Abstract—This article deals with a special type of Lyapunov functions, namely the solution of Zubov’s equation. Such a function can be used to characterize the exact boundary of the domain of attraction for systems of ordinary differential equations. In Theorem 2, we derive and prove an integral form solution to Zubov’s equation. For numerical computation, we develop two data-driven methods. One is based on the integration of an augmented system of differential equations; and the other one is based on deep learning. The former is effective for systems with a relatively low state space dimension and the latter is developed for high-dimensional problems. The deep learning method is applied to a New England 10-generator power system model. A feedforward neural network is trained to approximate the corresponding Zubov’s equation solution. The network characterizes the system’s domain of attraction. We prove that a neural network approximation exists for the Lyapunov function of power systems such that the approximation error is a cubic polynomial of the number of generators. The error convergence rate is $O(n^{-1/2})$, where $n$ is the number of neurons.

Index Terms—Deep learning, domain of attractions (DOAs), Lyapunov functions, power systems.

I. INTRODUCTION

For systems of ordinary differential equations (ODEs), it is well known that their stability can be determined by using Lyapunov functions. In addition, a Lyapunov function’s level set forms the boundary of a forward invariant set inside the domain of attraction (DOA). Originally introduced in the doctoral dissertation of Lyapunov, the theory has been actively studied for more than a century with applications to almost all fields of science and engineering that deal with dynamic phenomena.

There have existed tremendous works of literature on the study of how to find Lyapunov functions for systems of ODEs. Some widely used methodologies are based on the results and tools in related areas, such as Zubov’s equation [10], [12], [13], [17], [19], [22], [32], [46], [47], [48], linear programming [26], [27], and linear matrix inequality [7], [11], [16], [34], [38], [40]. In these works, the Lyapunov functions are represented using various types of approximations, including quadratic or general polynomials [9], [17], [28], [41], [48], rational functions [46], collocation or radial basis functions [19], [21], piecewise continuous functions [18], [27], and sum-of-squares [3], [5], [38]. For the interested readers, many survey papers were published in the last few decades summarizing research advancements at different phases in the development of the Lyapunov function theory and its computation [5], [15], [16], [20], [23], [40].

Existing results of computing DOAs face two challenges. They are either an inner approximation of DOAs, but cannot determine the exact boundary, or they are limited to low-dimensional systems. Computing the boundary of DOA for high-dimensional nonlinear systems is an open problem. Given a Lyapunov function of a system of ODEs, the interior of a level set of the Lyapunov function is a subset of the DOA. In general, the level sets of Lyapunov functions do not converge to the true boundary of the DOA. Only special Lyapunov functions, such as those that satisfy Zubov’s equation, can determine the exact boundary. In this case, a point in the state space is located inside the DOA if and only if the value of the Lyapunov function is less than one. In [47], the exact boundary of DOAs was approximated based on a Hamilton–Jacobi (HJ) equation. Both Zubov’s equation and the HJ equation are partial differential equations (PDEs). When solving these PDEs, existing computational algorithms have some fundamental limitations. Notably, discretization methods of PDEs suffer from the curse of dimensionality, i.e., the complexity of the solution increases exponentially with the dimension of the state space. In this case, the number of grid points in discretization or coefficients in polynomial approximation increases so fast that the computational algorithm is intractable in high-dimensional state spaces. The curse of dimensionality comes into effect when $n \geq 5$, depending on the computer’s capacity. In addition, existing PDE algorithms require boundary conditions. They are not applicable to Zubov’s equation because the boundary of the DOA is unknown. The proposed computational algorithms in this article enable innovative methods of characterizing the exact boundary of DOAs in high-dimensional state spaces that cannot be otherwise found using existing results. In this study, the limit
set can be a single equilibrium point or an asymptotically stable invariant set.

It is worth noting that ideas of using deep learning to mitigate the curse of dimensionality have been successfully applied to various PDEs over the past few years [25], [35], [42]. Its potential to applied problems that have high dimensions, such as numerical weather prediction, has attracted increasing attention [33], [37], [45]. For the specific problem of finding DOAs by solving Zubov’s equation, a method based on deep learning is studied in [22] under the assumption that the system satisfies a small-gain condition and it admits a compositional Lyapunov function.

An integral form solution to Zubov’s equation is introduced in [12] and [13] for systems that have unknown parameters and a single equilibrium point. In [10], an integral form Lyapunov function and a method of continuous and piecewise affine construction are combined. It is exemplified that the resulting Lyapunov function can be used to find subsets of DOAs. In [2], [14], and [43], subsets of the DOA are computed based on Lyapunov functions represented by neural networks. Different from abovementioned results, our goal is to approximate the exact boundary of DOAs, rather than a subset, by solving Zubov’s equation and to mitigate the curse of dimensionality. We deal with general attractors that are not limited to a single equilibrium.

The contributions of this article include the following.

1) We derive and prove an integral form solution to Zubov’s equation for general systems of ODEs.
2) We develop two data-driven computational methods for high-dimensional systems.
3) As a part of an applied example, we prove a proposition about the error upper bound and neural network complexity in the approximation of the DOA for a power system model.

One of the computational methods developed in this article is based on integrating an augmented system of ODEs, and the other one is based on deep learning. The former computes the Lyapunov function using ODE solvers, which is effective for relatively low-dimensional problems. We introduce deep neural networks to approximate the Lyapunov function if a system has a high dimension. For example, the computational methods are applied to a 10-generator power system, whose state space dimension is \( n = 20 \). The example is a reduced model of the power grid in New England. A feedforward neural network is trained that characterizes the DOA of the power system, which is an interesting research result in its own right.

II. INTEGRAL FORM SOLUTION TO ZUBOV’S EQUATION

In the stability theory for dynamical systems, it is known that a solution of Zubov’s equation is a special kind of Lyapunov functions. Given any point in the state space, the point is located inside the DOA if and only if the value of the solution of Zubov’s equation is less than one. In this section, we prove that a function defined in an integral form solves Zubov’s equation. In general, an analytic solution to Zubov’s equation does not exist. For low-dimensional systems, Algorithm 1 derived from the integral form solution is a practical way of numerically solving Zubov’s equation and characterizing the exact boundary of DOAs. For the supervised learning in Section IV, Algorithm 1 is critical because we need a computational algorithm to generate data for the training and validation of the neural network. Consider the following dynamical system:

\[
\dot{x}(t) = f(x(t)), \quad x \in \mathbb{R}^n
\]  

where \( x \) is the state variable and \( n \) is the dimension of the state space. The solution of (1) is denoted by

\[
x(t) = \phi(t, x_0)
\]  

where \( x_0 \) is the initial state and \( \phi(0, x_0) = x_0 \). We assume that, for any initial state, the solution of (1) is unique and defined in \( t \in [0, \infty) \). A point \( x \in \mathbb{R}^n \) is said to be an equilibrium if \( f(x) = 0 \). A forward invariant set, or simply invariant set, is a subset \( \mathcal{E} \subseteq \mathbb{R}^n \) so that

\[
\phi(t, x) \in \mathcal{E}, \quad t \geq 0
\]

for arbitrary \( x \in \mathcal{E} \). The concept of invariant set is a generalization of equilibrium because a set consisting of equilibrium points is an invariant set.

In this article, \( \|x\| \) represents the Euclidean norm on \( \mathbb{R}^n \). The distance between a point and a set is defined as the infimum of distances between the point and all points in the set, denoted by \( d(\cdot, \cdot) \). For example, the distance from a point, \( x_0 \), to an invariant set, \( \mathcal{E} \), is

\[
d(x_0, \mathcal{E}) = \inf \{ \|x_0 - x\| ; x \in \mathcal{E} \}.
\]

Given a positive number \( \delta \), the \( \delta \)-neighborhood of \( \mathcal{E} \) is defined by

\[
B_\delta(\mathcal{E}) = \{ x \in \mathbb{R}^n ; d(x, \mathcal{E}) < \delta \}.
\]

Definition 1 (Asymptotic stability [48]): (a) A closed invariant set \( \mathcal{E} \subseteq \mathbb{R}^n \) of (1) is called asymptotically stable if the following assumptions hold true.

1) For any \( \epsilon > 0 \), there exists a \( \delta > 0 \) such that \( x \in B_\delta(\mathcal{E}) \) implies \( d(\phi(t, x), \mathcal{E}) < \epsilon \) for all \( t \geq 0 \).
2) There exists a \( \delta > 0 \) such that

\[
\lim_{t \to \infty} d(\phi(t, x), \mathcal{E}) = 0.
\]

whenever \( x \in B_\delta(\mathcal{E}) \).

The DOA is denoted by \( \mathcal{D} \)

\[
\mathcal{D} = \left\{ x \in \mathbb{R}^n ; \lim_{t \to \infty} d(\phi(t, x), \mathcal{E}) = 0 \right\}.
\]

It is known that \( \mathcal{D} \) is always an open set [24]. The closure of \( \mathcal{D} \) is denoted by \( \overline{\mathcal{D}} \).

(b) The set \( \mathcal{E} \) is called uniformly asymptotically stable if, in addition, there exists a \( \delta \)-neighborhood of \( \mathcal{E} \) such that the limit (6) converges uniformly on \( B_\delta(\mathcal{E}) \). More specifically, for any \( \epsilon > 0 \), there exists \( T > 0 \) such that

\[
d(\phi(t, x), \mathcal{E}) < \epsilon
\]

for all \( x \in B_\delta(\mathcal{E}) \) and \( t \geq T \).

(c) An asymptotically stable closed invariant set \( \mathcal{E} \) is called uniformly attracting if there exists \( \delta > 0 \) such that, for any \( 0 < \delta \).
Under $h < \delta$, there exist $T > 0$ and $\gamma > 0$. Whenever $h < d(x, \mathcal{E}) < \delta$ and $t \in [0, T]$, we have
\[
d(\phi(t, x), \mathcal{E}) > \gamma.
\] (9)

The concept of uniformly asymptotically stable can easily be verified in some useful cases. For instance, if a system is autonomous, i.e., $f(x)$ is independent of $t$, a compact asymptotically stable $\mathcal{E}$ must be uniformly asymptotically stable [48]. Alternatively, if $\phi(t, x)$ converges to $\mathcal{E}$ exponentially for $x \in B_\delta(\mathcal{E})$, i.e.
\[
d(\phi(t, x), \mathcal{E}) \leq M d(x, \mathcal{E}) e^{-\lambda t}
\] (10)
for some $M > 0$ and $\lambda > 0$, then $\mathcal{E}$ is also uniformly asymptotically stable.

The main results of this section are based on the following assumptions.

Assumption 1: We assume that $f(x)$ in (1) is locally Lipschitz on $\mathbb{R}^n$ and $\phi(t, x)$ exists for all $(t, x) \in [0, \infty) \times \mathbb{R}^n$. Furthermore, $\|f(x)\|$ is bounded on $B_\delta(\mathcal{E})$ for some $\delta > 0$.

Assumption 2: Let $\mathcal{E}$ be a closed invariant set of (1). Let $W : \mathbb{R}^n \rightarrow \mathbb{R}$ be a continuous function. We assume
\[
W(x) = 0, \quad \text{if } x \in \mathcal{E}
\]
\[
W(x) > 0, \quad \text{otherwise}
\] (11)
and
\[
\lim_{d(x, \mathcal{E}) \rightarrow 0} W(x) = 0.
\] (12)

In addition, for any $\delta > 0$, there exists a $\gamma > 0$ such that
\[
W(x) > \gamma
\] (13)
whenever $d(x, \mathcal{E}) > \delta$.

Assumption 1 is standard for studies of ODEs, including Zubov’s theorem in [48]. Assumption 2 is required in the construction of an integral form solution to Zubov’s equation. Functions that satisfy Assumption 2 are easy to find. If $\mathcal{E}$ is a compact set, then (12) is implied by (11) and the continuity of $W(x)$. Two examples of $W(x)$ are
\[
W(x) = \|x - x_e\|^2 \quad \text{if } \mathcal{E} = \{x_e\}, \text{ where } x_e \text{ is a known equilibrium}
\] (14a)
or
\[
W(x) = \|f(x)\|^2 \quad \text{if the system has multiple equilibrium points.}
\] (14b)

The former is a simple function that can be used for problems with a known equilibrium. The latter is applicable to not only an isolated equilibrium but also systems that have a set of equilibrium points. For instance, the set $\mathcal{E}$ of the power system in Section V-A is a line. However, (14b) should be used with caution. One needs to verify (12) if $\mathcal{E}$ is noncompact. In addition, nonlinear systems may have both stable and unstable equilibrium points. The function in (14b) is applicable only if it is restricted inside a domain of interest that does not include unstable equilibrium. In the construction of a solution to Zubov’s equation, the following integral function is essential:
\[
I(x) = \int_0^\infty W(\phi(s, x))ds
\] (15)
where $\phi(t, x)$ is the solution of (1) satisfying $\phi(0, x) = x$.

Assumption 3: Let $\mathcal{E}$ be a closed invariant set. We assume that the integral in (15) converges in a neighborhood $x \in B_\delta(\mathcal{E})$ for some $\delta > 0$. Furthermore, we assume
\[
\lim_{x \in B_\delta(\mathcal{E}), d(x, \mathcal{E}) \rightarrow 0} I(x) = 0.
\] (16)

This assumption is not restrictive. If $W(x)$ is Lipschitz in $B_\delta(\mathcal{E})$ for some $\delta > 0$, and if $\phi(t, x)$ converges to $\mathcal{E}$ exponentially in the sense of (10), then Assumption 3 holds. To prove this claim, let $L$ be the Lipschitz constant of $W(x)$, then
\[
W(\phi(t, x)) \leq L \|\phi(t, x) - \bar{x}\|
\] (17)
for all $\bar{x} \in \mathcal{E}$. Therefore
\[
I(x) = \int_0^\infty W(\phi(s, x))ds \leq \int_0^\infty Ld(\phi(t, x), \mathcal{E})ds
\]
\[
\leq \int_0^\infty M d(x, \mathcal{E}) e^{-\lambda s}ds = \frac{M}{\lambda} d(x, \mathcal{E}).
\] (18)

Therefore, $I(x)$ is a finite number, and it satisfies (16). Zubov’s method is summarized in the following theorem. Its proof can be found in [48, Th. 19].

Theorem 1: (Zubov’s Theorem [48]) Let $\mathcal{E}$ be a closed invariant set of (1), in which $f(x)$ satisfies Assumption 1. Suppose $\mathcal{E}$ is uniformly asymptotically stable and uniformly attracting. Let $\mathcal{D}$ be an open set that contains a $\delta$-neighborhood of $\mathcal{E}$. Then, $\mathcal{D}$ is the DOA if and only if there exist continuous functions $V(x)$ defined on $\mathcal{D}$ and $\Psi(x)$ defined on $\mathbb{R}^n$ that satisfy the following properties.

1) If $x \in \mathcal{D}$ and $d(x, \mathcal{E}) \neq 0$, then $0 < V(x) < 1$.

2) If $x \in \mathbb{R}^n$ and $d(x, \mathcal{E}) \neq 0$, then $\Psi(x) > 0$. If $x \in \mathcal{E}$, then $\Psi(x) = 0$.

3) If $d(x, \mathcal{E}) \rightarrow 0$, then $V(x) \rightarrow 0$ and $\Psi(x) \rightarrow 0$.

4) For any sufficiently small $\delta > 0$, there exist $\gamma_1 > 0$ and $\gamma_2 > 0$ such that $d(x, \mathcal{E}) \geq \delta$ implies
\[
V(x) > \gamma_1 \text{ and } \Psi(x) > \gamma_2.
\] (19)

5) For any $\bar{x} \in \mathcal{D} \setminus \mathcal{D}$
\[
\lim_{x \in \mathcal{D}, x \rightarrow \bar{x}} V(x) = 1.
\] (20)

6) For any $x \in \mathcal{D}$, $V(\phi(t, x))$ is differentiable with respect to $t$ satisfying
\[
\left(\frac{dV(\phi(t, x))}{dt}\right)_{t=0} = -\Psi(x)(1 - V(x)).
\] (21)

Zubov’s equation introduced in [48] has different forms. We focus on the form in (21) because it is applicable to the most general case, in which $\mathcal{E}$ can be an unbounded set. Although the PDE (21) is defined for $x \in \mathcal{D}$ only, we treat $V(x)$ as a function defined in $\mathbb{R}^n$ because $V(x)$ can always be extended to $\mathbb{R}^n$ such that $V(x) = 1$ if $x \notin \mathcal{D}$. For the readers who are interested in the smoothness of $V(x)$, it is proved in [48, Th. 22] that, for systems of ODEs with an isolated equilibrium, there exists $V(x)$ that is...
continuously differentiable in \( D \) up to the order \( k \), where \( k \) is the smoothness of \( f \).

In the rest of this section, we will prove that the following function forms a solution to Zubov’s equation (21):

\[
V(x) = \begin{cases} 
\tanh(\alpha I(x)), & \text{if } I(x) < \infty \\
1, & \text{otherwise}
\end{cases} \\
\Psi(x) = \alpha(1 + V(x))W(x)
\]

(22)

where \( \alpha > 0 \) is a constant.

**Theorem 2:** Consider a system (1) and a closed invariant set \( \mathcal{E} \). Suppose \( \mathcal{E} \) is uniformly asymptotically stable. Suppose \( \bar{f}(x) \) satisfies Assumption 1. Suppose \( W : \mathbb{R}^n \to \mathbb{R} \) is a function satisfying Assumptions 2 and 3. Then, \( V(x) \) and \( \Psi(x) \) defined in (22) solves Zubov’s equation in the sense that they satisfy items (1)–(6) in Theorem 1.

Some properties of \( I(x) \) are essential. We prove the following lemma first, before the proof of Theorem 2.

**Lemma 1:** Let \( D \) be the DOA. Under the same assumption as in Theorem 2, \( I(x) \) has the following properties.

1. If \( I(x) < \infty \) for \( x \in D \) and \( I(x) \) is a continuous function on \( D \).

2. If \( I(x) > 0 \) if \( x \in D \setminus \mathcal{E} \) and \( I(x) = 0 \) if \( x \in \mathcal{E} \).

3. If \( x \not\in D \), \( I(x) = \infty \). If \( x \in D \setminus D \), then

\[
\lim_{x \to D, x \to x} I(x) = \infty.
\]

(23)

4. Given any \( x \in D \)

\[
\frac{d}{dt} I(\phi(t, x)) = -W(\phi(t, x)).
\]

(24)

**Proof:** (1) For any \( x \in D \) and a radius, \( \delta \), defined in Assumption 3, (7) implies that there exists a \( t_1 > 0 \) such that

\[
d(\phi(t, x), \mathcal{E}) < \delta
\]

if \( t \geq t_1 \). From Assumption 3, the integral

\[
\int_{0}^{\infty} W(\phi(s, \phi(t_1, x)))ds
\]

converges. Therefore

\[
I(x) = \int_{0}^{\infty} W(\phi(s, x))ds \\
= \int_{0}^{t_1} W(\phi(s, x))ds + \int_{t_1}^{\infty} W(\phi(s, \phi(t_1, x)))ds < \infty.
\]

(26)

In the next, we prove that \( I(x) \) is continuous on \( D \). Define

\[
z(t, x) = \int_{0}^{t} W(\phi(s, x))ds.
\]

(27)

Because \( \phi(t, x) \) is continuous, it is a known fact in real analysis that \( z(t, x) \) is a continuous function of \( t \) for any fixed value of \( x \) [44]. For any \( \epsilon > 0 \), Assumption 3 implies that

\[
I(x) = \int_{0}^{\infty} W(\phi(s, x))ds < \epsilon
\]

if \( x \in \mathcal{B}_{\delta_1}(\mathcal{E}) \) for some \( \delta_1 > 0 \). Given any \( x \in D \), there exists a \( t_1 > 0 \) such that \( \phi(t_1, x) \in \mathcal{B}_{\delta_1}(\mathcal{E}) \). Because \( \phi(t_1, x) \) is a continuous function of \( x \), there exists \( \delta_2 > 0 \) such that \( \phi(t_1, z) \in \mathcal{B}_{\delta_2}(\mathcal{E}) \) if \( ||z - x|| < \delta_2 \). From (28) and the fact that \( z(t, x) \) is a continuous function of \( x \), we have

\[
\lim_{x \to x} |I(z) - I(x)| = \lim_{x \to x} \left| z(t_1, z) + \int_{0}^{\infty} W(\phi(s, \phi(t_1, z)))ds - \cdots \\
-z(t_1, x) - \int_{0}^{\infty} W(\phi(s, \phi(t_1, x)))ds \right| \\
\leq \lim_{x \to x} |z(t_1, z) - z(t_1, x)| + 2\epsilon \\
= 2\epsilon.
\]

(29)

Because \( \epsilon \) can be arbitrarily small, (29) implies that \( I(x) \) is continuous at every \( x \in D \).

(2) Let \( x \) be a point in \( D \setminus \mathcal{E} \). From (11), \( W(\phi(t, x)) > 0 \) for small value of \( t > 0 \). Therefore, its integral is greater than zero, i.e., \( I(x) > 0 \). If \( x \in \mathcal{E} \), we have \( \phi(t, x) \in \mathcal{E} \) for \( t > 0 \) because \( \mathcal{E} \) is invariant. Then, (11) implies \( W(\phi(t, x)) \equiv 0 \) and \( I(x) = 0 \).

(3) If \( x \not\in D \), then \( \phi(t, x) \not\in D \) for all \( t > 0 \) (otherwise, \( \phi(t, x) \) approaches \( \mathcal{E} \)). From Assumption 2 and the assumption about \( D \) in Theorem 1, there exist numbers \( \delta > 0 \) and \( \gamma > 0 \) such that \( \mathcal{B}_{\delta}(\mathcal{E}) \subset D \) and

\[
W(x) > \gamma
\]

(30)

whenever \( d(x, \mathcal{E}) > \delta \). Therefore

\[
W(\phi(t, x)) > \gamma
\]

(31)

for all \( t > 0 \). Therefore

\[
I(x) = \int_{0}^{\infty} W(\phi(s, x))ds = \infty.
\]

(32)

For any \( x \in \overline{D} \setminus D, x \not\in D \). Therefore, \( I(x) = \infty \), i.e., for any

\[
M > 0,
\]

there exists a \( T > 0 \) such that

\[
\int_{0}^{T} W(\phi(s, x))ds > 2M.
\]

(33)

It is known that the integral in (33) is continuously dependent on \( \bar{x} \). There exists an open neighborhood of \( \bar{x} \) such that, for any \( x \) in the neighborhood

\[
\int_{0}^{T} W(\phi(s, x))ds > M.
\]

(34)

Therefore, for any \( M > 0 \), there exists an open neighborhood of \( \bar{x} \) in which

\[
I(x) \geq \int_{0}^{T} W(\phi(s, x))ds > M
\]

(35)

i.e., (23) is proved.

(4) Given any \( x \in D \), we have

\[
\frac{d}{dt} I(\phi(t, x)) = \frac{d}{dt} \left( \int_{0}^{\infty} W(\phi(s, \phi(t, x)))ds \right)
\]
\[ \frac{d}{dt} \left( \int_{-\infty}^{\infty} W(\phi(s, x))ds \right) = -W(\phi(t, x)). \]  

(36)

**Proof of Theorem 2:** In the following, we prove that \( V(x) \) and \( \Psi(x) \) satisfy items (1)–(6) in Theorem 1. For general dynamical systems, it is required in Theorem 1 that \( E \) is uniformly attracting. In this article, all dynamical systems are defined by equations in the form of (1), in which \( f(x) \) is continuous. If Assumption 1 holds, then \( \|f(x)\| \) is bounded on \( 0 < h \leq d(x, E) \leq \delta \). If \( T > 0 \) is small enough, then \( d(\phi(t, x), E) \) is uniformly bounded from below by a \( \gamma > 0 \) for all \( x \) satisfying \( 0 < h \leq d(x, E) \leq \delta \) and \( t \in [0, T] \). Therefore, \( E \) is uniformly attracting.

From (15), \( V(x) \equiv 1 \) is continuous in the interior of \( \mathbb{R}^n \setminus D \). From Lemma 1(1), \( V(x) \) is continuous on \( D \). From Lemma 1(3), \( V(x) \) is continuous at \( x \) on the boundary of \( D \). Therefore, \( V(x) \) is continuous on \( \mathbb{R}^n \). Consequently, \( \Psi(x) \) is continuous on \( \mathbb{R}^n \).

Lemma 1(1) and (2) as well as the facts that \( \tanh(0) = 0 \) and \( 0 < \tanh(x) < 1 \) for \( 0 < x < \infty \) imply \( 0 < V(x) < 1 \) if \( x \in D \setminus E \) and \( V(x) = 0 \) if \( x \in E \). Therefore, \( V(x) \) satisfy Condition (1) in Theorem 1.

If \( x \notin E \), then \( W(x) > 0 \). Therefore, \( \Psi(x) > 0 \). If \( x \in E \), then \( W(x) = 0 \). Therefore, \( \Psi(x) = 0 \). Condition (2) in Theorem 1 is satisfied.

Because of (12) and (16), we know \( \Psi(x) \to 0 \) and \( V(x) \to 0 \) as \( d(x, E) \to 0 \). Therefore, \( V(x) \) and \( \Psi(x) \) satisfy Condition (3) in Theorem 1.

From (13) in Assumption 2, \( \Psi(x) \) satisfies Condition (4). From Assumption 1, \( \|f(x)\| \) is bounded on \( B_{\delta}(E) \) for any \( \delta > 0 \) that is small enough. Therefore, there is a \( T > 0 \) such that \( d(\phi(T, x), E) > \delta/2 \) if \( d(x, E) > \delta \). From (13), we know

\[ I(x) \geq \int_{0}^{T} W(\phi(s, x))ds \geq \gamma T \]  

(37)

for some \( \gamma > 0 \), provided \( d(x, E) > \delta \). Therefore, \( V(x) \) satisfies Condition (4).

Lemma 1(3) implies Condition (5).

Now, we can prove that \( V(x) \) and \( \Psi(x) \) solves Zubov’s equation. For any \( x \in D \)

\[ \left( \frac{dV(\phi(t, x))}{dt} \right)_{t=0} = \left( \frac{d}{dt} \tanh(\alpha I(\phi(t, x))) \right)_{t=0} \]

\[ = (1 - \tanh^2(\alpha I(\phi(t, x)))) \alpha \left( \frac{dI(\phi(t, x))}{dt} \right)_{t=0} \]

\[ = -\alpha(1 - V^2(\phi(t, x)))W(\phi(t, x)) \bigg|_{t=0} \]

\[ = -\alpha(1 + V(\phi(t, x)))W(x)(1 - V(x)) \bigg|_{t=0} \]

\[ = -\Psi(x)(1 - V(x)). \]

(38)

**III. COMPUTATIONAL ALGORITHM BASED ON ODE SOLVERS**

The function, \( V(x) \), is a global Lyapunov function, which characterizes the exact DOA.

**Corollary 1:** Under the same assumption as in Theorem 2, \( x \in D \) if and only if \( V(x) < 1 \).

To characterize the DOA in this way, we must find effective and efficient computational algorithms for \( V(x) \). It is known from (23) that, when \( x \) approaches the boundary of \( D \), \( I(x) \to \infty \). The computation is increasingly time consuming in a thin layer along the boundary of \( D \). To be more specific, a boundary layer of \( D \) is defined as follows:

\[ D_{\text{layer}}(\delta) = \{ x \in D; 1 - \delta < V(x) < 1 \} \]  

(39)

where \( 0 < \delta < 1 \). In (22), \( V(x) \) is defined based on \( \tanh(z) \), which is shown in Fig. 1. We define the value of \( \delta \) as follows. Let \( \epsilon_{\text{prec}} > 0 \) be the machine precision of the computer. Let \( z_{\text{prec}} > 0 \) satisfy \( 1 - \tanh(z_{\text{prec}}) < \epsilon_{\text{prec}} \). Then, the boundary layer \( D_{\text{layer}}(\tanh(z_{\text{prec}})) \) is numerically indistinguishable from the boundary of \( D \). In all examples, we set \( z_{\text{prec}} = 20 \) and \( \epsilon_{\text{prec}} = 1 \times 10^{-10} \). This boundary layer is used to determine the scale factor, \( \alpha \). It is defined in (48) and illustrated in Example 1.

To approximate \( I(x) \) for \( x \in D \), the integral in (15) is approximated by integrating over a finite interval \([0, T]\), where \( T \) is large enough so that

\[ \frac{d}{dt} \int_{0}^{T} W(\phi(s, x))ds \bigg|_{t=T} < \delta_I \]  

(40)

where \( \delta_I > 0 \) is a small number. For \( x \) located outside the DOA, \( I(x) = \infty \). Numerically, we need a number \( M > 0 \) to terminate the integration process, i.e., if

\[ \int_{0}^{T} W(\phi(s, x))ds > M \]  

(41)

we terminate the integration. In this case, the value of \( I(x) \) is considered to be \( \infty \). Note that (41) does not guarantee that \( x \) is outside the DOA, no matter how large the value of \( M \) is. However, if the value of \( M \) is large enough, the set inside the DOA satisfying \( I(x) > M \) is contained in a thin boundary layer. In this case, the trajectory is practically unstable (a small perturbation may cause the trajectory to diverge). The values of \( M \) and \( \alpha \) are determined using a data-driven approach, which is shown in Fig. 2 and illustrated in Example 1. The following system of equations is called the augmented system of (1):

\[ \dot{x}(t) = f(x(t)), \quad x(0) \in \mathbb{R}^n \]

\[ \dot{z} = W(x), \quad z(0) = 0. \]

(42)

Note that the initial condition of the augmented system in \( \mathbb{R}^{n+1} \) has \( n \) degrees of freedom because \( z(0) = 0 \) is fixed. In the
following computational algorithm, $\delta_I$, $M$ and $\alpha$ are positive constants, in which $\delta_I$ is the stopping criterion for the computation of $I(x)$. No matter the DOA is compact or not, we compute $V(x)$ in a bounded set, which is denoted by $\mathcal{R}$.

**Algorithm 1:** Let $\delta_I$, $M$, $\alpha$, and $\Delta T$ be positive numbers. Let $\mathcal{R} \subset \mathbb{R}^n$ be a bounded set, in which $V(x)$ is computed.

1) **Numerical integration:** It uses an ODE solver to solve (42) for $t \in [(k-1)\Delta T, k\Delta T]$, $k = 1, 2, \ldots K$. In the solution, $x(0) = x$. The initial state of the $k$th solution is the final state of the $(k-1)$th solution. Stop the process at $K$ so that the $K$th solution satisfies either

$$z(K\Delta T) \geq M$$

(43)

or

$$\frac{\Delta z}{\Delta t} \Big|_{t=K\Delta T} < \delta_I.$$  

(44)

In (44), $\Delta t$ is the time step of the ODE solver, $\Delta z = z(K\Delta T) - z((K-1)\Delta T)$. In this case, $I(x) \approx z(K\Delta T)$.

2) **Evaluation:** Compute the value of $V(x)$

$$V(x) = \begin{cases} 1, & \text{if (43) holds} \\ \tanh(\alpha z(K\Delta T)), & \text{if (44) holds}. \end{cases}$$

(45)

In (44), $\Delta z/\Delta t$ can be replaced by $W(x(K\Delta T))$ if its value is available. It is important to point out that every ODE solver has approximation error, which grows with $K\Delta T$. If the length of time interval is too long, the integration error can deteriorate the accuracy of Algorithm 1. For systems that converge/diverge exponentially, this issue is less problematic because trajectories reach the set defined by (43) or (44) at an exponential rate over time. Before the end of this section, we introduce the van der Pol equation as an example of implementing the computational algorithm.

**Example 1:** Consider the following van der Pol equation:

$$\begin{align*}
\dot{x}_1 &= -x_2 \\
\dot{x}_2 &= x_1 - (1 - x_1^2)x_2.
\end{align*}$$

(46)

The origin, $x = 0$, is an asymptotically stable equilibrium. For the computation of $I(x)$, $W(x) = ||x||^2$ and $\delta_I = 10^{-6}$. A data-driven approach is adopted for the determination of $\alpha$ and $M$. We first generated a set of samples following the uniform distribution

$$\mathcal{S} = \{x_i; 1 \leq i \leq N\}$$

(47)

where $N = 3000$ is the sample size. We apply Step 1 in Algorithm 1 to the augmented system to compute the value of $I(x)$ for $x \in \mathcal{S}$. In Fig. 2, the cluster of points on the left contains the value of $I(x)$ satisfying (44). There is a clear gap between $M = 200$ and the cluster of $I(x)$ satisfying (44). We call this graph $I$-value plot. In Fig. 2, all values of $I(x)$ satisfying (43) are represented by a single point at $I(x) = 200$. It is numerically treated as $\infty$ in the computation of $I(x)$. Increasing the value of $M$ implies the increase of $K\Delta T$ in (43). The computation will take longer time if $x \notin \mathcal{D}$. However, if $M$ is too small, some points in $\mathcal{D}$ is assigned the value $I(x) = \infty$ or $V(x) = 1$, i.e., a point inside the DOA is classified as unstable. To avoid that, we choose the value of $M$ that should be larger than $I(x)$ for most states in $\mathcal{D}$. From the $I$-value plot, we see that any $M > 80$ has a clear gap from the data cluster on the left. By doing that, we reduce the possibility of assigning $V(x) = 1$ for $x \in \mathcal{D}$. In this example, we choose $M = 200$. This value should correspond to the boundary layer $D_{\text{layer}}(\tanh(\zeta_{\text{prec}})), \zeta_{\text{prec}} = 20$. For this purpose, we adopt the following scale factor in the computation of $V(x)$:

$$\alpha = \frac{\zeta_{\text{prec}}}{M}.$$  

(48)

Using the value of $\delta_I$, $M$, and $\alpha$ selected as above, $V(x)$ can be computed using Algorithm 1. The augmented system is solved using ODE45 in MATLAB. The averaged CPU time of evaluating $V(x)$ using a 2.4 GHz Intel Core i9 processor is $0.03$ s. The DOA is characterized by $V(x) < 1$. Its boundary can be approximated by a curve $V(x) = r$ if $r < 1$ is very close to $1$. Three level curves of $V(x)$ for $r = 0.7, 0.8,$ and $0.99$ are shown in Fig. 3.

For low-dimensional systems, such as that in Example 1, some existing methods can approximate the exact boundary of the DOA. For instance, examples of 2-D and 3-D systems are given in [47] based on a HJ equation. However, the discretization of the HJ equation on a grid is not applicable to high-dimensional systems because of the curse of dimensionality. In the next section, we introduce a learning-based method and apply it to a 20-D system.

**IV. DATA-DRIVEN METHOD BASED ON DEEP LEARNING**

The state space of the van der Pol equation in Example 1 is $\mathbb{R}^2$. Algorithm 1 is effective, although not super fast ($0.03$ s CPU time...
in Example 1), for this kind of problems that have a low state space dimension. When the dimension of systems increases, it can be challenging to conduct real-time computation based on Algorithm 1. However, Algorithm 1 has a special property namely causality-free, i.e., the solution of Zubov’s equation is computed at individual points without using the value of \( V(x) \) at other points in its neighborhood. Consequently, the computation does not require a grid. Demonstrated in several studies \([30, 31, 35, 36]\), causality-free algorithms are advantageous in data-driven computational methods for PDEs. They have perfect parallelism, a desirable property for generating a large amount of data. In contrast to PDE algorithms that are based on grids in space, a causality-free algorithm can solve high-dimensional problems. In addition, \( V(x) \) can be computed in any targeted area following any distribution that one may choose. The combination of a causality-free algorithm with supervised learning was proved to be effective and efficient for solving high-dimensional PDEs in optimal control \([35]\). In this section, we apply supervised learning and feedforward neural networks for the purpose of learning \( V(x) \) based on data generated using Algorithm 1. Once the training is done, \( V(x) \) is approximated by a neural network. Its evaluation is much faster than Algorithm 1.

**Algorithm 2:** Consider a system (1), a closed invariant set \( \mathcal{E} \), and a function \( W : \mathbb{R}^n \to \mathbb{R} \). Suppose that they satisfy Assumptions 1–3. Let \( R \subset \mathbb{R}^n \) be a bounded region, in which we compute \( V(x) \).

1) **Generating training data:** Randomly generate a set of samples \( S_0 \subset R \). For each \( x \in S_0 \), compute \( V(x) \) using Algorithm 1. The outcome, \( (x, V(x)) \), is a data point. Associated with each data point, Algorithm 1 results in a trajectory, \( (x(t), z(t)) \), of the augmented system (42). One can use this trajectory to collect more data points

\[
(x(t_k), V(x) - z(t_k)), \quad 1 \leq k \leq K
\]  

(49)

where \( (t_k > 0) \) is a sequence of time selected either randomly in \((0, K \Delta T)\) or based on other criteria. The final set of data are denoted by \( S_{\text{training}} \).

2) **Generating validation data:** Repeat the computation in Step 1 to generate another set of data, \( S_{\text{validation}} \), based on random initial conditions that are independent of \( S_{\text{training}} \).

3) **Designing neural network:** Design a feedforward neural network (Fig. 4)

\[
V^{NN}(x) = g_M \circ g_{M-1} \circ \cdots \circ g_1(x)
\]  

(50)

where \( g_k(z) = \sigma(W_k z + b_k) \), \( z \) is a vector, and \( \sigma \) is a vector-valued activation function, such as the hyperbolic tangent, the logistic, or ReLU function. A widely used loss function for training is

\[
l(W, b) = \frac{1}{|S_{\text{training}}|} \sum_{x \in S_{\text{training}}} (V^{NN}(x) - V(x))^2.
\]  

(51)

4) **Training and validation:** Find the parameters \( W \) and \( b \) for \( V^{NN}(x) \) that minimize \( l(W, b) \). Use \( S_{\text{validation}} \) to empirically evaluate the accuracy of \( V^{NN}(x) \).

There are many platforms and software packages one can use to conduct the training process for the problem of supervised learning formulated in Algorithm 2. In this article, the training is conducted using TensorFlow [1].

**Remark 1:** Although \( V^{NN}(x) \) in Algorithm 2 is an approximation of \( V(x) \), \( V^{NN}(x) \) is not necessarily a Lyapunov function because \( \frac{d}{dt} V^{NN}(\phi(t, x)) < 0 \) is not guaranteed. Similar issues were also found in [22] and [36]. In spite of this fact, \( V^{NN}(x) \) is still useful for the purpose of finding the DOA because \( V(x) < 1 \) if and only if \( x \in D \). If \( V^{NN}(x) < 1 - \epsilon_{\text{max}} \), where \( \epsilon_{\text{max}} \) is the error upper bound of \( V^{NN} \), then \( x \in D \). In reality, an analytically derived error upper bound is not available. The value of \( \epsilon \) is approximated based on randomly generated sample data using Algorithm 1.

V. APPLIED EXAMPLE—POWER SYSTEMS

A. Power System Model

In this section, we find a solution to Zubov’s equation that characterizes the DOA of a 10-generator 39-bus power system. The system’s topology is shown in Fig. 5. It is a reduction of the power grid in New England area that retains 39 main power substations and the backbone transmission network by its 46 branches. Among the 39 substations, there are ten generators.
The first nine generators are placed at buses No. 30 to No. 38 representing nine main power plants in the area and the tenth generator placed at No. 39 is an equivalent of the neighboring New York power grid. Effectively finding the stability of large-scale power systems in real time is an open problem. Illustrated in this section, the proposed Algorithm 2 enables a tool of checking stability based on deep learning, for which the accuracy can be empirically validated. The mathematical model of the power system is given by the following system of ODEs:

\[
\frac{d\omega_i}{dt} = \frac{\omega_0}{2H_i} \left( P_m - D \frac{\omega_i - \omega_0}{\omega_0} - E_i^2 G_{ii} + \sum_{j=1, j \neq i}^{10} E_i E_j [B_{ij} \sin(\delta_i - \delta_j) + G_{ij} \cos(\delta_i - \delta_j)] \right) \\
\frac{d\delta_i}{dt} = \omega_i - \omega_0
\]

(52)

where \( i = 1, \ldots, 10 \). In this model, each of the ten generators is modeled by two first-order ODEs, so-called swing equations. For the \( i \)th generator, the two state variables in (52) are \( \delta_i \), the rotor angle in radian, and \( \omega_i \), the rotor speed in radian per second. Other parameters, include \( H_i \) (the inertial constant of the generator), \( \omega_0 = 2\pi \times f_0 \) (the synchronous angular frequency in radian per second for an ac power system with frequency \( f_0 \)), \( D \) (the damping coefficient), \( P_m \) (the mechanical power input from the turbine), and \( E_i \) (the electromotive force or internal voltage of the generator). In addition, \( G_{ij} + j B_{ij} \), the mutual admittance between \( E_i \) and \( E_j \), is the \( i \)th row \( j \)th column element of the admittance matrix among all electromotive forces, and \( G_{ii} \) is the conductance representing the local load seen from \( E_i \). Details about the model and its parameters can be found in [6]. It is worth noting that the system has an infinite number of equilibrium points that are arranged in a straight line, in which \( \omega_i = \omega_0 \) and \( \delta_i - \delta_j \) are constants.

**B. Feedforward Neural Network Approximation of \( V(x) \)**

In the following, the state variables of (52) are denoted by:

\[
x = [\omega_1, \delta_1, \omega_2, \delta_2, \ldots, \omega_{10}, \delta_{10}]^T \in \mathbb{R}^{20}.
\]

The vector field on the right-hand side in (52) is denoted by \( f(x) : x \in \mathbb{R}^{20} \rightarrow \mathbb{R}^{20} \). The computation is around a fixed equilibrium point \( x_0 \), in which

\[
\omega_1 = 120\pi \\
\delta_0 \approx [-0.0335, 0.0470, 0.1586, 0.1641, 0.1114, \cdots, 0.1726, 0.2220, 0.1243, 0.2723, -0.1726]^T.
\]

(53)

The sample trajectories have random initial states following the uniform distribution in \( \mathcal{R} \):

\[
\mathcal{R} = \{ x \in \mathbb{R}^{20} : -0.4\pi < \delta_i - (\delta_0)_i < 0.4\pi, \cdots, 1.5 < w_i - 120\pi < 1.5 \text{ for } 1 \leq i \leq 20 \}.
\]

(54)

The system has infinitely many equilibria. In fact, the set of equilibria, \( \mathcal{E} \), is a line in \( \mathbb{R}^{20} \). We use the following \( W(x) \) in the computation of \( I(x) \):

\[
W(x) = \frac{1}{1000} f^T(x) f(x).
\]

(55)

The weight in this function is for rescaling \( I(x) \). Otherwise, its value gets very large as the initial state approaches the boundary of \( D \). For Algorithm 1, we choose \( M = 250 \). In the \( I \)-value plot (see Fig. 6) based on \( N = 28,092 \) samples, this value of \( M \) has a clear gap from the cluster of \( I(x) \) on the left-hand side. Corresponding to the boundary layer \( D_{layer}(\tanh(20)) \), the scale factor for \( V(x) \) should be \( \alpha = 20/250 = 0.08 \).

The datasets for training and validation are generated using Algorithm 1. For each dataset, a total of 6000 trajectories are computed by solving the augmented system in MATLAB using a MacBook Pro equipped with 3.1 GHz Quad-Core Intel Core i7. It can generate a trajectory within 0.5 s. Along each stable trajectory, four more data points are generated using (49) at \( t_k, k = 2, 4, 6, 8 \). The sizes of datasets are

\[
|S_{training}| = 28,092, |S_{validation}| = 28,100.
\]

(56)

The level surfaces of \( V(x) \) form the boundaries of invariant sets in \( D \). The datasets should contain information representing all level surfaces. The graph in Fig. 7 is the histogram of \( V(x) \) for \( x \in D_{training} \). It shows that the training data are distributed evenly from \( V(x) = 0.1 \) to \( V(x) = 0.9 \) without missing any segment. A lot of data are concentrated near \( V(x) = 1 \). This is reasonable because \( V(x) = 1 \) contains the boundary of DOA, where we would like to provide reliable prediction.

Fig. 4 shows a feedforward neural network defined in (50). In this example, the neurons in hidden layers are defined by \( \sigma(x) = \tanh(x) \). The output layer is a linear function. The input of the neural network is the state of the power system, \( x \in \mathbb{R}^{20} \). The
output is $V(x)$ that approximates $V(x)$. The neural network has 16 hidden layers; each layer has 40 neurons. The training is conducted using TensorFlow [1]. The accuracy is measured using root-mean-square error (RMSE)

$$
RMSE = \sqrt{\sum_{x \in D_{\text{validation}}} (V_{\text{NN}}(x) - V(x))^2} / |D_{\text{validation}}|.
$$

The training is a process whose result depends on the random initial guess. One of the trained $V_{\text{NN}}(x)$ has the following RMSE:

$$
RMSE = 2.0 \times 10^{-3}.
$$

The pointwise error over $D_{\text{validation}}$ has a bell-shaped distribution. The histogram and boxplot of pointwise error

$$\{V_{\text{NN}}(x) - V(x); \ x \in D_{\text{validation}}\}$$

are shown in Fig. 8. The error variation is small. The value of the 75th percentile is 0.00073 and the 25th percentile is −0.00053. The maximum pointwise error is 0.0445.

While the distribution of training data shown in Fig. 7 may provide a degree of confidence in the data, it should be noted that it does not guarantee the sufficiency of the dataset to train a neural network that generalizes well in high-dimensional state spaces. How much data are enough? This is, in general, an open problem for deep learning applications. To enhance our confidence in the trained neural network, we calculated the RMSE using a series of randomly generated validation datasets with varying sizes, ranging from 50 to 25 000. The outcomes are illustrated in Fig. 9. The neural network’s prediction error is stabilized when the data size is larger than 15 000.

**Remark 2:** The augmented system (42) has multiple variables, including $z$. The result of this section shows that a neural network can provide an approximation of $z$ without solving the entire system of equations (after the network is trained). This idea is promising in other applications, such as data assimilation of systems that require high-computational costs. In [37] and [45], for instance, deep learning is applied to nowcasting, i.e., predicting the future rainfall intensity in a local region over a relatively short period of time without running a full-scale data assimilation system. In general, applying deep learning in data assimilation for a local or partial approximation of large-scale systems is a new direction of research with potential applications to many areas.

### C. Is It Possible to Break the Curse of Dimensionality?

As a solution of a PDE, $V(x)$ is very difficult to compute, if not impossible, when the state space has a high dimension. For example, a PDE discretization based on $N$ points in each dimension requires a grid that has $N^n$ points in an $n$-dimensional state space. For the power system (52), in which $n = 20$, the grid size is $10^{40}$ if $N = 100$. Any numerical method based on a grid of such size is computationally intractable. Other numerical methods of solving PDEs, such as the finite element method, have the same problem if the state space has a high dimension. The fundamental challenge lies in the fact that the complexity of numerical approximation increases with the dimension of state space exponentially, a phenomenon called the curse of dimensionality. In [29], it is proved that the approximation of compositional functions using neural networks may not suffer from the curse of dimensionality. Rather than an exponential function, the approximation error depends on the dimension as a polynomial. The model in (52) is defined based on compositional functions in the same form as those addressed in [29]. Its compositional structure can be represented using a layered directed acyclic graph (layered DAG), which is shown in Fig. 10. The augmented system has an additional equation

$$
\dot{z} = W(x) = \frac{1}{1000} f^T(x) f(x), z(0) = 0
$$
where $f$ is the vector field representing the right-hand side in (52). The function in (59) is also a compositional function, whose layered DAG is shown in Fig. 11.

Applying the theory in [29], the colored nodes in Figs. 10 and 11 represent nonlinear functions. The white nodes represent linear functions. The compositional features, defined for nonlinear nodes only, are briefly introduced as follows.

1) $|\mathcal{V}|$ (complexity feature): It is the total number of nonlinear nodes in the layered DAG, where $\mathcal{V}$ is the set of nonlinear nodes of the compositional function.

2) $r_{max}$ (dimension feature): In this article, we assume that all nodes in $\mathcal{V}$ are $C^1$ functions. The dimension feature is defined to be the largest input dimension of nonlinear nodes in $\mathcal{V}$. For the compositional function in Figs. 10 and 11, $r_{max} = 1$.

3) $\Lambda$ (volume feature): Each nonlinear node, as a function, has a domain. Assume that the domain is a square of edge length $R$. The volume feature is defined to be the largest value in

$$\{\max\{R, 1\}||f||; f \in \mathcal{V}\}$$

where $|| \cdot ||$ is the Sobolev norm

$$||f|| = ||f||_{L^2} + \sum ||\frac{\partial f}{\partial x_i}||_{L^\infty}.$$  

4) $L_{max}$ (Lipschitz constant feature): It is the largest Lipschitz constants associated with nonlinear nodes. Note that this Lipschitz constant is defined based on the layered structure of the DAG. Discussions in the rest of this section have some further explanations. For more details, the readers are referred to [29].

Let us denote the augmented vector field in (52)-(59) by $\bar{f} = [f(x)]^T W(x)]^T$. Let $\bar{\phi}(t, x)$ represent the solution of (52)-(59), in which $x$ is the initial state. Given a time interval $[0, T]$, it is proved in [29, Th. 5.1] that there always exists a deep feedforward neural network with $C^\infty$ activation functions, denoted by $\tilde{\phi}_\text{NN}(x)$, satisfying

$$\left\| \tilde{\phi}_{\text{NN}}(x) - \bar{\phi}(T, x) \right\|_2 < (C_1 L_{max} |\mathcal{V}| + C_2) \bar{n}^{-1/r_{max}}$$

where $\mathcal{V}$ is the set of nonlinear nodes in the augmented system and $\bar{n}$ is the number of neurons used to approximate each nonlinear node. The complexity of neural networks is defined in the literature as either the number of neurons, for instance in [8] and [39], or the number of free parameters. We adopt the former definition. For the neural network in (62), the complexity of $\tilde{\phi}_\text{NN}(x)$ satisfies

$$\text{the complexity of } \tilde{\phi}_\text{NN} \leq \left( \bar{n}^{1/r_{max}} + 1 \right) \bar{n} |\mathcal{V}|.$$  

The constants, $C_1$ and $C_2$, in (62) are determined by $\|\bar{f}\|_2$, $\|\frac{\partial \bar{f}}{\partial x}\|_2$, $T$, and the input dimensions of the nodes in $\mathcal{V}$. From (45) in Algorithm 1, an approximate solution to Zubov’s equation has the following form:

$$\bar{V}(x) = \tanh(\alpha z(T, x))$$

where $z(t, x)$ is the solution of the augmented system (42) and $T > 0$ is a constant. For example, if $T = K\Delta T > M/\delta t$, then either (44) holds at some $k\Delta T, k \leq K$, or (43) holds at $T = K\Delta T$.

**Proposition 1:** Consider a power system (52) that has $m$ generators and the function $\bar{V}(x)$ defined in (64). Let $R \subset \mathbb{R}^2$ be a bounded set. Let $V_{\text{NN}}(x)$ represent a feedforward neural network that has $n$ neurons, which are hyperbolic tangent functions. Then, there exists $V_{\text{NN}}(x)$ that satisfies

$$|V_{\text{NN}}(x) - \bar{V}(x)| < (C_1 m^2 + C_2) \frac{m}{\sqrt{n} - 1}$$

for $x \in R$, where $C_1$ and $C_2$ are constants independent of $m$. They depend on $\|\bar{f}\|_2$, $\|\frac{\partial \bar{f}}{\partial x}\|_2$, $T$, $\alpha$, and the parameters in (52).

**Proof:** The proof is based on (62) and (63), which are proved in [29, Th. 5.1]. We first prove some properties of compositional features. Specifically, $L_{max}$ and $\Lambda$ are independent of $m$, and $|\mathcal{V}|$ is a polynomial function of $m$. Following the definition in [29], the Lipschitz constant associated with a node is the Lipschitz constant of the function with respect to the node when the node is treated as a variable. For example, consider the function in Fig. 10, in which $\cos(z)$ and $\sin(z)$ are the nonlinear nodes. If one of them, for instance the first $\cos(z)$ connecting to $\delta_1 - \delta_1$, is treated as a variable, the Lipschitz constant of the function associated with this node equals

$$\frac{\omega_0}{2H_i} E_i E_i G_{i1}.$$
where $\omega_0$, $H_i$, $E_i$, and $G_{ij}$ are parameters introduced in Section V-A. The Lipschitz constant of nonlinear nodes in Fig. 11 equals $1/1000$. Therefore, $L_{\max}$ is independent of $m$. The value of $\Lambda$ depends on the radius of the domain of nonlinear nodes [the sine and cosine functions in (52)] and their Sobolev norm (61). They are independent of $m$. The set of nonlinear nodes, $\mathcal{V}$, consists of $\cos(z)$, $\sin(z)$, and $z^2$ for the functions in (62) and (63). For each $1 \leq i \leq m$, there are $2(m-1)$ nonlinear nodes for the function in Fig. 10. There are $2m$ nonlinear nodes for the function in Fig. 11. Therefore, the total number of nonlinear nodes in the function of the augmented system (52)-(59) is $|\mathcal{V}| = 2m^2$. The dimension feature is $r_{\max} = 1$ because all nonlinear nodes have a single input. Let $\Phi(t, x)$ represent the solution of the augmented system (52)-(59). For any positive integer, $n$, there exists a neural network $\Phi_{\text{NN}}$ that satisfies the inequalities (62) and (63). For the power system, the inequalities are equivalent to

$$
\left\| \Phi_{\text{NN}}(x) - \Phi(T, x) \right\|_2 < (C_1 m^2 + \bar{C}_1) \frac{1}{n} \\
\tilde{n} < 2\tilde{n}^2 (2m^2) 
$$

(67)

where $\tilde{n}$ is the complexity of $\Phi_{\text{NN}}$. Solving for $\tilde{n}$ from the second equation in (67) and then applying it to the first equation, we have

$$
\left\| \Phi_{\text{NN}}(x) - \Phi(T, x) \right\|_2 < (C_1 m^2 + \bar{C}_1) \frac{2m}{\sqrt{n}},
$$

(68)

Adding one neuron to $\Phi_{\text{NN}}$, we define

$$
V_{\text{NN}}(x) = \tanh(\alpha \Phi_{2m+1}(x))
$$

where $\Phi_{2m+1}(x)$ is the last component of $\Phi_{\text{NN}}(x)$. Note that $z(t, x)$ is the last component of $\Phi(x)$ and $\bar{n} = n - 1$. Because the composition of neural networks is still a neural network and because $\tanh(\cdot)$ is Lipschitz, it is straightforward to justify that (68) implies (65).

Proposition 1 implies that the neural network approximation of $V(x)$ has an error that is a cubic polynomial of $n = 2m$, the state space dimension. The error does not grow exponentially. We can conclude that there exists a neural network approximate solution to Zubov’s equation that breaks the curse of dimensionality.

We would like to emphasize that finding the desired neural network in Proposition 1 is an open problem. It requires the identification of the constants, $C_1$ and $C_2$. It needs a neural network that has a compositional structure tailored to the right-hand side of the system’s ODE model. In neural network training and validation, no matter how large the dataset is, there are no general methodologies that can guarantee the error upper bound (65) for all $x \in \mathcal{R}$. In addition, the training process is a nonconvex optimization. It may converge to a local minimum. Many of these drawbacks are challenges facing general practice of deep learning, not limited only to the problems studied in this article.

VI. CONCLUSION

Characterizing the exact boundary of DOAs for general nonlinear systems of ODEs has been an open problem for decades. In this article, we develop practical methods of describing the DOA by finding a Lyapunov function that solves Zubov’s equation. The theoretical foundation is a theorem, in which we construct and prove a solution to Zubov’s equation. This solution can be represented in an integral form that is easy to compute. All that is needed is an ODE solver. The parameters in the algorithm can be determined based on simulation data. For problems that ODE solvers do not meet the requirement of real-time computation, such as large-scale power grids, this algorithm can still be used to generate data so that a feedforward neural network is trained to approximate the Lyapunov function. In fact, we mathematically prove that the swing equation of power systems admits a neural network approximate solution to Zubov’s equation. The approximation error and complexity of the neural network are polynomial functions of the number of generators. It implies that a solution exists that breaks the curse of dimensionality. The deep learning approach is applied to the New England 10-generator 39-bus power system, an interesting research problem in its own right.

REFERENCES

[1] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2016, arXiv:1605.04467v2.
[2] A. Abate, D. Ahmed, and A. Peruffo, “Formal synthesis of Lyapunov neural networks,” IEEE Contr. Syst. Lett., vol. 5, no. 3, pp. 773–778, Jul. 2021.
[3] A. Ahmadi, K. Krstic, and P. Parrilo, “A globally asymptotically stable polynomial vector field with no polynomial Lyapunov function,” in Proc. IEEE 50th Conf. Decis. Control, 2011, pp. 7579–7580.
[4] P. M. Anderson and A. A. Fouad, Power Systems Control and Stability, Piscataway, NJ, USA: IEEE Press, 1994.
[5] J. Anderson and A. Papachristodoulou, “Advances in computational Lyapunov analysis using sum-of-squares programming,” Discrete Continuous Dynamical Syst. Ser. B, vol. 20, no. 8, pp. 2361–2381, 2015.
[6] T. Athay, R. Podmore, and S. Virmani, “A practical method for the direct analysis of transient stability,” IEEE Trans. Power Appl. Syst., vol. PAS-98, no. 2, pp. 573–584, Mar. 1979.
[7] R. Bartels and G. Stewart, “Solution of the matrix equation AX XB=C,” Commun. ACM, vol. 15, pp. 820–826, 1972.
[8] A. R. Barron, “Universal approximation bounds for superpositions of a sigmoidal function,” IEEE Trans. Inf. Theory, vol. 39, no. 3, pp. 930–945, May 1993.
[9] R. Bellman, Introduction to Matrix Analysis, 2nd ed., New York, NY, USA: McGraw-Hill Book Company, 1970.
[10] J. Björnsson, P. Giesl, S. Hafstein, C. M. Kellett, and H. Li, “Computation of continuous and piecewise affine Lyapunov functions by numerical approximations of the Massera construction,” in Proc. IEEE 53rd Conf. Decis. Control, 2014, pp. 5506–5511.
[18] R. Goebel, A. Teel, T. Hu, and Z. Lin, “Conjugate convex Lyapunov functions for dual linear differential inclusions,” IEEE Trans. Automat. Control, vol. 51, no. 4, pp. 661–666, Apr. 2006.

[19] P. Giesl, Construction of Global Lyapunov Functions Using Radial Basis Functions, Lecture Notes in Math., 2004, 2007.

[20] P. Giesl and S. Hafstein, “Review on computational methods for Lyapunov functions,” Discrete Continuous Dynamical Syst. Ser. B, vol. 20, no. 8, pp. 2291–2313, 2015.

[21] P. Giesl, B. Hamzi, M. Rasmussen, and K. N. Webster, “Approximation of Lyapunov functions from noisy data,” J. Comput. Dyn., vol. 7, pp. 57–81, 2020.

[22] L. Grüne, “Computing Lyapunov functions using deep neural networks,” J. Comput. Dyn., vol. 8, pp. 131–152, 2021.

[23] O. Gürel and L. Lapidus, “A guide to the generation of Liapunov functions,” Ind. Eng. Chem., vol. 61, pp. 30–41, 1969.

[24] W. Hahn, Stability of Motion, Berlin, Germany: Springer-Verlag, 1967.

[25] J. Han and W. E., “Deep learning approximation for stochastic control problems,” 2016, arXiv:1611.07422v1.

[26] T. Johansen, “Computation of Lyapunov functions for smooth, nonlinear systems using convex optimization,” Automatica, vol. 36, pp. 1617–1626, 2000.

[27] P. Julian, J. Guivant, and A. Desages, “A parametrization of piecewise linear Lyapunov functions via linear programming,” Int. J. Control, vol. 72, pp. 702–715, 1999.

[28] D. Kalise and K. Kunisch, “Polynomial approximation of high-dimensional Hamilton-Jacobi-Bellman equations and applications to feedback control of semilinear parabolic PDEs,” SIAM J. Sci. Comput., vol. 40, 2, pp. A629–A652, 2018. [Online]. Available: https://epubs.siam.org/doi/10.1137/S003613991663620X

[29] W. Kang and Q. Gong, “Neural network approximations of compositional functions with applications to dynamical systems,” SIAM J. Control Optim., vol. 60, no. 2, pp. 786–813, 2022.

[30] W. Kang, Q. Gong, T. Nakamura-Zimmerer, and F. Fahroo, “Algorithms of data development for deep learning and feedback design: A survey,” Physica D: Nonlinear Phenomena, vol. 425, 2021, Art. no. 132955. [Online]. Available: https://doi.org/10.1016/j.physd.2021.132955.

[31] W. Kang and L. C. Wilcox, “Mitigating the curse of dimensionality: Sparse grid characteristic method for optimal feedback control and HJB equations,” Comput. Optim. Appl., vol. 68, no. 2, pp. 289–315, 2017.

[32] E. Kinnen and C. Chen, “Liapunov functions derived from auxiliary exact differential equations,” Automatica, vol. 4, pp. 195–204, 1968.

[33] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, “Machine learning – accelerated computational fluid dynamics,” Proc. Nat. Acad. Sci., vol. 118, no. 21, 2021, Art. no. e2101784118. [Online]. Available: https://doi.org/10.1073/pnas.2101784118.

[34] C. Mikkelsen, “Numerical methods for large Lyapunov equations,” Ph.D. thesis, Purdue Univ., West Lafayette, IN, USA, 2009.

[35] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, “Adaptive deep learning for high-dimensional Hamilton-Jacobi-Bellman equations,” SIAM J. Sci. Comput., vol. 43, no. 4, pp. 1221–1247, 2021.

[36] T. Nakamura-Zimmerer, Q. Gong, and W. Kang, “QRnet: Optimal regulator design with LQR-augmented neural networks,” IEEE Contr. Syst. Lett., vol. 5, no. 4, pp. 1303–1308, Oct. 2021.

[37] S. Otsuka, M. Ohighashi, V. P. Huyan, P. Tandeo, and T. Miyoshi, “A deep-learning approach to three-dimensional precipitation nowcasting,” in Proc. 2nd R-CCC Int. Symp., 2020. [Online]. Available: https://cccenter.co.jp/crrm2019program/data/abstracts/Poster1-87.pdf

[38] P. Parrilo, “Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization,” Ph.D. thesis, California Inst. Technol., Pasadena, CA, USA, 2000.

[39] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, “Why and when can deep - but not shallow - networks avoid the curse of dimensionality: A review,” Int. J. Automat. Comput., vol. 14, pp. 503–519, 2017.

[40] I. Polik and T. Terlaky, “A survey of the S-lemma,” SIAM Rev., vol. 49, pp. 371–418, 2007.

[41] C. J. Qian and W. Lin, “A continuous feedback approach to global strong stabilization of nonlinear systems,” IEEE Trans. Autom. Control, vol. 46, no. 7, pp. 1061–1079, Jul. 2001.

[42] M. Raisi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” J. Comput. Phys., vol. 378, pp. 686–707, 2019.

[43] S. M. Richards, F. Berkenkamp, and A. Krause, “The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems,” in Proc. 2nd Conf. Robot Learn., 2018, pp. 466–476. [Online]. Available: https://proceedings.mlr.press/v87/

[44] H. L. Royden and P. M. Fitzpatrick, Real Analysis, 3rd ed., Pearson, 1988.

[45] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, “Convolutional LSTM network: A machine learning approach for precipitation nowcasting,” in Proc. 28th Int. Conf. Neural Inf. Process. Syst., vol. 1, 2015, pp. 802–810.

[46] A. Vannelli and M. Vidyasagar, “Maximal lyapunov functions and domains of attraction for autonomous nonlinear systems,” Automatica, vol. 21, pp. 69–80, 1985.

[47] G. Yuan and Y. Li, “Estimation of the regions of attraction for autonomous nonlinear systems,” Trans. Inst. Meas. Control, vol. 41, no. 1, pp. 97–106, 2019.

[48] V. I. Zubov, Methods of A. M. Lyapunov and Their Application, Groningen, The Netherlands: P. Noordhoff Ltd., 1964.

Wei Kang (Fellow, IEEE) received the B.S. and M.S. degrees in mathematics from Nankai University, Tianjin, China, in 1982 and 1985, respectively, and the Ph.D. degree in mathematics from the University of California, Davis, Davis, CA, USA, in 1991.

He is currently a Professor of applied mathematics with U.S. Naval Postgraduate School, Monterey, and an Adjunct Professor of applied mathematics with the University of California, Santa Cruz, Santa Cruz. His research interests include optimal control, computational methods for dynamical systems, data assimilation, deep learning, and their engineering applications.

Kai Sun (Senior Member, IEEE) received the B.S. degree in automation and the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China, in 1999 and 2004, respectively.

He is currently a Professor with the Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, USA. From 2007 to 2012, he was a Project Manager in grid operations and planning with Electric Power Research Institute, Palo Alto, CA, USA.

Dr. Sun is an Associate Editor for IEEE TRANSACTIONS ON POWER SYSTEMS and IEEE OPEN ACCESS JOURNAL OF POWER AND ENERGY.

Liang Xu received the Ph.D. degree in marine, Earth, and atmospheric sciences from North Carolina State University, Raleigh, NC, USA, in 1995.

He is currently the Head of Atmospheric Dynamics and Prediction Branch and a Meteorologist with the Marine Meteorology Division, Naval Research Laboratory, Monterey, CA, USA, respectively. He and his team have developed, tested, and transitioned to the Fleet Numerical Meteorology and Oceanographic Center, Monterey, CA, USA, an operational global atmospheric 4DVar data assimilation system. His research interests include numerical weather prediction and data assimilation, focusing on critical issues related to analyzing and predicting atmospheric processes and phenomena within the Navy’s Earth system prediction capability.