Computation of Neural Networks Lyapunov Functions for Discrete and Continuous Time Systems with Domain Of Attraction Maximization

Benjamin Bocquillon, Philippe Feyel, Guillaume Sandou, Pedro Rodriguez-Ayerbe

Computation of Neural Networks Lyapunov Functions for Discrete and Continuous Time Systems with Domain Of Attraction Maximization

Benjamin Bocquillon[1], Philippe Feyel[1], Guillaume Sandou[2] and Pedro Rodriguez-Ayerbe[2]
[1]*Safran Electronics & Defense, 100 avenue de Paris, Massy, France*
[2]*Université Paris-Saclay, CentraleSupélec, CNRS, L2S, 3 rue Joliot Curie, 91192 Gif-Sur-Yvette, France*

Keywords: Lyapunov function, Domain of attraction, Optimization, Neural network, Nonlinear system.

Abstract: This contribution deals with a new approach for computing Lyapunov functions represented by neural networks for nonlinear discrete-time systems to prove asymptotic stability. Based on the Lyapunov theory and the notion of domain of attraction, the proposed approach deals with an optimization method for determining a Lyapunov function modeled by a neural network while maximizing the domain of attraction. Several simulation examples are presented to illustrate the potential of the proposed method.

1 INTRODUCTION

Lyapunov theory, introduced in the late nineteenth century (Lyapunov, 1892), is a classical way to investigate for the stability of an equilibrium point for a dynamical system. The method relies on the search for a function that exhibits three important properties that are sufficient for establishing the Domain Of Attraction (DOA) of a stable equilibrium point : (1) it must be a local positive definite function; (2) it must have continuous partial derivatives, and (3) its time derivative along any state trajectory must be negative semi-definite. Although efficient to prove stability once the so-called Lyapunov function is known, there is no general method for constructing such a function.

The Lyapunov function construction is still an open problem, but several methods, often based on optimization, have emerged in the literature. One can cite (Panikhom and Sujitjorn, 2012), where the best quadratic Lyapunov function is looked for. However, these methods are too conservative in case of industrial complex systems. The work of (Argáez et al., 2018) proposes a new iterative algorithm that aims to avoid obtaining trivial solutions when constructing completive Lyapunov functions. This algorithm is based on mesh-free numerical approximation and analyses the failure of convergence in certain areas to determine the chain-recurrent set. Once again, the method appears too difficult for being used in an industrial context where flexibility is needed. Finally, the survey (Giesl and Hafstein, 2015) has brought different methods and gave a wide overview of the methods that can be used for the Lyapunov function computation. It proposes conservative methods when the system is complex and highly non-linear.

However, to the authors' mind, Artificial Intelligence, Machine Learning and Neural Network bring a great opportunity to design powerful tools to justify and quickly certificate complex industrial systems such as in the aerospace field for instance. One of the first papers using Artificial Intelligence to compute Lyapunov function is (Prokhorov, 1994), where a so called Lyapunov Machine, which is a special-design artificial neural network, is described for Lyapunov function approximation. The author indicates that the proposed algorithm, the Lyapunov Machine, has substantial computational complexity among other issues to be resolved and defers their resolution to future work. The work of (Banks, 2002) suggests a Genetic Programming for computing Lyapunov functions. However, the computed Lyapunov functions may have locally a conservative behavior. In this study, the use of Neural Networks allows to overcome these limits. Neural Networks are known to be powerful regressors that can approximate any nonlinear function. As a result, they appear as a good candidate for the construction of a Lyapunov function. In the literature, one can find other works using neural network to construct or approximate a Lyapunov function (Serpen, 2005) and the paper (Petridis and Petridis, 2006) where the authors propose an interesting and promising approach for the construction of Lyapunov functions represented by neural networks.

In (Bocquillon et al., 2020), the authors propose to use a new constrained optimization scheme such that the weights of this neural network are calculated in a way that is mathematically proven to result in a Lyapunov function while maximizing the DOA.

Although most of works relative to Lyapunov computation deal with continuous time systems, only few ones related to the discrete time case. Numerical methods to compute Lyapunov functions for nonlinear discrete time systems have been presented, for example in (Giesl, 2007), where collocation is used to solve numerically a discrete analogue to Zubov's partial differential equation using radial basis functions. For nonlinear systems with a certain structure there are many more approaches in the literature. To name one, in (Milani, 2002) the parameterization of piecewise-affine Lyapunov functions for linear discrete systems with saturating controls is discussed. However, although effective, these approaches either look difficult to implement for real complex systems or are too conservative.

In this paper, we extend the proposition in (Bocquillon et al., 2020) dedicated to continuous systems to the case of discrete-time ones. We enhance the usability of the algorithm by extending this approach for discrete time systems. Our contribution is to unify the continuous and discrete time cases towards an efficient formulation of the optimization problem. Note that the goal of this study is not to stabilize a given plant, but to analyze its stability, computing an approximation of the potential domain of attraction. Our paper does not deal of the stabilization of the system itself.

The paper is composed of 4 sections: the first part is related to the design of neural network Lyapunov function for the continuous part. The second part, which is the main contribution of this paper, proposes an extension to the discrete time case. The third part is where we present the proposed algorithm to compute a candidate Lyapunov function through optimal neural network weights calculation while maximizing the DOA. And the last one describes examples showing efficiency of the paper.

2 Continuous Time Case

2.1 Theoretical Background

Notations and definitions used in the paper are called in this section. Let $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}_+$ denote the set of positive real numbers, $\|.\|$ denote a norm on $\mathbb{R}^n$, and $\mathbb{X} \subset \mathbb{R}^n$, be a set containing $X = 0$.

Consider the autonomous system given by (1).

$$\dot{X} = f(X) \tag{1}$$

where $f : \mathbb{X} \to \mathbb{R}^n$ is a locally Lipschitz map from a domain $\mathbb{X} \subset \mathbb{R}^n$ into $\mathbb{R}^n$ and there is at least one equilibrium point X_e, that is $f(X_e) = 0$.

Theorem 2.1 (Lyapunov Theory) (Khalil and Grizzle, 2002). Let $X_e = 0$ be an equilibrium point for (1). Let $V : D \to \mathbb{R}$ be a continuously differentiable function,

$$V(0) = 0 \text{ and } V(X) > 0 \text{ in } D - \{0\} \tag{2}$$

$$\dot{V}(X) \leq 0 \text{ in } D \tag{3}$$

then, $X_e = 0$ is stable. Moreover, if

$$\dot{V}(X) < 0 \text{ in } D - \{0\} \tag{4}$$

then $X_e = 0$ is asymptotically stable.

Where $D \subset \mathbb{X} \subset \mathbb{R}^n$ is called Domain Of Attraction (DOA) and the system will converge to 0 from every initial point X_0 belonging to D.

2.2 Neural Network Formalism

The work in this section is based on the approach developed in the paper (Bocquillon et al., 2020) and is reported here for sake of clarity.

Let us consider the autonomous system in (1), in which we assume without any loss of generality, that the equilibrium point considered for the stability analysis is the point 0 (X=0). Therefore,

$$f(0) = 0 \tag{5}$$

Suppose $V(X)$ is a scalar, continuous and differentiable function and its derivative with respect to time is given in (6).

$$G(X) = \frac{dV}{dt} = \sum_{j=1}^{n} \frac{\partial V}{\partial x_j} f_j(X) \tag{6}$$

with $X = [x_1,...,x_n]^T$.

The Lyapunov function is modeled by a neural network with a single hidden layer whose weights are calculated in such a way that is proven mathematically that the resulting neural network implements indeed a Lyapunov function, showing the asymptotic stability in the neighborhood of 0. We assume that the Lyapunov function $V(X)$ is represented by a neural network where the x_i are the inputs, w_{ji} are the weights of the hidden layer, a_i the weights of the output layer, h_i are the biases of the hidden layer, θ is the bias of the output layer; i=1,...,n and j=1,...,K where K is the number of neurons of the hidden layer and σ is the activation function of the neural network.

Therefore, $V(X)$ can be expressed as:

$$V(X) = \sum_{i=1}^{K} a_i \sigma(\nu_i) + \theta \tag{7}$$

$$\nu_i = \sum_{j=1}^{n} w_{ji} x_j + h_i \tag{8}$$

From (2) and (4), sufficient conditions for the point 0 of system (1) to be asymptotic stable in the sense of Lyapunov are:

(a1) $V(0) = 0$.

(a2) $V(X) > 0$ for all nonzero X in a neighbourhood of 0.

(a3) $G(0) = 0$.

(a4) $G(X) < 0$ for all nonzero X in a neighbourhood of 0.

From (a1) and (a2), sufficient conditions for $V(X)$ to have a local minimum at 0 are (Petridis and Petridis, 2006):

(v1) $V(0) = 0$.

(v2) $\left.\frac{\partial V}{\partial x_j}\right|_{X=0} = 0$ for all j=1,2,...,n.

(v3) H^V (the matrix of 2nd derivatives of V at X=0) is positive definite.

In the same way from (a3) and (a4), sufficient conditions for $G(X)$ to have a local maximum at 0 are (Petridis and Petridis, 2006):

(d1) $G(0) = 0$.

(d2) $\left.\frac{\partial G}{\partial x_j}\right|_{X=0} = 0$ for all j=1,2,...,n.

(d3) H^G (the matrix of 2nd derivatives of G at X=0) is negative definite.

Then, the second derivative of $V(X)$ and $G(X)$ are computed as functions of the neural network :

$$\begin{aligned} V_{qr} &= \left.\frac{\partial^2 V}{\partial x_q \partial x_r}\right|_{X=0} \\ &= \sum_{i=1}^{K} a_i \left.\frac{d^2\sigma(\nu_i)}{d\nu_i^2}\right|_{X=0} \left.\frac{\partial v_i}{\partial x_r}\right|_{X=0} w_{qi} \\ &= \sum_{i=1}^{K} a_i \left.\frac{d^2\sigma(\nu_i)}{d\nu_i^2}\right|_{X=0} w_{ri} w_{qi} \end{aligned} \tag{9}$$

$$\begin{aligned} G_{lp} &= \left.\frac{\partial^2 G}{\partial x_l \partial x_p}\right|_{X=0} \\ &= \sum_{j=1}^{n} \left(\sum_{i=1}^{K} a_i \left.\frac{d^2\sigma(\nu_i)}{d\nu_i^2}\right|_{X=0} w_{ji} w_{li} \right) J_{jp} + \\ &+ \sum_{j=1}^{n} \left(\sum_{i=1}^{K} a_i \left.\frac{d^2\sigma(\nu_i)}{d\nu_i^2}\right|_{X=0} w_{ji} w_{pi} \right) J_{jl} \end{aligned} \tag{10}$$

where $J_{qr} = \left.\frac{\partial f_q}{\partial x_r}\right|_{X=0}$ q=1,...,n; r=1,...,n; l=1,...,n; p=1,...,n.

Therefore,

$$H^{\mathrm{V}} = [V_{qr}(X=0)] \quad V_{qr} \text{ is given by (9)} \tag{11}$$

$$H^{\mathrm{G}} = [G_{lp}(X=0)] \quad G_{lp} \text{ is given by (10)} \tag{12}$$

Assuming that the Lyapunov function is represented by a neural network, conditions (v1) - (v3) and (d1) - (d3) reduce to (we choose here $\sigma(\nu) = \tanh(\nu)$):

$$(t1) \ \sum_{i=1}^{K} a_i \sigma(h_i) + \theta = 0. \tag{13}$$

$$(t2) \ \sum_{i=1}^{K} a_i (1 - tanh^2(h_i)) w_{qi} = 0 \quad \text{for} \quad q=1,...,n. \tag{14}$$

(t3) H^V as given by (11) is positive definite.

(t4) H^G as given by (12) is negative definite.

We only deal with differentiable activation functions.

3 Discrete Time Case

3.1 Additional Theoretical Background

Consider the autonomous discrete time system given by (15).

$$X(k+1) = f(X(k)) \tag{15}$$

where $f: \mathbb{X} \to \mathbb{R}^n$ is a locally Lipschitz map from a domain $\mathbb{X} \subset \mathbb{R}^n$ into $\mathbb{R}^n$ and there is at least one equilibrium point X_e , that is $f(X_e) = X_e$.

Theorem 2.2 (Lyapunov Theory Discrete Time)(Khalil and Grizzle, 2002).

Consider the autonomous system given by (15).

Let X_e = 0 be an equilibrium point for (15). Let $V : D \to \mathbb{R}$ be a continuous function,

$$V(0) = 0 \text{ and } V(X(k)) > 0 \text{ in } D - \{0\} \tag{16}$$

$$V(X(k+1)) - V(X(k)) \leq 0 \text{ in } D \tag{17}$$

then, X_e = 0 is stable. Moreover, if

$$V(X(k+1)) - V(X(k)) < 0 \text{ in } D \tag{18}$$

then X_e = 0 is asymptotically stable.

3.2 Adaptation of the Neural Network Formalism

Let us consider the autonomous system in (15), in which we assume that the equilibrium point of interest for the stability analysis is the point 0. Therefore,

$$f(0) = 0 \tag{19}$$

Suppose $V(X(k))$ the Lyapunov function for the discrete time case and,

$$G(X(k)) = V(X(k+1)) - V(X(k)) \tag{20}$$

with $X(k) = [x_1(k), ..., x_n(k)]^T$.

In the discrete time case, sufficient conditions (a1) - (a4) for the point 0 of system to be asymptotically stable in the sense of Lyapunov are still the same than in the continuous part explained above. Conditions (v1) - (v3) and (d1) - (d3) which prove that $V(X(k))$ to have a local minimum at 0 and $G(X(k))$ to have a local maximum at 0 are also equivalent. However, the derivative of the function $V(X)$ is not defined in the same way for the continuous and the discrete time case. Therefore, in this extension we will present how to define the new H^{G} matrix.
First and second derivatives of $G(X(k))$ can be calculated from (20) :

$$G_l = \frac{\partial G}{\partial x_l} = \sum_{h=1}^{n} \frac{\partial V}{\partial x_h}\frac{\partial f_h}{\partial x_l} - \frac{\partial V}{\partial x_l} \tag{21}$$

$$\begin{aligned} G_{lp} &= \frac{\partial^2 G}{\partial x_l \partial x_p} \\ &= \sum_{h=1}^{n} \frac{\partial^2 V}{\partial x_h \partial x_p}\frac{\partial f_h}{\partial x_l} + \sum_{h=1}^{n} \frac{\partial V}{\partial x_h}\frac{\partial^2 f_h}{\partial x_l \partial x_p} \\ &- \frac{\partial^2 V}{\partial x_l \partial x_p} \quad \text{with } l = 1, ..., n \text{ and } p = 1, ..., n \end{aligned} \tag{22}$$

In view of condition (v2), (21) and (22) result in,

$$G_l(0) = 0 \text{ for all } l = 1, 2, ..., n \tag{23}$$

and

$$\begin{aligned} G_{lp}(0) &= \sum_{h=1}^{n} \left.\frac{\partial^2 V}{\partial x_h \partial x_p}\right|_{X(k)=0} \left.\frac{\partial f_h}{\partial x_l}\right|_{X(k)=0} \\ &- \left.\frac{\partial^2 V}{\partial x_l \partial x_p}\right|_{X(k)=0} \end{aligned} \tag{24}$$

On the basis of (24) we can write,

$$H^{\mathrm{G}} = [G_{lp}(X(k) = 0)] = H^{\mathrm{V}} J - H^{\mathrm{V}} \tag{25}$$

since H^{V} is symmetric. H^{G} is a matrix whose elements are $G_{lp}(0)$ and J is the Jacobian of $f(X(k))$ at $X(k) = 0$.

Let us assume that the Lyapunov function, $V(X(k))$, is represented by the neural network defined in (7) and (8). Therefore, the second derivative of $G(X(k))$ is computed as functions of the neural network weights,

$$\begin{aligned} G_{lp}(0) &= \sum_{j=1}^{n} \Big(\sum_{i=1}^{K} a_i \frac{\partial^2 \sigma(\nu_i)}{\partial^2 \nu_i} w_{ji} w_{li}\Big) J_{jp} \\ &- \sum_{j=1}^{n} \Big(\sum_{i=1}^{K} a_i \frac{\partial^2 \sigma(\nu_i)}{\partial^2 \nu_i} w_{ji} w_{pi}\Big) \end{aligned} \tag{26}$$

where $J_{qr} = \left.\frac{\partial f_q}{\partial x_r}\right|_{X(k)=0}$ q=1,...,n; r=1,...,n; l=1,...,n; p=1,...,n.

Therefore,

$$H^{\mathrm{V}} = [V_{qr}(X(k) = 0)] \quad V_{qr} \text{ is given by (9)} \tag{27}$$

$$H^{\mathrm{G}} = [G_{lp}(X(k) = 0)] \quad G_{lp} \text{ is given by (26)} \tag{28}$$

Point $X(k) = 0$ is asymptotically stable if conditions (t1) - (t4) hold with the new definition of the matrix H^{G}.

4 The Proposed Algorithm

4.1 Optimization Scheme

First, for an appropriate Lyapunov function to be determined, values of the weights of the neural network should be calculated such that the conditions (t1)-(t4) are satisfied. To this end, a cost function, Q, should

be selected so that positivity and negativity respectively of H^V and H^G are constrained. The symmetric matrix H^G is negative definite if all its eigenvalues are negative.

Denote λ_i^v, i=1,...,n_v the set of the n_v eigenvalues of H^V and λ_i^g, i=1,...,n_g the set of the n_g eigenvalues of H^G.

4.2 DOA Maximization problem

Conditions (a2) and (a4) prove the asymptotic stability only "in a neighbourhood of 0". Consider that a Lyapunov function $V(X)$ is given, by definition of D in (2) and (3), the system will converge to 0 from every initial point X_0 belonging to D. Thus, to maximize the DOA we search for an approximation $\hat{D}$ as large as possible, where the system is stable. Therefore, $\hat{D}$ = D would be the best possible case.

Consider $\mathcal{Z}$ a set of points obtained from a hypercube whose faces are gridded in order to cover a sufficiently large enough domain $\subset \mathbb{X}$. Note that, $\mathcal{Z}$ has to be chosen large enough such as it contains D. We denote P = max ($ratio_v$, $ratio_{dv}$) where $ratio_v$ are the number of points $X \subset \mathcal{Z}$ where evaluated $V(X)<$ 0 and $ratio_{dv}$ are the number of points $X \subset \mathcal{Z}$ where evaluated $\dot{V}(X)> 0$. In order to maximize $\hat{D}$, we have to minimize P.

Note that, there are other more intelligent methods to determine the gridding but so far, we used the one presented previously. For example, in the future, a possible approach is to use the gradient of the current Lyapunov function candidate during the optimization. An idea could be to analyze the first run of the optimization algorithm and the larger the gradient is, the thinner the gridding has to be in the next runs.

4.3 Constrained Implementation

We now formalize the problem as a constrained one to avoid the use of barrier function which would lead to a suboptimal problem (Petridis and Petridis, 2006), whose solution needs to be a posteriori verified. The scheme proposed here is flexible so that more complex problems such as exponential stability, robust stability or Input-to-state stability (ISS), will efficiently be tackled in future works.

The problem can be expressed in the general form of an optimization problem in which the cost function $\mathcal{Q}$ needs to be minimized.

To this purpose, we set down:

$$H^{V\prime} = H^V \times -1$$

Denote $\lambda^{v\prime}$ as the eigenvalues of $H^{V\prime}$.

$$\bar{\lambda}^{v\prime} = \max(\text{real}(\lambda^{v\prime}))$$

$$\bar{\lambda}^{g} = \max(\text{real}(\lambda^{g}))$$

$$\bar{\lambda} = \max\,(\,\bar{\lambda}^{v\prime}\,,\,\bar{\lambda}^{g})$$

According to (7) and (8), we denote α as the decision variables where:

$$\alpha = [w_{ji}, a_i, h_i, \theta] \tag{29}$$

Then, the cost function to be minimized has the following form:

$$\min_{\alpha} \mathcal{Q}$$

If $\bar{\lambda} > 0$

$$\mathcal{Q} = \bar{\lambda}$$

Else

$$\mathcal{Q} = -\frac{1}{P+1}$$

which is a similar formulation of the cost function that can be find in (Feyel, 2017) and has proven its efficiency. According to the definition of the problem $\mathcal{Q}$, a neural network candidate is a Lyapunov function if $\mathcal{Q} < 0$. Assuming $D \subset \mathcal{Z}$, best case, $\mathcal{Q}$ = - 1, refers to $\hat{D}$ = D. The P+1 avoids singularities when P = 0. The benefit of this formulation is its great flexibility: easy adaptation for a multitude of complex problems, extension to other type of stability and no additional parameter to tune for the penalty function. Besides, we have the guarantee that in the domain $\hat{D}$ the eigenvalues of H^V and H^G have respectively real positive and negative values. Finally, no parameter is needed.

5 Simulation Results

In this section, we apply our approach to one continuous time system and to two different discrete time systems to validate the extension. For clarity purposes, we have chosen some two dimensional examples so that the results can be easily plotted.

The entire test was performed on a machine equipped with an Intel Core i5 - 8400H (2.5 GHz) processor and 16 GB RAM.

5.1 Parameter Settings

The parameters settings used in the tests are as follows:

- We consider 1 hidden layer and the number of neurons of this hidden layer is arbitrarily set to $K = 12$.
- In both continuous and discrete time cases, we set x as a rectangle of 21 $\times$ 21 centered at 0. Therefore, $V(X)$ and $\dot{V}(X)$ are evaluated in 441 points in the range of each system.
- The number of searched variables α (29) is: $K \times (2n) + 1 = 49$, and each of them has its search space interval arbitrarily defined by [-4; 4].
- The optimization method to calculate the weights of the neural network to compute a Lyapunov function is the Genetic Algorithm from the Global Optimization Toolbox in Matlab, used, for instance, in (Krishna et al., 2019). Other parameters that are not mentioned are the default value in GA, like the mutation rate for instance.

5.2 Continuous Time System

Example 1

Let us consider the following system:

$$\begin{cases} \dot{x}_1 &= -tan\, x_1 + {x_2}^2 \\ \dot{x}_2 &= -x_2 + x_1 \end{cases}$$

The ranges for x_1 and x_2 are $x_1 \in [-1,1]$ and $x_2 \in [-1,1]$. The stability of the origin is considered and the Figures 1 and 2 show the result.

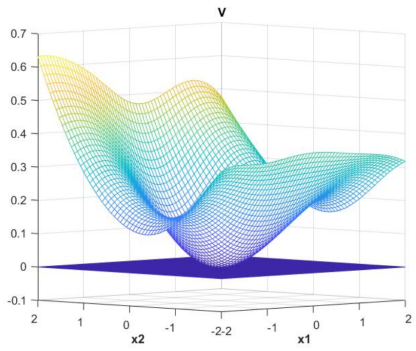

Figure 1: The constructed Lyapunov function for this system.

The proposed method is compared to the method from (Banks, 2002), based on genetic programming techniques. From Figures 1 and 2, we can easily check that our method provides a larger $\hat{D}$ than the one found in the paper, which shows the efficiency of our approach. Therefore, the origin of the system is asymptotically stable.

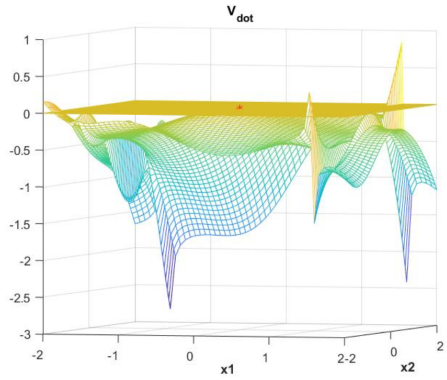

Figure 2: The time derivative of the constructed Lyapunov function.

5.3 Discrete Time Systems

Example 2

Let us consider the following system:

$$\begin{cases} x^+ &= \frac{1}{2}x + x^2 - y^2 \\ y^+ &= -\frac{1}{2}y + x^2 \end{cases}$$

The ranges for x and y are $x \in [-1,1]$ and $y \in [-1,1]$. The stability of the origin is considered and the Figures 3 and 4 show the result. The Figure 4 shows the function $V(X(k+1)) - V(X(k))$ of the system.

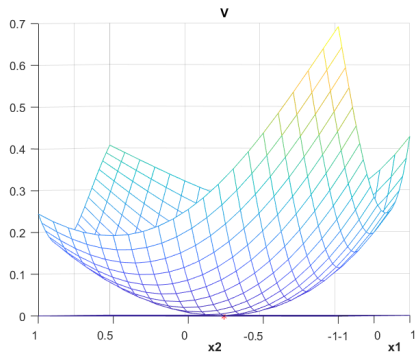

Figure 3: The constructed Lyapunov function for this system.

We can easily check that the underlying function expressed by the network input-output relation is a Lyapunov function for this system. Therefore, the origin of the system is asymptotically stable.

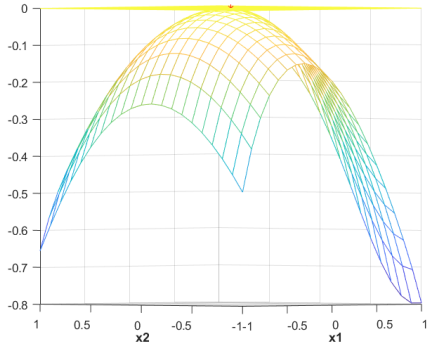

Figure 4: The difference function of the constructed Lyapunov function.

Example 3

Let us consider the following system:

$$\begin{cases} x^+ &= -0.125y - 0.125(1-x^2-y^2)x \\ y^+ &= 0.125x - 0.125(1-x^2-y^2)y \end{cases}$$

The ranges for x and y are $x \in [-0.75, 0.75]$ and $y \in [-0.75, 0.75]$. The stability of x_e = [0; 0] is considered and the Figures 5 and 6 show the result. The Figure 6 shows the function $V(X(k+1)) - V(X(k))$ of the system.

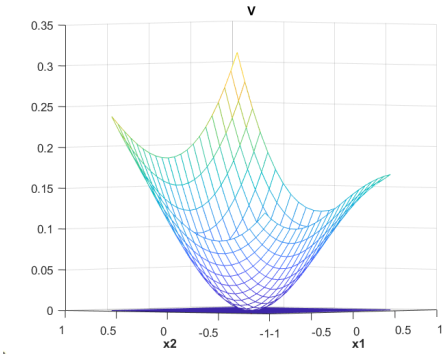

Figure 5: The constructed Lyapunov function for this system.

We can easily check that the underlying function expressed by the network input-output relation is a Lyapunov function for this system. Therefore, the equilibrium point x_e = [0; 0] of the system is asymptotically stable.

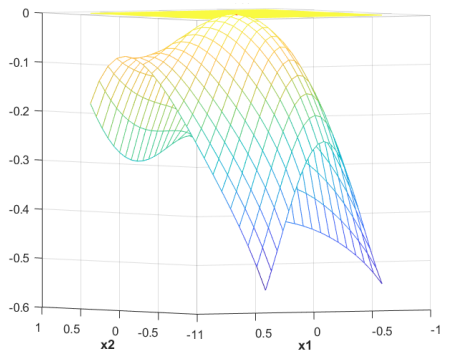

Figure 6: The difference function of the constructed Lyapunov function.

5.4 Performance Measurement

Since the tested optimization algorithm is stochastic, a statistical analysis of the results is required. Thus, the performance measurement rule is as follows.

The cost function Q is defined in III.B. Each test of each system is subjected to 10 successive runs: we note the minimum value of Q obtained (Q_{min}), the mean value of Q (Q_{mean}), its standard deviation (Q_{std}) and finally, the average calculation time (t_{cpu}) taken to perform these 10 runs. The results are presented in Table 1.

$$\begin{cases} Q_{min} = \min\limits_{i=1,\ldots,nruns} Q_i \\ Q_{mean} = \dfrac{1}{n}\sum\limits_{i=1}^{nruns} Q_i \\ Q_{std} = \sqrt{\dfrac{1}{n}\sum\limits_{i=1}^{n\ runs} (Q_i - Q_{mean})^2} \end{cases} \tag{30}$$

Table 1: Algorithm Performance Measurement.

	Q_{min}	Q_{mean}	Q_{std}	$t_{cpu/run(mn)}$
Example 1	- 1	- 0.95	0.16	68.3
Example 2	- 1	- 0.85	0.23	154.4
Example 3	- 1	- 0.82	0.32	149.7

According to the definition of problems Q, a neural network candidate is a Lyapunov function if $Q <$ 0 in Table 1. In these cases, we have the best domain if Q = - 1.Therefore, we see that our extension of our approach to discrete time cases has very good results. The optimization algorithm finds a Lyapunov function 26 times out of 30 runs with these 3 examples. The best runs lead to the figures presented in section 5.

6 Conclusions

In this paper, we investigate a new approach for the construction of a Lyapunov function modelled by a Neural network with the optimization of the domain of attraction for discrete time systems. We propose to use a constrained method such that the weights of the neural network are calculated in a way that is proven mathematically that the result function is a Lyapunov function while maximizing the DOA. Future work deals with proving more complex stability properties, such as the exponential, the robust stability and the Input-to-state stability (ISS). Besides, future work deals with developing this algorithm for real complex systems that we can find in industrial frameworks.

REFERENCES

Argáez, C., Giesl, P., and Hafstein, S. F. (2018). Iterative construction of complete Lyapunov functions. In *SIMULTECH*, pages 211–222.

Banks, C. (2002). Searching for Lyapunov functions using genetic programming. *Virginia Polytech Institute, unpublished*.

Bocquillon, B., Feyel, P., Sandou, G., and Rodriguez-Ayerbe, P. (2020). Efficient construction of neural networks Lyapunov functions with domain of attraction maximization. In *17th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*.

Feyel, P. (2017). *Robust control optimization with metaheuristics*. John Wiley & Sons.

Giesl, P. (2007). On the determination of the basin of attraction of discrete dynamical systems. *Journal of Difference Equations and Applications*, 13(6):523–546.

Giesl, P. and Hafstein, S. (2015). Review on computational methods for Lyapunov functions. *Discrete and Continuous Dynamical Systems-Series B*, 20(8):2291–2331.

Khalil, H. K. and Grizzle, J. W. (2002). *Nonlinear systems*, volume 3. Prentice Hall Upper Saddle River, NJ.

Krishna, A. V., Sangamreddi, C., and Ponnada, M. R. (2019). Optimal design of roof-truss using ga in matlab. *i-Manager's Journal on Structural Engineering*, 8(1):39.

Lyapunov, A. M. (1892). The general problem of the stability of motion. *International journal of control*, 55(3):531–534.

Milani, B. E. (2002). Piecewise-affine Lyapunov functions for discrete-time linear systems with saturating controls. *Automatica*, 38(12):2177–2184.

Panikhom, S. and Sujitjorn, S. (2012). Numerical approach to construction of Lyapunov function for nonlinear stability analysis. *Research Journal of Applied Sciences, Engineering and Technology*, 4(17):2915–2919.

Petridis, V. and Petridis, S. (2006). Construction of neural network based Lyapunov functions. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 5059–5065. IEEE.

Prokhorov, D. V. (1994). A Lyapunov machine for stability analysis of nonlinear systems. In *Proceedings of 1994 IEEE International Conference on Neural Networks*, volume 2, pages 1028–1031. IEEE.

Serpen, G. (2005). Empirical approximation for Lyapunov functions with artificial neural nets. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, volume 2, pages 735–740. IEEE.