Abstract—Deep neural networks (DNNs) training can be difficult due to vanishing and exploding gradients during weight optimization through backpropagation. To address this problem, we propose a general class of Hamiltonian DNNs (H-DNNs) that stem from the discretization of continuous-time Hamiltonian systems and include several existing DNN architectures based on ordinary differential equations. Our main result is that a broad set of H-DNNs ensures nonvanishing gradients by design for an arbitrary network depth. This is obtained by proving that, using a semi-implicit Euler discretization scheme, the backward sensitivity matrices involved in gradient computations are symplectic. We also provide an upper bound to the magnitude of sensitivity matrices that show that exploding gradients can be controlled through regularization. The good performance of H-DNNs is demonstrated on benchmark classification problems, including image classification with the MNIST dataset.

Index Terms—Deep neural networks (DNNs), Hamiltonian systems, ordinary differential equations (ODE) discretization.

I. INTRODUCTION

Deep learning has achieved remarkable success in various fields, such as computer vision, speech recognition, and natural language processing [1], [2]. Within the control community, there is also a growing interest in using deep neural networks (DNNs) to approximate complex controllers [3], [4]. In spite of recent progress, the training of DNNs still presents several challenges such as the occurrence of vanishing or exploding gradients during training based on gradient descent. These phenomena are related to the convergence to zero or the divergence, respectively, of the backward sensitivity matrices (BSMs) arising in gradient computations through backpropagation.

Both situations are very critical as they imply that the learning process either stops prematurely or becomes unstable [5].

Heuristic methods for dealing with these problems leverage subtle weight initialization or gradient clipping [5]. More recent approaches focus on the study of DNN architectures and associated training algorithms for which exploding/vanishing gradients can be avoided or mitigated by design [6], [7], [8], [9], [10], [11], [12], [13]. For instance, in [6] and [8], unitary and orthogonal weight matrices are used to control the magnitude of BSMs during backpropagation. Moreover, in [7] and [10], methods based on clipping singular values of weight matrices are utilized to constrain the magnitude of BSMs. These approaches, however, require expensive computations during training [7], [10], introduce perturbations in gradient descent [7], [10], or use restricted classes of weight matrices [6], [8].

Recently, it has been argued that specific classes of DNNs stemming from the time discretization of ordinary differential equations (ODEs) are less affected by vanishing and exploding gradients [11], [12], [13]. The arguments provided in [11] rely on the stability properties of the underlying continuous-time nonlinear systems for characterizing relevant behaviors of the corresponding DNNs obtained after discretization. Specifically, it has been suggested that the problem of vanishing gradients might be less relevant for DNN architectures based on dynamical systems that are marginally stable, i.e., that produce bounded and nonvanishing state trajectories. An example is provided by first-order ODEs based on skew-symmetric maps, which have been used in [11] and [12] for defining antisymmetric DNNs. Another example is given by dynamical systems in the form

\[
\dot{p} = -\nabla_q H(p, q), \quad \dot{q} = \nabla_p H(p, q)
\]

where \(p, q \in \mathbb{R}^n\) and \(H(\cdot, \cdot)\) is a Hamiltonian function. This class of ODEs has motivated the development of Hamiltonian-inspired DNNs in [11], whose effectiveness has been shown in several benchmark classification problems [11], [12], [14].

However, these approaches consider only restricted classes of weight matrices or particular Hamiltonian functions, which, together with the specific structure of the dynamics in (1), limit the representational power of the resulting DNNs. Moreover, the behavior of BSMs arising in backpropagation has been analyzed only in [12], which, however, focuses on DNNs with identical weights in all layers and relies on hard-to-compute quantities, such as kinematic eigenvalues [15].

A. Contributions

The contribution of this article is fourfold. First, leveraging general models of time-varying Hamiltonian systems [16], we provide a unified framework for defining H-DNNs, which encompass antisymmetric [11], [12] and Hamiltonian-inspired networks [11]. Second, for H-DNNs stemming from semi-implicit Euler (S-IE) discretization, we prove that the norm of the associated BSMs can never converge to zero irrespective of the network depth and weights. This result hinges on the symplectic properties of BSMs and is first shown in the continuous-time setting and then for H-DNNs models. For the result in the discrete-time case, we leverage developments in the field of geometric numerical integration [17], [18]. Third, we analyze the phenomenon of exploding gradients and show that they can be kept beyond the H-DNN framework.

II. Related Work

The BSM of a neural network denotes the sensitivity of the output of the last layer with respect to the output of intermediate layers. Its formal definition can be found in Section II-C.
under control by including suitable regularization terms in the training loss. Finally, we provide numerical results on benchmark classification problems demonstrating the flexibility of H-DNNs and showing that thanks to the absence of vanishing gradients, H-DNNs can substantially outperform standard multilayer perceptron (MLP) networks. From a more general perspective, our results show that one can characterize and analyze properties of deep networks by combining methods from system theory and numerical analysis.

A preliminary version of this work has been presented in the LADC conference [19]. Compared with the work in [19], this article offers a formal proof of nonvanishing gradients for arbitrarily deep H-DNNs, and it does not restrict to constant weights across layers nor H-DNNs stemming from forward Euler (FE) discretization. We also highlight that H-DNNs are fundamentally different from the architecture proposed in [20], which has a similar name but is designed to learn the Hamiltonian functions of mechanical systems.

The remainder of our article is organized as follows. H-DNNs and the gradient computations for the backpropagation algorithm are described in Section II. Our main results characterizing the nonvanishing properties of the BSM in continuous-time and discrete-time are described in Section III. Moreover, an informative upper-bound on the BSM norm is provided. Finally, numerical examples are discussed in Section IV and concluding remarks are given in Section V.

B. Notation

We use \( 0_{n \times n} \) (\( 1_{m \times n} \)) to denote the matrix of all zeros (all ones) of dimension \( m \times n \), \( I_n \) to denote the identity matrix of size \( n \times n \), \( 0_n \) to denote the square-zero matrix of dimension \( n \times n \) and \( 1_n \) to denote the vector of all ones of length \( n \). For a vector \( x \in \mathbb{R}^n \), \( \text{diag}(x) \) is the \( n \times n \) diagonal matrix with the elements of \( x \) on the diagonal. For vectors \( x \in \mathbb{R}^n, y \in \mathbb{R}^m \), we denote the vector \( x \in \mathbb{R}^{m+n} \) stacking them one after the other as \( z = (x, y) \). We adopt the denominator layout for derivatives, that is, the derivative of \( y \in \mathbb{R}^n \) with respect to \( x \in \mathbb{R}^m \) is \( \frac{\partial y}{\partial x} \in \mathbb{R}^{m \times n} \).

II. HAMILTONIAN DNNs

This section, besides providing a short introduction to DNNs defined through the discretization of nonlinear systems, presents all the ingredients needed for the definition and implementation of H-DNNs. Throughout the article, we focus on classification tasks since they have been used as benchmarks for similar architectures (see Section IV). However, the main results apply to regression tasks as well, which only require modifying the output layer of the network. We illustrate how H-DNNs generalize several architectures recently appeared in the literature. We also introduce the problem of vanishing/exploding gradients, which is analyzed in the rest of the article. Finally, by using backward sensitivity analysis, we derive a continuous-time representation of the backpropagation algorithm.

A. DNN Induced by ODE Discretization

We consider the first-order nonlinear dynamical system

\[
\dot{y}(t) = f(y(t), \theta(t)), \quad 0 \leq t \leq T
\]

where \( y(t) \in \mathbb{R}^n, y(0) = y_0 \) and \( \theta(t) \in \mathbb{R}^m \) is a vector of parameters. For specifying a DNN architecture, we discretize (2) with sampling period \( h = \frac{T}{N} \in \mathbb{N} \) and utilize the resulting discrete-time equations for defining each of the \( N \) network layers [21]. For instance, using FE discretization, one obtains

\[
y_{j+1} = y_j + h f(y_j, \theta_j), \quad j = 0, 1, \ldots, N - 1.
\]

The above equation can be seen as the model of a residual neural network [1], where \( y_j \) and \( y_{j+1} \in \mathbb{R}^n \) represent the input and output of layer \( j \), respectively. Clearly, it may be convenient to replace FE with more sophisticated discretization methods, depending on the desired structural properties of the DNN. For instance, Haber and Ruthotto [11] use Verlet discretization for a specific class of DNNs. Later in this work, we show that S-IE discretization will allow us to formally prove that the phenomenon of vanishing gradients cannot occur in H-DNNs.

A remarkable feature of ODE-based DNNs is that their properties can be studied by using nonlinear system theory for analyzing the continuous-time model (2). Furthermore, this allows one to study the effect of discretization independently.

B. From Hamiltonian Dynamics to H-DNNs

We consider the neural network architectures inspired by time-varying Hamiltonian systems [16], [22] defined as

\[
\dot{y}(t) = J(t) \frac{\partial H(y(t), t)}{\partial y(t)}, \quad y(0) = y_0
\]

where \( J(t) \in \mathbb{R}^{n \times n} \) is skew-symmetric i.e., \( J(t) = -J^T(t) \) at all times and the continuously differentiable function \( H : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R} \) is the Hamiltonian function. In order to recover the DNNs proposed in [11], [12], and [14], we consider the Hamiltonian function

\[
H(y(t), t) = [\sigma(K(t)y(t) + b(t))]^T 1_n
\]

where \( \sigma : \mathbb{R} \to \mathbb{R} \) is a differentiable map, applied elementwise when the argument is a matrix, and the derivative of \( \sigma(\cdot) \) is called activation function \( \sigma'(\cdot) \). Specifically, as it is common for neural networks, we consider activation functions \( \sigma : \mathbb{R} \to \mathbb{R} \) that are differentiable almost everywhere and satisfy

\[
|\sigma'(x)| \leq M
\]

for some \( M > 0 \), where \( \sigma'(x) \) denotes any subderivative. This assumption holds for common activation functions, such as tanh(\cdot), ReLU(\cdot), and the logistic function. By computing \( \frac{\partial H(y(t), t)}{\partial y(t)} \), system (4) can be rewritten as

\[
\dot{y}(t) = J(t)K^T(\sigma(K(t)y(t) + b(t))), \quad y(0) = y_0.
\]

The ODE (7) serves as the basis for defining H-DNNs. Indeed, as outlined in the previous section, a given discretization scheme for (7) naturally leads to a neural network architecture. In this context, we focus on FE and S-IE discretizations, resulting in the following DNN architectures.

\[
H_1\text{-DNN: By discretizing (7) with FE, we obtain}
\]

\[
y_{j+1} = y_j + h J_j K_j \sigma(K_j y_j + b_j)
\]

which can be interpreted as the Hamiltonian counterpart of the layer (3).

\[
H_2\text{-DNN: Assume that the number of features } n \in \mathbb{N} \text{ is even and split the feature vector as } y_j = (p_j, q_j) \text{ for } j = 0, 1, \ldots, N \text{ where } p_j, q_j \in \mathbb{R}^\frac{n}{2}. \text{ Furthermore, assume that } J_j = J \text{ does not vary across layers. Then, S-IE discretization of (7) leads to the layer equation}
\]

\[
\begin{bmatrix}
J_{p,j+1} \\
J_{q,j+1}
\end{bmatrix} =
\begin{bmatrix}
J_j \\
J_j
\end{bmatrix}
\]

\[
+ h J K_j \sigma
\]

\[
\begin{bmatrix}
J_{p,j+1} \\
J_{q,j+1}
\end{bmatrix}
\]

\[
+ b_j
\]

\]

This condition can be always fulfilled by performing feature augmentation [23].

\[
J =
\begin{bmatrix}
0_{\frac{n}{2}} & -X^T \\
X & 0_{\frac{n}{2}}
\end{bmatrix},
K_j =
\begin{bmatrix}
K_{p,j} \\
0_{\frac{n}{2}}
\end{bmatrix},
\]

\[
b_j =
\begin{bmatrix}
b_{p,j} \\
b_{q,j}
\end{bmatrix}
\]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
which yields the layer equations

\[ p_{j+1} = p_j - hX^T K_{q_j} \sigma(K_{q_j} q_j + b_{q_j}) \]  \hspace{1cm} (11)

\[ q_{j+1} = q_j + hXK_{p_j} \sigma(K_{p_j} p_{j+1} + b_{p_j}) . \] \hspace{1cm} (12)

It is easy to see that one can first compute \( p_{j+1} \) through (11) while \( q_{j+1} \) is obtained as a function of \( p_{j+1} \) through (12).\(^5\)

H\(_1\)-DNNs are motivated by the simplicity of the FE discretization scheme and will be compared in Section IV with existing DNNs proposed in [11] and [14] using benchmark examples. However, even if system (7) is marginally stable,\(^6\) FE discretization might introduce instability and lead to layer features \( y_l \) that grow exponentially with the network depth [19]. Instead, H\(_2\)-DNNs do not suffer from this problem because, as shown in [18], S-IE discretization preserves the marginal stability of (7). More importantly, as we show in Section III-B, H\(_2\)-DNN architectures completely prevent the phenomenon of vanishing gradients.

We conclude this section by highlighting that H\(_1\)- and H\(_2\)-DNNs are more general than the DNN architectures proposed in [11], [12], and [14]. A precise comparison is provided in Appendix A1.

Remark 1: Note that even when \( J \) is time-invariant, H-DNN models have a strong representation power as \( K_j \) and \( b_j \) inside the nonlinear function can change at every layer \( j \). The expressivity of H-DNNs is also validated by the numerical experiments in Section IV and the examples of [11], [12], and [14], which use subcases of H-DNN architectures.

C. Training of H-DNNs

Similarly to [11], [12], and [14], we consider multiclass classification tasks based on the training set \( \{ \{ y_{n,k}^0, c^k \}, k = 1, \ldots, S \} \), where \( S \) denotes the number of examples, \( y_{n,k}^0 \) are the feature vectors, and \( c^k \in \{ 1, \ldots, n \} \) are the corresponding labels. As standard in classification through DNNs, the architectures (8) and (9) are complemented with an output layer \( y_{N+1} = f_N(y_N, \theta_N) \) composed, e.g., by the softmax function, to rescale elements of \( y_N \) for representing class membership probabilities [5]. H-DNNs are trained by solving the following empirical risk minimization problem:

\[
\min_{\theta} \frac{1}{S} \sum_{k=1}^{S} L(f_N(y_{N,k}^0, \theta_N), c^k) + R(\theta)
\]

s.t. (8) or (9), \( j = 0, 1, \ldots, N-1 \) \hspace{1cm} (13)

where \( \theta \) denotes trainable parameters, i.e., \( \theta = \theta_0, \ldots, \theta_N \) with \( \theta_j = \{ J_j, K_j, b_j \} \) for \( j = 0, \ldots, N-1 \) and \( R(\theta) \) is a regularization term given by \( R(\theta) = \alpha_R R_K(K_{\theta,n-1}, b_{\theta,n-1}) + \alpha_I R_I(\theta_0, \ldots, \theta_{n-1}) + \alpha_Y R_Y(\theta_N) \). The term \( R_K \) is defined as \( \frac{1}{2} \sum_{j=1}^{N-1} \| \triangledown K_j - K_{j-1} \|_2^2 + \| b_j - b_{j-1} \|_2^2 \), which follows the work in [11] and [14], and favors smooth weight variations across consecutive layers. The terms \( R_I(\cdot) \) and \( R_Y(\cdot) \) refer to a standard \( L_2 \) regularization for the inner layers and the output layer, respectively. The coefficients \( \alpha \geq 0, \alpha_R \geq 0 \) and \( \alpha_I \geq 0 \) are hyperparameters representing the trade-off between fitting and regularization [11].

When using gradient descent to minimize (13), at each iteration, the gradient of the loss function \( L \) with respect to the parameters needs to be calculated. This gradient, for parameter \( i \) of layer \( N - j - 1 \) is computed according to the chain rule as

\[
\frac{\partial L}{\partial \theta_i_{N,j-1}} = \left( \frac{\partial y_{N,j}}{\partial \theta_i_{N,j-1}} \right) \left( \frac{\partial L}{\partial y_{N,j}} \right) = \left( \frac{\partial y_{N,j}}{\partial \theta_i_{N,j-1}} \right) \prod_{t=N-j}^{N-1} \left( \frac{\partial y_{l+1}}{\partial y_l} \right) \frac{\partial L}{\partial y_N} . \] \hspace{1cm} (14)


\[ ^3 \]The layer equations (11)–(12) are analogous to those obtained in [11] and [14] by using Verlet discretization.

\[ ^4 \]This is always the case for constant parameters \( J, K, \) and \( b \), see [16].

\[ ^5 \]\( \alpha_Y \) and \( \alpha_N \) are usually called weight decays.

Throughout the article, we will refer to the matrix

\[
\frac{\partial y_{N}}{\partial y_{N-j}} = \sum_{t=N-j}^{N-1} \frac{\partial y_{l+1}}{\partial y_l} \frac{\partial L}{\partial y_N} \] \hspace{1cm} (15)

as the BSM at layer \( N - j \), for \( j = 1, \ldots, N-1 \), which, according to (14), allows one to compute the partial derivatives \( \frac{\partial L}{\partial \theta_i_{N,j-1}} \). As shown in the next section, this quantity is the key to studying the phenomena of vanishing and exploding gradients.

D. Vanishing/Exploding Gradients

Gradient descent methods for solving (13) update the vector \( \theta \) as \( \theta^{(k+1)} = \theta^{(k)} - \gamma \cdot \nabla_{\theta^{(k)}} L \), where \( k \) is the iteration number, \( \gamma > 0 \) is the optimization step size, and the elements of \( \nabla_{\theta} L \) are given in (14). The problem of vanishing/exploding gradients is related to the BSM (15). Indeed, when \( \| \frac{\partial y_{N}}{\partial y_{N-j}} \| \) is very small, from (14), the gradients \( \frac{\partial L}{\partial \theta_i_{N,j-1}} \) vanish despite not having reached a stationary point, and the training stops prematurely. Vice-versa, if \( \| \frac{\partial y_{N}}{\partial y_{N-j}} \| \) is very large, the derivative \( \frac{\partial L}{\partial \theta_i_{N,j-1}} \) becomes very sensitive to perturbations in the vectors \( \frac{\partial y_{N}}{\partial y_{N-j}} \) and \( \frac{\partial L}{\partial y_N} \), and this can make the learning process unstable or cause overflow issues. Both problems are generally exacerbated when the number of layers \( N \) is large [5].

In Section III, we analyze in detail the properties of BSMS. To this purpose, it is convenient to first adopt the continuous-time perspective enabled by system (4).

E. Continuous-Time Backward Sensitivity Analysis

In this section, we study the continuous-time counterpart of the BSM (15), which is given by \( \frac{\partial y(T)}{\partial y(T-t)} \). The following lemma, which is based on the backward sensitivity analysis of (7) and whose proof can be found in Appendix A21, shows that \( \frac{\partial y(T)}{\partial y(T-t)} \) is the solution to a linear time-varying (LTV) system.

Lemma 1: Given the ODE (7) associated with an H-DNN, the continuous-time BSM \( \frac{\partial y(T)}{\partial y(T-t)} \) verifies

\[
\frac{d}{dt} \frac{\partial y(T)}{\partial y(T-t)} = A(t) - A(t) \frac{\partial y(T)}{\partial y(T-t)} \] \hspace{1cm} (16)

where \( t \in [0,T] \) and \( A(\tau) = K^T(\tau)D(y(\tau), \tau)K(\tau)J^T(\tau) \) with \( D(y(\tau), \tau) = \text{diag}(\sigma^2(K(\tau) y(\tau) + b(\tau))) \).

By Lemma 1, the phenomena of vanishing and exploding gradients are avoided if the LTV system (16) is marginally stable, i.e., its solutions are neither diverging nor asymptotically converging to zero.

When \( \theta(t) = \theta \) for every \( t \in [0,T] \), the matrix \( A(T-t) \) has all eigenvalues on the imaginary axis [12]. Our work [19] has further shown that \( A(T-t) \) is diagonalizable. While Chiang et al. [12] suggest that these spectral properties of \( A(T-t) \) may lead to nonvanishing and nonexploding gradients when \( y(t) \) varies slowly enough, to the best of the authors’ knowledge, there is no direct link between how fast \( A(T-t) \) varies and the stability of \( \frac{\partial y(T)}{\partial y(T-t)} \). Indeed, as opposed to linear-time-invariant systems, the stability of (16) cannot be determined solely based on the eigenvalues of the time-varying matrix \( A(T-t) \) [26]. As pointed out in [12], a rigorous analysis of the properties of \( A(T-t) \) can be conducted by using the notion of kinematic eigenvalues [15]. These are determined by finding a time-varying transformation that diagonalizes \( A(T-t) \). However, such transformation depends explicitly on the LTV ODE solution \( \frac{\partial y(T)}{\partial y(T-t)} \), which makes the computation of kinematic eigenvalues as hard as solving (16).

Motivated as above, rather than studying the properties of \( A(T-t) \), in this article, we analyze the properties of \( \frac{\partial y(T)}{\partial y(T-t)} \) directly.
III. MAIN RESULTS

In this section, we first prove that continuous-time BSMs are lower-bounded in norm by the value 1, independent of the choice of $T > 0$ and the time-varying weights $K(t)$ and $b(t)$. Second, we show that such property is preserved after S-IE discretization. This fact implies that arbitrarily deep H$_2$-DNNs enjoy nonvanishing gradients by design. Finally, we observe that, contrary to what has been conjectured in previous work [12], the gradients of general Hamiltonian networks may explode with the network depth. To mitigate this issue, we derive an informative upper-bound on the norm of the continuous-time BSM that holds for the general H-DNN architecture. This bound suggests utilizing a regularizer on the norms for the weights during training.

A. Nonvanishing BSM: A Continuous-Time Perspective

Our first goal is to establish that, under the assumption that $J(t) = J$ is constant for all $t$, $\frac{\partial y(T)}{\partial y(T-t)}$ is a symplectic matrix with respect to $J$.

**Definition 1 (Symplectic matrix):** Let $Q \in \mathbb{R}^{n \times n}$ be a skew-symmetric matrix, i.e., $Q + Q^T = 0$. A matrix $M$ is symplectic with respect to $Q$ if $M^T Q M = Q$.

Symplectic matrices are usually defined by assuming that $Q = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}$ with $n$ being an even integer [18]. In this respect, Definition 1 provides a slightly generalized notion of symplecticity.

**Lemma 2:** Consider an H-DNN as per (7) with $J(t) = J$ for all $t \in [0, T]$, where $J$ is any nonzero skew-symmetric matrix. Then, $\frac{\partial y(T)}{\partial y(T-t)}$ is symplectic with respect to $J$, i.e.,

$$J = \frac{\partial y(T)}{\partial y(T-t)} \top \frac{\partial y(T)}{\partial y(T-t)} = J$$

for all $t \in [0, T]$.

**Proof:** For brevity, let $\Psi = \frac{\partial y(T)}{\partial y(T-t)}$, $\tau = T - t$ and $D(\tau) = \text{diag}(\sigma'(K(y)\beta + b))$. We have

$$\frac{d}{dt}(\Psi^\top J \Psi) = \Psi^\top J \Psi + \Psi^\top J K(\tau)D(\tau)K(\tau)J \Psi + \Psi^\top J K(\tau)D(\tau)K(\tau)J \Psi - \Psi^\top J K(\tau)D(\tau)K(\tau)J \Psi = 0_n.$$

Since $\frac{\partial y(T)}{\partial y(T)} = I_n$ by definition, then $\frac{\partial y(T)}{\partial y(T-t)} \top \frac{\partial y(T)}{\partial y(T-t)} = J$. As the time derivative of $\frac{\partial y(T)}{\partial y(T-t)}$, $J \frac{\partial y(T)}{\partial y(T-t)}$ is equal to zero for every $t \in [0, T]$, then (17) follows.

We highlight that Lemma 2 is an adaptation of Poincaré theorem [18], [27] to the case of the time-varying Hamiltonian functions (5) and the notion of symplecticity provided in Definition 1. Next, we exploit Lemma 2 to prove that the norm of $\frac{\partial y(T)}{\partial y(T-t)}$ cannot vanish, for all $t \in [0, T]$.

**Theorem 1:** Consider an H-DNN as per (7) with $J(t) = J$ for all $t \in [0, T]$, where $J$ is any nonzero skew-symmetric matrix. Then

$$\left\| \frac{\partial y(T)}{\partial y(T-t)} \right\| \geq 1$$

for all $t \in [0, T]$, where $\| \cdot \|$ denotes any submultiplicative norm.

**Proof:** We know by Lemma 2 that (17) holds. Hence, we have

$$\left\| J \right\| = \left\| \left( \frac{\partial y(T)}{\partial y(T-t)} \right)^\top J \frac{\partial y(T)}{\partial y(T-t)} \right\| \leq \left\| \frac{\partial y(T)}{\partial y(T-t)} \right\|^2 \left\| J \right\|$$

for all $t \in [0, T]$. The above inequality implies (18).

B. Nonvanishing Gradients for H$_2$-DNNs

In this section, we show that, under mild conditions, S-IE discretization preserves the symplectic property (17) for the BSM. In turn, this allows us to show that gradients cannot vanish for any H$_2$-architecture based on S-IE discretization. We analyze the symplecticity of $\frac{\partial y_{l+1}}{\partial y_l}$ with respect to $J$ as per Definition 1.

**Lemma 3:** Consider the time-varying system (9) and assume that the time-invariant matrix $J$ has the block structure in (10). Then

$$\left[ \frac{\partial y_{l+1}}{\partial y_l} \right]^\top J \left[ \frac{\partial y_{l+1}}{\partial y_l} \right] = J$$

for all $l = 1, \ldots, N - 1$.

The proof of Lemma 3 can be found in Appendix A22 and is built upon the result of [18, Sec. VI, Th. 3.3] and the definition of extended Hamiltonian systems [17]. Lemma 3 allows us to prove that the BSMs of H$_2$-DNNs are lower-bounded in norm by 1 irrespective of the parameters and the depth of the network.

**Theorem 2:** Consider the H$_2$-DNN in (9). Assume that $J$ has the block structure in (10) and $J \neq 0_n$. Then, for all $j = 0, \ldots, N - 1$

$$\left\| \frac{\partial y_N}{\partial y_j} \right\| \geq 1$$

where $\| \cdot \|$ denotes any submultiplicative norm.

**Proof:** We know by Lemma 3 that (19) holds. Moreover

$$\frac{\partial y_{l+2}}{\partial y_l} = \frac{\partial y_{l+1}}{\partial y_l} \frac{\partial y_{l+2}}{\partial y_{l+1}}.$$

Then, by developing the terms in $\frac{\partial y_N}{\partial y_j}$ as per (21), calculating its transpose when needed, and applying iteratively (19), we obtain

$$\left\| J \right\| = \left\| \left( \frac{\partial y_N}{\partial y_j} \right)^\top J \left( \frac{\partial y_N}{\partial y_j} \right) \right\| \leq \left\| \frac{\partial y_N}{\partial y_j} \right\|^2 \left\| J \right\|$$

for all $j = 0, \ldots, N - 1$. This inequality implies (20).

C. Towards Nonexplosive Gradients

The next question is whether $\left\| \frac{\partial y(T)}{\partial y(T-t)} \right\|$ (respectively, its discrete-time counterpart $\left\| \frac{\partial y_N}{\partial y_j} \right\|$) may diverge to infinity as $T$ (respectively, $N$) increases. It was conjectured in [12] that, when all the weights $\theta(t) = \theta$ are time-invariant and $y(t)$ varies slowly enough, the gradients will not explode for arbitrarily deep networks. Unfortunately, gradients may explode even under such conditions. A simple example can be found in our technical report [28] where we consider the two-dimensional H-DNN with forward equation given by

$$\dot{y}(t) = c J \tanh(y(t))$$

where $J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ and $c \in \mathbb{R}$. Using a nominal trajectory starting at $y_0$ and a perturbed one starting at $y_0 + \gamma \beta$, with $\gamma > 0$ and $\beta \in \mathbb{R}^2$, we construct an estimate of the BSM norm. We study its limit value when $\gamma \to 0^+$ for sufficiently large $T$ and we show that it reaches a maximum value, which is proportional to $\frac{1}{\epsilon}$, thus, diverging to infinity as $\gamma$ approaches 0. Notice that the example is valid for arbitrarily small $\epsilon > 0$. Hence, even if $y(t)$ varies arbitrarily slowly, exploding gradients can still occur for increasing $T$. Furthermore, the growth of the gradients is independent of whether the weights are chosen in a time-invariant or time-varying fashion.

In the sequel, we consider time-varying weights $\theta(t)$ and general H-DNNs along with their continuous-time ODEs. While we expect exploding gradients as the network depth increases to infinity, we
derive upper bounds on the continuous-time BSMs. To this purpose, we provide the following result whose proof is given in Appendix A23.

**Proposition 1:** Consider the system (7) \( t \in [0, T], T > 0. \) Then
\[
\left\| \frac{\partial y(T)}{\partial y(T-t)} \right\|_2 \leq \sqrt{n} \exp(Qt) \quad \forall t \in [0, T]
\]  
(23)
where \( Q = M \sqrt{\frac{1}{n}} \max_{t \in [0, T]} \| K(t) \|_2 \| J(t) \|_2 \) and \( M \) satisfies (6).

**Proposition 1** reveals that the term \( \| K(t) \|_2 \| J(t) \|_2 \) is crucial in keeping the growth of continuous-time BSMs under control. The same is expected for discrete-time implementations.

This fact leads to the following observation. When implementing an H-DNN, it is beneficial to use the regularizer \( R_{t}(\theta_0, \ldots, \theta_{N-1}) = \sum_{j=0}^{N-1} (\| b_j \|_2 + \| J_j \|_2) \) in (13) to control, albeit indirectly, the magnitude of the BSMs. We exploit this regularization technique for image classification with the MNIST dataset in Section IV-B.

### IV. NUMERICAL EXPERIMENTS

In this section, we demonstrate the potential of H-DNNs on various classification benchmarks, including the MNIST dataset.\(^9\) Our first goal is to show that H-DNNs are expressive enough to achieve state-of-the-art performance on these tasks. Then, we validate our main theoretical results by showing that gradients do not vanish despite considering deep H\(_2\)-DNNs. Instead, when using the same data, standard MLP networks do suffer from this problem, which causes early termination of the learning process. Further numerical experiments showing how H-DNNs can be embedded in complex architectures for image classification can be found in [28, Appendix]. Specifically, results in [28] on CIFAR-10 dataset show comparable performance with state-of-the-art DNN architectures.

#### A. Binary Classification Examples

We consider two benchmark classification problems from [11] with two categories and two features. We show that H\(_2\)- and H\(_2\)-DNNs perform as well as the networks MS\(_1\)-DNNs, \( i = 1, 2, 3 \) given in Appendix A1 and introduced in [11], [12], and [14]. We optimize over the weights \( k_j \) and \( b_j \) and set \( J_j = J = \begin{pmatrix} 0 & -I \\ I & 0 \end{pmatrix} \) for \( j = 0, \ldots, N-1 \).

As outlined in Section II-C, we complement the DNNs with an output layer \( y_{N+1} = f_H(y_N, \theta_N) \), consisting of a sigmoid activation function applied over a linear transformation of \( y_N \). In (13), we use a standard binary cross-entropy loss function \( \ell \). The optimization problem is solved by using the Adam algorithm; we refer to Appendix A31 for details on the implementation.

**1) Comparison With Existing Networks:** We consider the “Swiss roll” and the “Double moons” datasets shown in Fig. 1. By performing feature augmentation [23], we use input feature vectors given by \( y_0, y_{2,1} \in \mathbb{R}^2 \) where \( y_0 \in \mathbb{R}^2, k = 1, \ldots, S \) are the input datapoints.

Table I presents the classification accuracies over test sets when using MS- and H-DNNs with different number of layers. We also give

\(^9\)The code is available at https://github.com/DecodEPL/HamiltonianNet.
C. Gradient Analysis

Our next aim is to provide a numerical validation of the main property of H-DNNs: the absence of vanishing gradients. We consider the “Double moons” dataset shown in Fig. 1(b) and analyze the norm of BSMs during the training of an H2-DNN and a fully connected MLP network.\textsuperscript{12}

Fig. 2(a) displays the BSM norms during the 960 iterations of the training phase of a 32-layer H2-DNN and Fig. 2(b) presents the same quantities for an MLP network with 8 and 32 layers. While the H2-DNN and the 8-layer MLP network achieve good performance at the end of the training (100% and 99.7% accuracy over the test set, respectively), the 32-layer MLP network fails to classify the input data since no BSM norm is smaller than 1 at any time.\textsuperscript{13}

In addition, it can be seen from Fig. 2(a) that the BSM norms do not explode and remain bounded as

\[
1 \leq \left\| \frac{\partial N}{\partial y_{N+1,\ell}} \right\| \leq 11 \quad \forall \ell = 0, \ldots, N - 1.
\]

This is in line with Proposition 1, where we show that, at each iteration of the training phase, BSM norms are upper-bounded by a quantity depending on the network depth and the parameters \( \theta_0, \ldots, N-1 \) of that specific iteration.

V. CONCLUSION

In this article, we proposed a class of H-DNNs obtained from the time discretization of Hamiltonian dynamics. We proved that H-DNNs stemming from S-IE discretization do not suffer from vanishing gradients and also provided methods to control the growth of associated BSMs. These results are obtained by combining concepts from system theory, as per Hamiltonian systems modeling, and discretization methods developed in numerical analysis. Although we limited our analysis to S-IE discretization, one can leverage the rich literature on symplectic integration\textsuperscript{18} for defining even broader classes of H-DNNs. This avenue will be explored in future research. It is also relevant to study applications of H-DNNs to optimal control problems—we refer the interested reader to the work in [29] for preliminary results in this direction.

\section*{APPENDIX}

\subsection*{A. Relationship of H-DNNs With Existing Architectures}

We introduce existing architectures proposed in [11], [12], and [14] and show how they can be encompassed in our framework. Since, for constant weights across layers, these architectures stem from the discretization of marginally stable systems, we call them MS-DNN (\( t = 1, 2, 3 \)).

\textbf{MS}_1-DNN: In [11], Haber and Ruthotto propose to use Verlet integration method to discretize

\[
\begin{bmatrix}
\dot{p}(t) \\
\dot{q}(t)
\end{bmatrix} = \sigma\left( \begin{bmatrix}
0 & K_0(t) & b_1(t) \\
0 & -K_0(t) & b_1(t)
\end{bmatrix}\begin{bmatrix}
p(t) \\
q(t)
\end{bmatrix} + \begin{bmatrix}
b_2(t) \\
b_2(t)
\end{bmatrix}\right)
\]

where \( p, q \in \mathbb{R}^2 \), obtaining the layer equations \( q_{j+1} = q_j - h\sigma(K_0p_j + b_{1,j}) \) and \( p_{j+1} = p_j + h\sigma(K_0q_j + b_{2,j}) \) as the layer equation. The resulting DNN is an instance of an H2-DNN when assuming \( K_0 \) to be invertible and setting \( J_0K_0 = I_n \) and \( K_j = \begin{bmatrix} 0 & K_{0,j} \\ -K_{0,j} & 0 \end{bmatrix} \) for all \( j = 0, \ldots, N - 1 \).

\textbf{MS}_2-DNN: Haber and Ruthotto [11] and Chang et al. [12] propose to use FE to discretize \( \dot{y}(t) = \sigma(K(t)y(t) + b(t)) \) where \( K(t) \) is skew-symmetric for all \( t \in [0, T] \), obtaining \( y_{j+1} = y_j + h\sigma(K_jy_j + b_j) \) as the layer equation. The resulting DNN is an instance of an H2-DNN by assuming that \( K_0 \) is invertible and by setting \( J_0K_0 = I_n \) and \( K_j = -K_j \) for all \( j = 0, \ldots, N - 1 \).

\textbf{MS}_3-DNN: In [14], Chang et al. propose to use Verlet integration method to discretize

\[
\begin{bmatrix}
\dot{p}(t) \\
\dot{q}(t)
\end{bmatrix} = \sigma\left( \begin{bmatrix}
0 & K_1(t) \\
0 & -K_2(t)
\end{bmatrix}\begin{bmatrix}
p(t) \\
q(t)
\end{bmatrix} + \begin{bmatrix}
b_2(t) \\
b_2(t)
\end{bmatrix}\right)
\]

where \( p, q \in \mathbb{R}^2 \), obtaining \( p_{j+1} = p_j + hK_{1,j}\sigma(K_{1,j}p_j + b_{1,j}) \) and \( q_{j+1} = q_j - hK_{2,j}\sigma(K_{2,j}p_{j+1} + b_{2,j}) \) as the layer.
equations. This is an instance of an H₂-DNN by setting
\[ K_j = \begin{bmatrix} 0 & 2 & 0 \\ 0 & K_{2,j} & 0 \\ 0 & 0 & 2 \end{bmatrix} \] and \[ J_j = \begin{bmatrix} 0 & 2 & 0 \\ 0 & J_{2,j} & 0 \\ 0 & 0 & 2 \end{bmatrix}. \]

In [11] and [14], MS₂⁻ and MS₂⁻DNNs have been called Hamiltonian-inspired in view of their similarities with Hamiltonian models, although a precise Hamiltonian function for the corresponding ODE has not been provided. Moreover, note that the Verlet discretization used coincides with S-IE.

We highlight that a necessary condition for the skew-symmetric \( n \times n \) matrix \( K_j \) to be invertible is that the size \( n \) of input features is even.\(^{14}\) If \( n \) is odd, however, one can perform input-feature augmentation by adding an extra state initialized at zero to satisfy the previous condition [23].

### B. Proofs

1) **Proof of Lemma 1:** Given the ODE (2) with \( y(0) = y_0 \), we want to calculate the dynamics of \( \frac{\partial y(T)}{\partial y} \). The solution to (2) can be expressed as
\[
y(t) = y(0) + \int_0^t f(y(\tau), \theta(\tau)) \, d\tau. \tag{24}\]

Analogously to [19, Lemma 2], evaluating (24) in \( t = T \) and \( t = T - t \), and taking the limit of their ratio as \( \delta \to 0 \), we obtain
\[
\frac{d}{dt} \frac{\partial y(T)}{\partial y} = -K_j \frac{\partial y(T)}{\partial y}. \tag{25}\]

Since in our case,\( f(y(T), \theta(T)) = J(T)(K(T)) \sigma(K(T)(y(T) + b(T))) \), then, dropping the time-dependence for brevity, we have
\[
\left. \frac{\partial}{\partial y} \right|_{y, \theta} = \left. \frac{\partial}{\partial y} \right|_{\sigma(Ky + b)} K J = K^\top \sigma'(Ky + b) K J^\top. \tag{26}\]

2) **Proof of Lemma 3:** We study the Hamiltonian system (4) in the extended phase space [17], i.e., we define an extended state vector \( \tilde{y} = (p, q, x, t) \),\(^{15}\) an extended interconnection matrix \( J = \begin{bmatrix} J & 0_{n+2 \times n} \\ 0_{2 \times n} & \Omega \end{bmatrix} \), \( \Omega = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \) and an extended Hamiltonian function \( H = H(p, q, x, t) + \varepsilon \), such that \( \frac{dt}{ds} = -\frac{\partial H}{\partial \varepsilon} \). Note that the extended Hamiltonian system defined by \( H \) is time-invariant by construction, i.e., \( \frac{dt}{ds} = \frac{\partial H}{\partial \varepsilon} = 0 \). Then, following [18, Sec. VI, Th. 3.3], it can be shown that \( \frac{\partial \tilde{y}_{i+1}}{\partial y_i} \) is a symplectic matrix with respect to \( \tilde{J} \), i.e., it satisfies
\[
\begin{bmatrix} \frac{\partial \tilde{y}_{i+1}}{\partial y_i} & \frac{\partial \tilde{y}_{i+1}}{\partial y_i} \end{bmatrix} \tilde{J} = \tilde{J}. \tag{26}\]

Next, we show that (26) implies symplecticity for the BS of the original time-varying system (9). First, we introduce the S-IE layer equations for the extended Hamiltonian dynamics, which reads \( \dot{p}_{i+1} = p_i - h\mathcal{X} \frac{\partial H}{\partial p}(p_i, q_i, t_i) \), \( \dot{q}_{i+1} = q_i + h\mathcal{X} \frac{\partial H}{\partial q}(p_i, q_i, t_i) \), \( \dot{t}_{i+1} = t_i - h \frac{\partial H}{\partial t}(p_i, q_i, t_i) \), and \( \ddot{e}_{i+1} = e_i - h \frac{\partial H}{\partial e}(p_i, q_i, t_i) \). Then, differentiation with respect to \( \tilde{y}_i = (p_i, q_i, e_i, t_i) \) yields\(^{16}\)
\[
\frac{\partial \tilde{y}_{i+1}}{\partial y_i} \begin{bmatrix} I_{n+2} + h \begin{bmatrix} H_{pp} & H_{pq} & 0 & H_{pt} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} J \end{bmatrix} \tag{27}\]

where \( H_{xy} = \frac{\partial H(p_{i+1}, q_i, t_i)}{\partial y} \) and \( x, y \) indicate any combination of two variables in the set \( \{p, q, t\} \).

14For a \( n \times n \) skew-symmetric matrix \( A \), we have, \( \det(A) = (\det(A^T)) = \det(A) \). If \( n \) is odd, then \( \det(A) = -\det(A) = 0 \). Thus, \( A \) is not invertible.

15Note that perturbing the elements of \( \tilde{y} \), the state vector can be rewritten as \((\tilde{p}, \tilde{q})\) where \( \tilde{p} = (p, \varepsilon) \) and \( \tilde{q} = (q, t) \).

16To improve readability, the dimension of the zero matrices has been omitted.
where the last equality follows from \( \|z_i(0)\|_2 = \|z_i\|_2 \) = 1 for all \( i = 1, 2, \ldots, n \). Then, applying the Gronwall inequality, we have for all \( t \in [0, T] \)

\[
\|z_i(t)\|_2 \leq \exp(QT),
\]

To characterize \( Q \), note that for \( \tau \in [0, T] \)

\[
\|A(\tau)\|_2 = \|K^\top(\tau)D(y(\tau), \tau)K(\tau)J(\tau)\|_2 \\
\leq \|K^\top(\tau)\|_2 \|D(y(\tau), \tau)\|_2 \|K(\tau)\|_2 \|J(\tau)\|_2 \\
\leq \|K^\top(\tau)\|_2 \|J(\tau)\|_2 M \sqrt{n}
\]

where \( D(y(\tau), \tau) = \text{diag}(\sigma'(K(\tau)y(\tau) + b(\tau))) \). The last inequality is obtained by applying Lemma 4 to \( D(y(\tau), \tau) \) and noticing that each column of \( D(y(\tau), \tau) \) can be expressed as \( d_i(y(\tau), \tau) = e_i^\top \sigma'(K(\tau)y(\tau) + b(\tau)) \), and (6) holds by assumption. Hence, we can set \( Q = M \sqrt{n} \max_{\tau \in [0, T]} \|K(\tau)\|_2 \|J(\tau)\|_2 \). Last, having bounded the column vectors \( z_i \), as per (30), we apply again Lemma 4 to obtain (23) and conclude the proof.

### C. Implementation Details

DNN architectures and training algorithms are implemented using the PyTorch library.\(^\text{17}\)

1) **Binary Classification Datasets:** For two-class classification problems we use 8000 datapoints and a mini-batch size of 125, for both training and test data. Training is performed using coordinate gradient descent, i.e., a modified version of stochastic gradient descent (SGD) with Adam (\( \beta_1 = 0.9, \beta_2 = 0.999 \)). Following [11], in every iteration of the algorithm, first the optimal weights of the output layer are computed given the last updated parameters of the hidden layers, and then, a step update of the hidden layers’ parameters is performed by keeping fixed the output parameters. The training consists of 50 epochs and each of them has a maximum of 10 iterations to compute the output layer weights. The learning rate, or optimization step size as per \( \gamma \) introduced in Section II-D, is set to 2.5 \times 10^{-2}. For the regularization, \( \alpha_r = 0 \), the weight decay for the output layer (\( \alpha_N \)) is constant and set to 1 \times 10^{-4} and \( \alpha = 5 \times 10^{-4} \).

2) **MNIST Dataset:** We use the complete MNIST dataset (60 000 training examples and 10 000 test examples) and a minibatch size of 100. For the optimization algorithm, we use SGD with Adam and cross-entropy loss. The learning rate, or optimization step size as per \( \gamma \) introduced in Section II-D, is initialized to 0.04 with a decay rate of 0.8 at each epoch. The total training step is 40 epochs. For MS1-DNN, we set \( \alpha_N = \alpha_r = 1 \times 10^{-3} \) and \( \alpha = 1 \times 10^{-3} \). For H1-DNN, we set \( \alpha_N = \alpha_r = 4 \times 10^{-3} \) and \( \alpha = 8 \times 10^{-3} \).

### References

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[2] W. Xiong et al., “The Microsoft 2016 conversational speech recognition system,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 5255–5259.

[3] S. Lucia and B. Karg, “A deep learning-based approach to robust non-linear model predictive control,” *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 511–516, 2018.

[4] R. Zoppoli, M. Sanguineti, G. Gnecco, and T. Parisini, *Neural Approximations for Optimal Control and Decision*. Berlin, Germany: Springer, 2020.

[5] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[6] M. Arjovsky, A. Shah, and Y. Bengio, “Unitary evolution recurrent neural networks,” *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1120–1128.

[7] K. Jia, D. Tao, S. Gao, and X. Xu, “Improving training of deep neural networks via singular value bounding,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4344–4352.

[8] E. Vorontsov, C. Trabelsi, S. Kadoury, and C. Pal, “On orthogonality and learning recurrent networks with long term dependencies,” in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3570–3578.

[9] K. Helfrich, D. Willmott, and Q. Ye, “Orthogonal recurrent neural networks with scaled Cayley transform,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1969–1978.

[10] J. Zhang, Q. Lei, and I. Dhillon, “Stabilizing gradients for deep neural networks via efficient SVD parameterization,” in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, vol. 80, pp. 8066–8114.

[11] E. Haber and L. Ruthotto, “Stable architectures for deep neural networks,” *Inverse Problems*, vol. 34, no. 1, 2017, Art. no. 014004.

[12] B. Chang, M. Chen, E. Haber, and E. H. Chi, “AntisymmetricRNN: A dynamical system view on recurrent neural networks,” in *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[13] Y. Lu, A. Zhong, Q. Li, and B. Dong, “Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations,” in *Proc. 6th Int. Conf. Learn. Representations*, 2018, pp. 3276–3285.

[14] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham, “Reversible architectures for arbitrarily deep residual neural networks,” in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2811–2818.

[15] U. Ascher, R. Mattheij, and R. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* (Classics in Applied Mathematics). Philadelphia, PA, USA: SIAM, 1995.

[16] A. van der Schaft, *L²-Gain and Passivity Theories in Nonlinear Control*. Berlin, Germany: Springer, 2017.

[17] M. A. de Gosson, *Symplectic Methods in Harmonic Analysis and in Mathematical Physics*. Berlin, Germany: Springer, 2011.

[18] E. Hainer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, 2nd ed. Berlin, Germany: Springer, 2006.

[19] C. L. Galimberti, L. Xu, and G. Ferrari-Trecate, “A unified framework for Hamiltonian deep neural networks,” in *Proc. 3rd Annu. Learn. Dyn. Control Conf.*, 2021, vol. 144, pp. 275–286.

[20] S. Greydanus, M. Dzamba, and J. Yosinski, “Hamiltonian neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 15379–15389.

[21] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural ordinary differential equations,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 1–16.

[22] Y. Guo and D. Cheng, “Stabilization of time-varying Hamiltonian systems,” *IEEE Trans. Control Syst. Technol.*, vol. 14, no. 5, pp. 871–880, Sep. 2006.

[23] E. Dupont, A. Doucet, and Y. W. Teh, “Augmented neural ODEs,” *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 3140–3150, 2019.

[24] L. El Ghaoui, F. Gu, B. Travacca, A. Arik, and A. Tsai, “Implicit deep learning,” *SIAM J. Math. Data Sci.*, vol. 3, no. 3, pp. 930–958, 2021.

[25] S. Bai, J. Z. Kolter, and V. Koltun, “Deep equilibrium models,” *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 688–699, 2019.

[26] M. Wu, “A note on stability of linear time-varying systems,” *IEEE Trans. Autom. Control*, vol. 19, no. 2, pp. 162–162, Apr. 1974.

[27] H. Poincaré, *Les Méthodes Nouvelles De La Mécanique Céleste*. Paris, France: Gauthier-Villars, 1899, vol. 3.

[28] C. L. Galimberti, L. Furieri, L. Xu, and G. Ferrari-Trecate, “Hamiltonian deep neural networks generalizing non-vanishing gradients by design,” 2022, arXiv:2105.13205.

[29] L. Furieri, C. L. Galimberti, M. Zakwan, and G. Ferrari-Trecate, “Distributed neural network control with dependability guarantees: A compositional port-Hamiltonian approach,” in *Proc. 4th Annu. Learn. Dyn. Control Conf.*, 2022, vol. 168, pp. 571–583.

\(^\text{17}\)https://pytorch.org/