{\|W_{L-1}\|_1}W_{L-1} \in \mathbb{R}^{n_{\ell_i}}$. $\mathcal{N} \odot z \in \mathbb{R}^{n_\ell}$ will represent the (NINN #1) feedback terms for the whole ResNet system. Here $\mathcal{N} = [\mathcal{N}_1, \dots, \mathcal{N}_{N_R}]$ with $\mathcal{N}_i \in \mathbb{R}$ and $z = [z_{\ell_1}, \dots, z_{\ell_{N_R}}]$ with $z_{\ell_i} \in \mathbb{R}^{n_{\ell_i}}$. Moreover $\mathcal{N} \odot z$ is defined as $\mathcal{N} \odot z := [\mathcal{N}_1 z_{\ell_1} \dots \mathcal{N}_{N_R} z_{\ell_{N_R}}] \in \mathbb{R}^{n_\ell}$ where N_R is the number of ResNets in the system and $n_\ell = \sum_{i=1}^{N_R} n_{\ell_i}$ which is defined in Definition 4.1. If ResNet i does not have a NINN feedback term then we assume $\mathcal{N}_i z_{\ell_i} = 0 \in \mathbb{R}^{n_{\ell_i}}$. To be more specific, we will write

$$\mathcal{N}(t) = I_M(L_{out}[F_{NN}(w(t), (k+1)\Delta t - t)] - u(k\Delta t)), \quad \forall t \in (k\Delta t, (k+1)\Delta t).$$

Here $I_M(u(k\Delta t))$ is the QoI. Next, $I_M(L_{out}(F_{NN}(w(t), (k+1)\Delta t - t)))$ evolves $w(t)$ to $w((k+1)\Delta t)$ and produces output with L_{out}. Using Assumption 4.2 we can estimate $|\mathcal{N}|$,

$$\begin{aligned}|\mathcal{N}(t)| &= |I_M(L_{out}[F_{NN}(w(t), (k+1)\Delta t - t)] - u(k\Delta t))| \\ &\le (c_M + 1)|L_{out}(F_{NN}(w(t), (k+1)\Delta t - t)) - u(k\Delta t)| \\ &= c_{\mathcal{N}}|L_{out}(F_{NN}(w(t), (k+1)\Delta t - t)) - u(k\Delta t)|,\end{aligned} \tag{4.8}$$

for $t \in (k\Delta t, (k+1)\Delta t)$. By rescaling, z satisfies $|z| = 1$. We define the continuous NINN using (NINN #1) as described in Section 3.1 with partial observations $\{I_M(u(k\Delta t))\}_{k=0}^{\infty}$ occurring every Δt time units,

$$\partial_t w = f_{NN}(w) - \mu \chi_{\Omega_{QoI}}(\mathcal{N} \odot z), \qquad \forall t \in (k\Delta t, (k+1)\Delta t), \tag{4.9}$$

$$w(k\Delta t) = L(\lim_{t \to k\Delta t^-} w(t)),$$

$$w(0) = L_{in}(w_0).$$

Next we add and subtract $\widetilde{w} = v - w$ to (4.9) and define $\epsilon := \mathcal{N} \odot z + \widetilde{w}$. This gives us the alternate formulation of (4.9),

$$\partial_t w = f_{NN}(w) - \mu \chi_{\Omega_{QoI}}(w - [v + \epsilon]), \qquad \forall t \in (k\Delta t, (k+1)\Delta t), \tag{4.10}$$

$$w(k\Delta t) = L(\lim_{t \to k\Delta t^-} w(t)),$$

$$w(0) = L_{in}(w_0).$$

We will need the following metric G in the proof which represents the cost of translating between the state space of the dynamics and the state space of the ResNet (see Fig. 4).

Definition 4.7 (*G-metric*). Let $G := \max_k |L_{out} \circ L_{in}(u(k\Delta t)) - u(k\Delta t)|$ with $k \in \mathbb{Z}^+$.

Before, we prove our main results we introduce an assumption on $\chi_{\Omega_{QoI}}$ defined in (4.2).

Assumption 4.8 (*Stability of* $\chi_{\Omega_{QoI}}$). Let v be as given in (4.6) and let w represent the continuous NINNs solution (4.9). If $\widetilde{w}(t) = v(t) - w(t)$, then we assume that for all $t \in [0, \infty)$, $\exists\, \alpha \ge 0$ such that $|\chi_{\Omega_{QoI}}(\widetilde{w}(t)) - \widetilde{w}(t)| \le \alpha|\widetilde{w}(t)|$ with $0 \le \alpha < 1$.

The above assumption is likely to hold. Assume $\alpha = 1$ is the smallest α such that the inequality in Assumption 4.8 holds. Since α is smallest, it follows that $|\chi_{\Omega_{QoI}}(\widetilde{w}(t^*)) - \widetilde{w}(t^*)| = |\widetilde{w}(t^*)|$ for some $t^* \in [0, \infty)$. This in turn implies $\chi_{\Omega_{QoI}}(\widetilde{w}(t^*)) = 0$. The equality holds from the definition of $\chi_{\Omega_{QoI}}$ in (4.2). Next, we can infer from $\chi_{\Omega_{QoI}}(\widetilde{w}(t^*)) = 0$ that $[\widetilde{w}(t^*)]_i = 0$ for all $i \in \Omega_{QoI}$ which is unlikely to occur given the presence of neural network error, NINN error and incomplete observations.

Lemma 4.9. *Assume that Assumptions 4.2 and 4.3 hold. Let* w *satisfy (4.9),* v *satisfy (4.6) and* u *satisfy (4.1). Recall Lipschitz constant* K_{out} *from Definition 4.4. Then* $\epsilon := \mathcal{N} \odot z + \widetilde{w}$ *satisfies the following bound for* $t \in (k\Delta t, (k+1)\Delta t)$,

$$|\epsilon| \le |\mathcal{N}|\Big(1 + \frac{1}{c_{\mathcal{N}}K_{out}}\Big) + \epsilon_{2,w}(\Delta t) + \frac{G}{K_{out}}, \tag{4.11}$$

where the error term $\epsilon_{2,w} \to 0$ *as* $\Delta t \to 0$.

Proof. Recall $|z| = 1$ which implies $|\mathcal{N} \odot z| \le |\mathcal{N}|$. Then we have

$$\begin{aligned}|\epsilon| &= |\mathcal{N} \odot z + \widetilde{w}| \\ &\le \Big|\mathcal{N} \odot z + \frac{|\mathcal{N} \odot z|\widetilde{w}}{c_{\mathcal{N}}K_{out}|\widetilde{w}|}\Big| + \Big|\frac{|\mathcal{N} \odot z|\widetilde{w}}{c_{\mathcal{N}}K_{out}|\widetilde{w}|} - \widetilde{w}\Big| \\ &\le |\mathcal{N}|(1 + \frac{1}{c_{\mathcal{N}}K_{out}}) + \frac{\|\mathcal{N} \odot z| - c_{\mathcal{N}}K_{out}|\widetilde{w}\|}{c_{\mathcal{N}}K_{out}}.\end{aligned} \tag{4.12}$$

The second term in the above estimate is handled next. Let $t \in (k\Delta t, (k+1)\Delta t)$ and recall the estimate (4.8).

$$\begin{aligned}|(\mathcal{N} \odot z)(t)| \le |\mathcal{N}(t)| &\le c_{\mathcal{N}}|L_{out}(F_{NN}(w(t), (k+1)\Delta t - t)) - u(k\Delta t)| \\ &= c_{\mathcal{N}}|L_{out}(w) - L_{out}(w) + L_{out}(v) - L_{out}(v) \\ &\quad + L_{out}(F_{NN}(w(t), (k+1)\Delta t - t)) - u(k\Delta t)| \\ &\le c_{\mathcal{N}}K_{out}|F_{NN}(w(t), (k+1)\Delta t - t) - w(t)| \\ &\quad + c_{\mathcal{N}}|L_{out}(v) - u(k\Delta t)| + c_{\mathcal{N}}K_{out}|\widetilde{w}| \\ &\le c_{\mathcal{N}}K_{out}|F_{NN}(w(t), (k+1)\Delta t - t) - w(t)| \\ &\quad + c_{\mathcal{N}}|L_{out}(v) - L_{out} \circ L_{in}(u(k\Delta t))| \\ &\quad + c_{\mathcal{N}}|L_{out} \circ L_{in}(u(k\Delta t)) - u(k\Delta t)| + c_{\mathcal{N}}K_{out}|\widetilde{w}|.\end{aligned}$$

This implies

$$\begin{aligned}\Big||\mathcal{N} \odot z| - c_{\mathcal{N}}K_{out}|\widetilde{w}|\Big| &\le c_{\mathcal{N}}K_{out}|F_{NN}(w(t), (k+1)\Delta t - t) - w(t)| \\ &\quad + c_{\mathcal{N}}K_{out}|v - L_{in}(u(k\Delta t))| \\ &\quad + c_{\mathcal{N}}|L_{out} \circ L_{in}(u(k\Delta t)) - u(k\Delta t)|.\end{aligned}$$

Notice the first two terms on the right hand side tend towards 0 as $\Delta t \to 0$. To see this, the first term is equal to

$$c_{\mathcal{N}}K_{out}|F_{NN}(w(t), (k+1)\Delta t - t) - w(t)| = c_{\mathcal{N}}K_{out}\Big|\int_0^{(k+1)\Delta t - t} f_{NN}\,ds\Big|,$$

by (4.7). Then recall that $t \in (k\Delta t, (k+1)\Delta t)$. Thus the above right-hand-side goes to zero as $\Delta t \to 0$. Similarly, the second term is equal to

$$\begin{aligned}c_{\mathcal{N}}K_{out}|v(t) - L_{in}(u(k\Delta t))| &= c_{\mathcal{N}}K_{out} \\ &\quad \times |F_{NN}(L_{in}(u(k\Delta t)), t - \Delta t) - L_{in}(u(k\Delta t))| \\ &= c_{\mathcal{N}}K_{out}\Big|\int_0^{t-\Delta t} f_{NN}\,ds\Big|,\end{aligned}$$

which also goes to zero as $\Delta t \to 0$. Recall f_{NN} is Lipschitz by Assumption 4.3, therefore it is continuous and both integrals converge to 0 as $\Delta t \to 0$. Let us return to estimate (4.12) and recall G from Definition 4.7. Then an appropriate bound is

$$|\epsilon| \leq |\mathcal{N}|\Big(1 + \frac{1}{c_{\mathcal{N}} K_{out}}\Big) + \epsilon_{2,w}(\Delta t) + \frac{G}{K_{out}}. \tag{4.13}$$

This concludes the proof. □

Next we prove an error estimate for (NINN #1).

Theorem 4.10. *Assume that Assumptions 4.2, 4.3 and 4.8 hold. Let μ in (4.9) satisfy $\mu > \max\{\frac{2\ln(2K_L^2)}{(1-\alpha)\Delta t}, \frac{4\widetilde{K}}{1-\alpha}\}$. Recall K_L is given in Definition 4.4 and ϵ_{NN} is given in Definition 4.5. Then for u and w satisfying (4.1) and (4.10) respectively, the following estimate holds*

$$|L_{out}(w^-(k\Delta t)) - u(k\Delta t)|^2 \leq 4\epsilon_{NN} + \frac{8K_{out}^2}{(1-\alpha)^2} \max_t |\epsilon|^2 + 2^{-k}|w_0 - u_0|^2.$$

Proof. Define $\widetilde{w} = v - w$ and assume $t \in (k\Delta t, (k+1)\Delta t)$. Then from (4.6) and (4.10) it is easy to see that

$$\partial_t \widetilde{w} = f_{NN}(v) - f_{NN}(w) - \mu \chi_{\Omega_{QoI}}(\widetilde{w}) - \mu \chi_{\Omega_{QoI}}(\epsilon).$$

Multiply both sides by $\widetilde{w}$,

$$\frac{1}{2}\partial_t|\widetilde{w}|^2 + \mu|\widetilde{w}|^2 = (f_{NN}(v) - f_{NN}(w), \widetilde{w})$$
$$-\mu(\chi_{\Omega_{QoI}}(\widetilde{w}) - \widetilde{w}, \widetilde{w}) - \mu(\chi_{\Omega_{QoI}}(\epsilon), \widetilde{w}). \tag{4.14}$$

Each term on the right hand side is estimated next. Recall (4.4), Young's inequality, and we immediately obtain that

$$|(f_{NN}(v) - f_{NN}(w), \widetilde{w})| \leq |f_{NN}(v) - f_{NN}(w)||\widetilde{w}| \leq \widetilde{K}|\widetilde{w}|^2.$$

$$\mu|(\chi_{\Omega_{QoI}}(\epsilon), \widetilde{w})| \leq \mu|\epsilon||\widetilde{w}| \leq \frac{\mu}{2(1-\alpha)}|\epsilon|^2 + \frac{(1-\alpha)\mu}{2}|\widetilde{w}|^2.$$

Next, using 4.8, we obtain that

$$\mu|(\chi_{\Omega_{QoI}}(\widetilde{w}) - \widetilde{w}, \widetilde{w})| \leq \mu|\chi_{\Omega_{QoI}}(\widetilde{w}) - \widetilde{w}||\widetilde{w}| \leq \mu\alpha|\widetilde{w}|^2.$$

Substituting the above estimates in (4.14), we obtain that

$$\frac{1}{2}\partial_t|\widetilde{w}|^2 + \Big(\frac{(1-\alpha)\mu}{2} - \widetilde{K}\Big)|\widetilde{w}|^2 \leq \frac{\mu}{2(1-\alpha)}|\epsilon|^2.$$

After choosing μ large enough to satisfy $\frac{(1-\alpha)\mu}{2} - \widetilde{K} > \frac{(1-\alpha)\mu}{4}$ or $\mu > \frac{4\widetilde{K}}{(1-\alpha)}$,

$$\partial_t|\widetilde{w}|^2 + \frac{(1-\alpha)\mu}{2}|\widetilde{w}|^2 \leq \frac{\mu}{(1-\alpha)}|\epsilon|^2. \tag{4.15}$$

Recall v and w from (4.6) and (4.10) at $t = k\Delta t$. In particular, we have that $\widetilde{w}(k\Delta t) = v(k\Delta t) - w(k\Delta t) = L_{in}(u(k\Delta t)) - L(w^-(k\Delta t))$. We use the notation z^- to indicate a limit from the left. Integrating (4.15), from $k\Delta t$ to $(k+1)\Delta t$, we obtain that

$$\begin{aligned}|\widetilde{w}^-((k+1)\Delta t)|^2 &\leq \frac{2}{(1-\alpha)^2}\max_t|\epsilon|^2 + e^{-(1-\alpha)\mu\Delta t/2}|\widetilde{w}(k\Delta t)|^2 \\ &= \frac{2}{(1-\alpha)^2}\max_t|\epsilon|^2 + e^{-(1-\alpha)\mu\Delta t/2} \\ &\quad \times |L(w^-(k\Delta t)) - L_{in}(u(k\Delta t))|^2. \\ &\leq \frac{2}{(1-\alpha)^2}\max_t|\epsilon|^2 + e^{-(1-\alpha)\mu\Delta t/2}K_{in}^2 \\ &\quad \times |L_{out}(w^-(k\Delta t)) - u(k\Delta t)|^2,\end{aligned} \tag{4.16}$$

where in the last step we have used that $L = L_{in} \circ L_{out}$. By the triangle inequality and Definition 4.4 we have for all k,

$$\begin{aligned}|L_{out}(w^-(k\Delta t)) - u(k\Delta t)| &\leq K_{out}|\widetilde{w}^-(k\Delta t)| + |L_{out}(v^-(k\Delta t)) - u(k\Delta t)| \\ &\leq K_{out}|\widetilde{w}^-(k\Delta t)| + (\epsilon_{NN})^{1/2}.\end{aligned} \tag{4.17}$$

By Young's inequality (4.17) becomes,

$$|L_{out}(w^-(k\Delta t)) - u(k\Delta t)|^2 \leq 2K_{out}^2|\widetilde{w}^-(k\Delta t)|^2 + 2\epsilon_{NN}. \tag{4.18}$$

The first term has been estimated in (4.16). The second term is a ResNet error which we have denoted by ϵ_{NN}, see Definition 4.5. Choose μ such that $e^{-(1-\alpha)\mu\Delta t/2}K_L^2 < \frac{1}{2}$, recall $K_L = K_{in}K_{out}$ from Definition 4.4. This results in $\mu > \frac{2\ln(2K_L^2)}{(1-\alpha)\Delta t}$. Next, we insert the (4.16) estimate into the right hand side of (4.18),

$$\begin{aligned}|L_{out}(w^-((k+1)\Delta t)) - u((k+1)\Delta t)|^2 &\leq 2\epsilon_{NN} + \frac{4K_{out}^2}{(1-\alpha)^2}\max_t|\epsilon|^2 \\ &\quad + \frac{1}{2}|L_{out}(w^-(k\Delta t)) - u(k\Delta t)|^2.\end{aligned}$$

After applying this estimate recursively,

$$\begin{aligned}|L_{out}(w^-(k\Delta t)) - u(k\Delta t)|^2 &\leq \frac{1-2^{-k}}{1-2^{-1}}\Big(2\epsilon_{NN} + \frac{4K_{out}^2}{(1-\alpha)^2}\max_t|\epsilon|^2\Big) \\ &\quad + 2^{-k}|w_0 - u_0|^2 \\ &\leq 4\epsilon_{NN} + \frac{8K_{out}^2}{(1-\alpha)^2}\max_t|\epsilon|^2 + 2^{-k}|w_0 - u_0|^2.\end{aligned}$$

This concludes the proof. □

4.1. Error summary

The estimate in Theorem 4.10 consist of three terms. The first term, ϵ_{NN} (Definition 4.5), is expected to be small in practice as the user has freedom over which ResNet is used. The middle terms consist of $\epsilon = \mathcal{N}_\ell \mu_W + \widetilde{w}$ and is estimated by (4.11). This error can be minimized by choosing ResNets with Δt and G small. The third term is the standard exponential decay term that arises in nudging error estimates.

4.2. Discrete dynamics and convergence estimates

In the previous section we proved convergence error results for the continuous NINNs described in (4.9). Notice, that the discrete NINNs have ResNet structure (2.5). To derive the estimates in the discrete setting, we can simply use a triangle inequality. Let u represent the true solution (4.1), w_{cont} represent the continuous NINN solution and let w represent the time discrete NINN solution. Then for all $k \in \mathbb{Z}^+$,

$$\begin{aligned}\| L_{out}(w^-(k\Delta t)) - u(k\Delta t) \| \leq & \\ &\|L_{out}(w^-(k\Delta t)) - L_{out}(w_{cont}^-(k\Delta t))\| \\ &+ \|L_{out}(w_{cont}^-(k\Delta t)) - u(k\Delta t)\|.\end{aligned}$$

The first term on the right-hand-side is the time discretization error for DNN, for instance, forward Euler. The approximation of the second term has been discussed in Theorem 4.10.

5. Experimental introduction

The purpose of this section is to set a stage for our numerical experiments. Section 5.1 details how the training is done for the ResNets in the experiments. Section 5.2 details the setup required for the data assimilation experiments and the evaluation metric. Section 5.3 covers a technique useful when applying nudging algorithms. The following sections contain the numerical experiments. In Section 6, NINNs are implemented as data assimilation algorithms, where we test the algorithms on the chaotic Lorenz 63 and 96 ODE systems. In Section 7, NINNs aid ResNets in replicating stiff ODEs arising from chemically reacting flows.

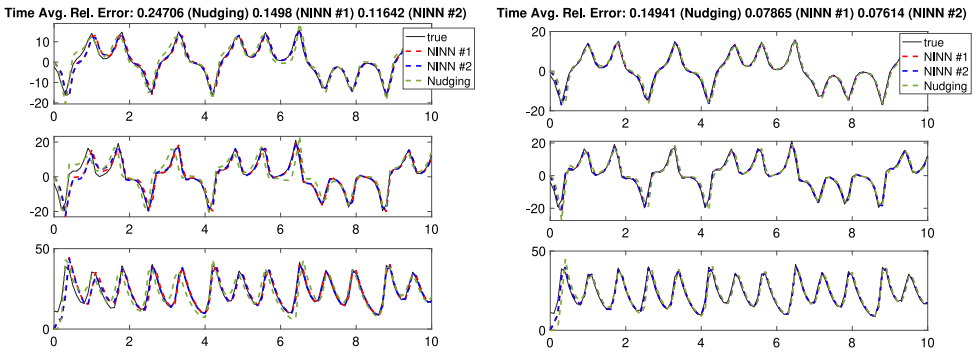

Fig. 5. Left: x-component observations. Right: y-component observations. From top to bottom the x, y, z components of the reference (true), (NINN #1), (NINN #2), and nudging solutions over 10 time units for the Lorenz 63 model.

5.1. ResNet training

The ResNet systems in Sections 6 and 7 are trained using the specifications in this section. The systems will be comprised of multiple ResNets where each ResNet has output in $\mathbb{R}$ and is responsible for one component of the output. Fig. 2 is one such example for a system with input/output in $\mathbb{R}^3$. Additionally, the ResNet systems will be trained on 15,000 training samples. The training samples are input/output pairs generated from a dynamical system corresponding to a specific time step. Specifically, many random initial conditions are generated and propagated forward using the dynamical system. In Section 6, the time step size for the Lorenz ODEs is 10^{-2} and in Section 7 the time step size is $5 \cdot 10^{-8}$ for the chemically reacting flow ODEs. The training samples will be split 80–20, i.e. 80% of the samples will be used for training and 20% will be used for validation. A patience of 400 iterations is used with the training data. The latter means that if the validation error increases then training will continue for 400 more iterations. The BFGS optimization routine is used in conjunction with bias ordering, see Section 2.2 for the details. The parameters are initialized with box initialization [26].

5.2. Data assimilation protocol

In Section 6, ResNets are trained to learn the Lorenz 63 and 96 ODE systems. We generate synthetic partial observations and, by equipping the ResNets with NINNs, recover the solution corresponding to the partial observations. Therefore showing NINNs are effective as data assimilation algorithms. The time averaged relative error will be used to measure the convergence and is computed as the approximation of the integral,

$$\frac{1}{T}\int_0^T \frac{\|x^{ALG}(t) - x^{ref}(t)\|}{\|x^{ref}(t)\|}\,dt. \tag{5.1}$$

Here x^{ALG} represents the algorithm we are computing the error for, (NINN #1), (NINN #2), or nudging (2.1). The goal is to drive (5.1) towards zero by combining the ResNet model and the observations of x^{ref} using NINNs.

5.3. Exponential decay

As stated in Section 5.1, the ResNets in Section 6 are given training samples corresponding to a time step size of 10^{-2}. As stated in below in Section 6, the synthetic observations are available every 10^{-1} time units. Therefore, NINNs compute 10 ResNet evaluations before the observations are updated again. During this time, the observation becomes outdated as more ResNet evaluations are done, advancing in time. We dampen the μ parameter between observations with $e^{-k\Lambda}$ for $k = 0, 1, \ldots, 9$, i.e. $\mu \to \mu e^{-k\Lambda}$. For the experiments in Section 6 we use an exponential decay factor of $\Lambda = 1$.

6. Data assimilation

In this section we apply NINNs to ResNets which have learned the Lorenz 63 and 96 ODEs. (NINN #1) and (NINN #2) will be compared as data assimilation algorithms on the Lorenz ODEs.

6.1. Lorenz 63

The Lorenz 63 model is given by the three coupled ODEs

$$d_t x = \sigma(y - x), \tag{6.1a}$$

$$d_t y = x(\rho - z) - y, \tag{6.1b}$$

$$d_t z = xy - \beta z. \tag{6.1c}$$

We set $\sigma = 10$, $\beta = 8/3$, $\rho = 28$, which is known to exhibit chaotic behavior [27]. The Eqs. (6.1a)–(6.1c) are solved using an explicit Runge–Kutta (4,5) in MATLAB. The training data is generated from 1000 different initial conditions. The initial conditions are generated from three i.i.d. normal distributions with mean 0 and standard deviation of 10. The initial conditions are propagated forward to 110 time units using the ODEs (6.1a)–(6.1c), and observations are selected every $\delta = 0.1$ time units starting at 100 time units. The ResNets trained on the training data have 8 layers each with a width of 15. We will focus on two types of observations, x component only and y component only observations. In Fig. 5, we compute solutions for the two NINNs and nudging on a randomly generated reference solution that is outside of the training set. The initial condition for the reference solution is generated from three i.i.d. normal distributions with mean 0 and standard deviation of 50. For (NINN #2), we found it was only necessary to control the output of the ResNets. This corresponds to $g_0 = 0$ and g_L as defined in (NINN #2). Both NINNs solutions are performing better than nudging based on the time averaged relative error provided on Fig. 5.

6.2. Lorenz 96

The Lorenz 96 model is given by the following set of ODEs:

$$d_t x_i = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \quad i = 1, 2, \ldots, 40, \tag{6.2}$$

$$x_{-1} = x_{39}, \quad x_0 = x_{40}, \quad \text{and} \quad x_{41} = x_1. \tag{6.3}$$

We set $F = 10$ which is known to exhibit chaotic behavior [28]. We follow a similar procedure as with the Lorenz 63 model and we consider two different observation patterns. We observe approximately 33%, and 50% of the state which corresponds to observing 13 and 20 components, respectively. Specifically, we observe every third component and every even component, respectively. The ResNets are trained using the same setup from the Lorenz 63 section with one major difference. We use the structure of (6.2) to reduce the input for the ResNets to $\mathbb{R}^4$ from $\mathbb{R}^{40}$. The ResNet corresponding to the ith component of the

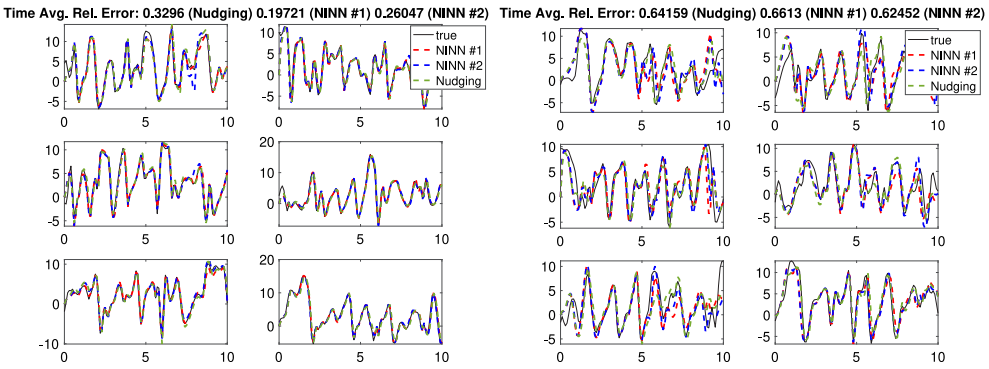

Fig. 6. Left: 20 observations on even components. Right: 13 observations on every third component. From left to right, top to bottom the first 6 components of the reference (true), (NINN #1), (NINN #2), and nudging solutions over 10 time units for the Lorenz 96 model.

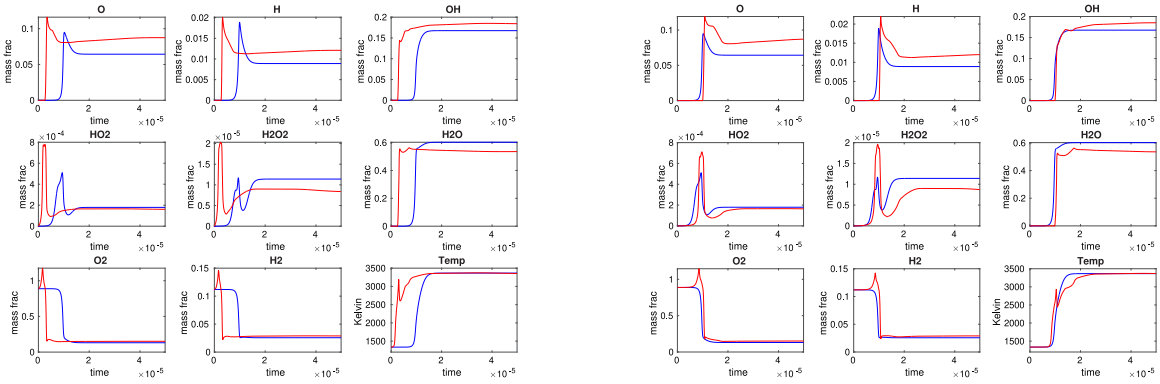

Fig. 7. Left: No nudging. Right: Nudging towards initial data. Initial temperature 1336 K. Blue is the CHEMKIN solution and red is the ResNet solution.

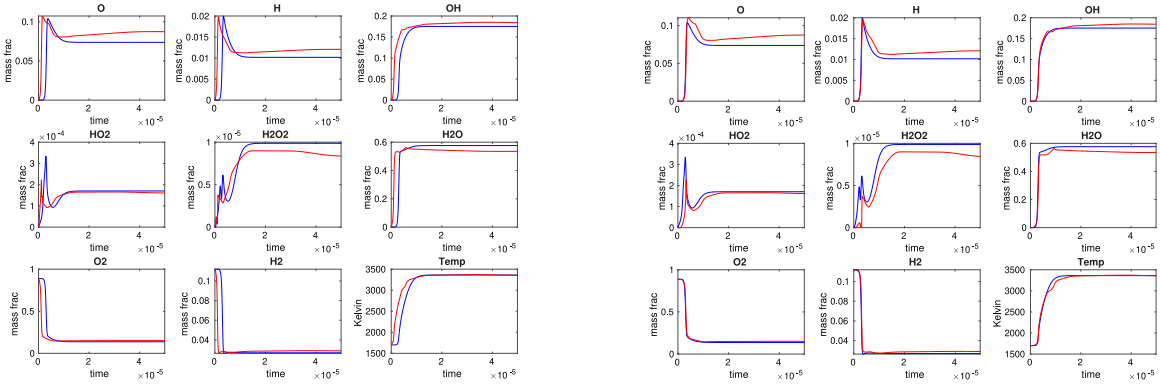

Fig. 8. Left: No nudging. Right: Nudging towards initial data. Initial temperature 1700 K. Blue is the CHEMKIN solution and red is the ResNet solution.

state vector takes in as input the $i-2, i-1, i, i+1$ components of the state vector. We refer to these ResNets as being reduced as their input size is 4 compared to the state space size of 40. The ResNet used in the experiments has 9 layers each with a width of 15. In Fig. 6, we compute solutions for the two NINNs and nudging on a randomly generated reference solution. For (NINN #2), we found it was necessary to control the input and output of the ResNets. This corresponds to g_0 and g_L as defined in (NINN #2). For this particular reference solution, all three algorithms are performing similarly based on the time averaged relative error provided on Fig. 6.

7. Chemical kinetics

The purpose of this section is to demonstrate the effectiveness of NINNs in improving pre-existing neural networks. In particular we show how NINNs can be used to aid neural networks designed to learn ODEs describing chemically reacting flows. We present an experiment where ResNets learn one time step of a stiff ODE modeling a reduced $H_2 - O_2$ reaction. The model tracks the reactions of eight species and temperature over time, for more details on the model we refer the reader to the Ref. [29]. For more information approximating the model with ResNets see the recent works [23,24]. The training data for the ResNets is generated by CHEMKIN [30]. For each species and temperature there corresponds a ResNet for a total of 9 ResNets. The training data is generated from initial conditions with an equivalence ratio of one, $H_2 = \frac{2}{3}$ mol and $O_2 = \frac{1}{3}$ mol. In addition, the temperature ranges from 1300 to 2500 Kelvin in increments of 12 K. Training data is collected every $5 \cdot 10^{-8}$ time units with 1000 samples per solution. The ResNets used in this example have 7 hidden layers with a width of 30. This particular ResNet system is able to replicate the dynamics well for temperatures above 1600 K and struggles with capturing the dynamics for temperatures below 1600 K with 1300 K being the worst.

To improve the ResNet accuracy we can introduce NINNs. As we are given the exact initial data, we can nudge each of the 9 ResNets towards the initial data for a period of time initially. This nudge corrects the ResNet in the lower temperature region. See Figs. 7 and 8. For this example we used (NINN #1).

8. Conclusions

This paper has introduced nudging induced neural networks (NINNs). NINNs can be implemented onto pre-existing neural networks which allow the user to control the neural network given observations or a quantity of interest. We demonstrated how NINNs are effective as data assimilation algorithms on the Lorenz 63 and Lorenz 96 ODEs. NINNs were able to outperform the classical nudging data assimilation algorithm in our experiments while retaining the ease of computation found in neural networks. One drawback of NINNs in comparison to standard data assimilation methods is the requirement of training data for the neural network. We demonstrated the uses of NINNs in replacing stiff ODE dynamics with a neural network. The convergence analysis gave us insight into the errors involved with NINNs and how to minimize them.

CRediT authorship contribution statement

Harbir Antil: Conceptualization, Data curation, Funding acquisition, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Rainald Löhner:** Conceptualization, Data curation, Funding acquisition, Methodology, Project administration, Resources, Software, Supervision, Writing – review & editing. **Randy Price:** Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Harbir Antil reports financial support was provided by Defense Threat Reduction Agency. Harbir Antil reports financial support was provided by National Science Foundation. Harbir Antil reports financial support was provided by Air Force Office of Scientific Research.

Data availability

Data will be made available on request.

References

[1] E. Haber, L. Ruthotto, Stable architectures for deep neural networks, Inverse Problems 34 (1) (2018) 014004, 22, http://dx.doi.org/10.1088/1361-6420/aa9a90.

[2] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[3] L. Ruthotto, E. Haber, Deep neural networks motivated by partial differential equations, J. Math. Imaging Vision (2019) http://dx.doi.org/10.1007/s10851-019-00903-1.

[4] W. E, A proposal on machine learning via dynamical systems, Commun. Math. Stat. 5 (1) (2017) 1–11, http://dx.doi.org/10.1007/s40304-017-0103-z.

[5] H. Antil, H.C. Elman, A. Onwunta, D. Verma, A deep neural network approach for parameterized pdes and bayesian inverse problems, Machine Learning: Science and Technology 4 (3) (2023) 035015.

[6] H. Antil, R. Khatri, R. Löhner, D. Verma, Fractional deep neural network via constrained optimization, Mach. Learn.: Sci. Technol. 2 (1) (2020) 015003.

[7] R.A. Anthes, Data assimilation and initialization of hurricane prediction models, J. Atmos. Sci. 31 (3) (1974) 702–719, http://dx.doi.org/10.1175/1520-0469(1974)031<0702:DAAIOH>2.0.CO;2, https://journals.ametsoc.org/view/journals/atsc/31/3/1520-0469_1974_031_0702_daaioh_2_0_co_2.xml.

[8] D. Auroux, J. Blum, A nudging-based data assimilation method: the Back and Forth Nudging (BFN) algorithm, Nonlinear Process. Geophys. 15 (2) (2008) 305–319, http://dx.doi.org/10.5194/npg-15-305-2008, https://npg.copernicus.org/articles/15/305/2008/.

[9] J.E. Hoke, R.A. Anthes, The initialization of numerical models by a dynamic-initialization technique, Mon. Weather Rev. 104 (12) (1976) 1551–1556, http://dx.doi.org/10.1175/1520-0493(1976)104<1551:TIONMB>2.0.CO;2, https://journals.ametsoc.org/view/journals/mwre/104/12/1520-0493_1976_104_1551_tionmb_2_0_co_2.xml.

[10] C. Le Provost, J. Verron, E. Blayo, J. Molines, B. Barnier, Assimilation of Topex/Poseidon altimeter data into a circulation model of the North Atlantic, in: Proceedings of OCEANS'94, Vol. 3, 1994, pp. III/63–III/68, http://dx.doi.org/10.1109/OCEANS.1994.364173.

[11] H. Nijmeijer, A dynamical control view on synchronization, Phys. D 154 (3–4) (2001) 219–228, http://dx.doi.org/10.1016/S0167-2789(01)00251-2.

[12] D.R. Stauffer, N.L. Seaman, Use of four-dimensional data assimilation in a limited-area mesoscale model. Part I: Experiments with synoptic-scale data, Mon. Weather Rev. 118 (6) (1990) 1250–1277, http://dx.doi.org/10.1175/1520-0493(1990)118<1250:UOFDDA>2.0.CO;2, https://journals.ametsoc.org/view/journals/mwre/118/6/1520-0493_1990_118_1250_uofdda_2_0_co_2.xml.

[13] D.A.F. Albanez, H.J. Nussenzveig Lopes, E.S. Titi, Continuous data assimilation for the three-dimensional Navier–Stokes-α model, Asymptot. Anal. 97 (1–2) (2016) 139–164, http://dx.doi.org/10.3233/ASY-151351.

[14] H. Bessaih, E. Olson, E.S. Titi, Continuous data assimilation with stochastically noisy data, Nonlinearity 28 (3) (2015) 729–753, http://dx.doi.org/10.1088/0951-7715/28/3/729.

[15] A. Biswas, R. Price, Continuous data assimilation for the three-dimensional Navier–Stokes equations, SIAM J. Math. Anal. 53 (6) (2021) 6697–6723.

[16] A. Farhat, M.S. Jolly, E.S. Titi, Continuous data assimilation for the 2D Bénard convection through velocity measurements alone, Phys. D 303 (2015) 59–66, http://dx.doi.org/10.1016/j.physd.2015.03.011.

[17] A. Farhat, E. Lunasin, E.S. Titi, Data assimilation algorithm for 3D Bénard convection in porous media employing only temperature measurements, J. Math. Anal. Appl. 438 (1) (2016) 492–506, http://dx.doi.org/10.1016/j.jmaa.2016.01.072.

[18] A. Farhat, E. Lunasin, E. Titi, On the charney conjecture of data assimilation employing temperature measurements alone: The paradigm of 3D planetary geostrophic model, Math. Clim. Weather Forecast. 2 (2016).

[19] A. Farhat, E. Lunasin, E.S. Titi, A data assimilation algorithm: the paradigm of the 3D Leray-α model of turbulence, in: Partial Differential Equations Arising from Physics and Geometry, in: London Math. Soc. Lecture Note Ser., vol. 450, Cambridge Univ. Press, Cambridge, 2019, pp. 253–273.

[20] C. Foias, C.F. Mondaini, E.S. Titi, A discrete data assimilation scheme for the solutions of the two-dimensional Navier–Stokes equations and their statistics, SIAM J. Appl. Dyn. Syst. 15 (4) (2016) 2109–2142.

[21] P.A. Markowich, E.S. Titi, S. Trabelsi, Continuous data assimilation for the three-dimensional Brinkman-Forchheimer-extended Darcy model, Nonlinearity 29 (4) (2016) 1292–1328, http://dx.doi.org/10.1088/0951-7715/29/4/1292.

[22] H. Antil, R. Löhner, R. Price, Data assimilation with deep neural nets informed by nudging, in: Reduction, Approximation, Machine Learning, Surrogates, Emulators and Simulators: RAMSES, Springer, 2024, pp. 17–41.

[23] T.S. Brown, H. Antil, R. Löhner, F. Togashi, D. Verma, Novel dnns for stiff odes with applications to chemically reacting flows, in: International Conference on High Performance Computing, Springer, 2021, pp. 23–39.

[24] H. Antil, T.S. Brown, R. Löhner, F. Togashi, D. Verma, Deep neural nets with fixed bias configuration, Numerical Algebra, Control and Optimization (2022).

[25] H. Antil, M. Gupta, R. Price, A note on dimensionality reduction in deep neural networks using empirical interpolation method, 2023, arXiv preprint arXiv:2305.09842.

[26] E.C. Cyr, M.A. Gulian, R.G. Patel, M. Perego, N.A. Trask, Robust training and initialization of deep neural networks: An adaptive basis viewpoint, in: Mathematical and Scientific Machine Learning, PMLR, 2020, pp. 512–536.

[27] E.N. Lorenz, Deterministic nonperiodic flow, J. Atmos. Sci. 20 (2) (1963) 130–141, http://dx.doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

[28] E. Lorenz, Predictability: a problem partly solved, in: Seminar on Predictability, 4-8 September 1995, Vol. 1, ECMWF, ECMWF, Shinfield Park, Reading, 1995, pp. 1–18, https://www.ecmwf.int/node/10829.

[29] E.L. Petersen, R.K. Hanson, Reduced kinetics mechanisms for ram accelerator combustion, J. Propuls. Power 15 (4) (1999) 591–600.

[30] R. Design, Chemkin, 2000, URL http://www.reactiondesign.com/products/chemkin.