Efficient Construction of Neural Networks Lyapunov Functions 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]*L2S, CentraleSupélec, CNRS, Université Paris-Saclay, 3 rue Joliot Curie, 91192 Gif-Sur-Yvette, France*

Keywords: Lyapunov Function, Domain of Attraction, Optimization, Neural Network, Nonlinear System.

Abstract: This work deals with a new method for computing Lyapunov functions represented by neural networks for autonomous nonlinear systems. Based on the Lyapunov theory and the notion of domain of attraction, we propose an optimization method for determining a Lyapunov function modelled by a neural network while maximizing the domain of attraction. The potential of the proposed method is demonstrated by simulation examples.

1 INTRODUCTION

In 1892, Lyapunov introduced a way to prove stability of mechanical nonlinear systems (Lyapunov, 1892). He defined a scalar function inspired by a classical energy function, which has 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. While Lyapunov theory provides powerful guarantees concerning equilibrium points' stability once an appropriate function is identified, there is no general method for constructing such a function, call in the sequel a Lyapunov function.

Various methods to compute Lyapunov functions have surfaced in the literature, especially with the emergence of efficient optimization methods. Attempts have been made to compute the best quadratic Lyapunov function, with (Panikhom and Sujitjorn, 2012) for example. However, these methods are too conservative in case of industrials complex systems. Several other computational approaches to construct complete Lyapunov functions have been published, for instance (Argáez et al., 2018), where the authors present a new iterative algorithm that avoids obtaining trivial solutions when constructing complete 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. Although efficient, this method looks like difficult to implement for real complex systems that we can find in industrial framework in which flexibility is needed. Endless, the survey (Giesl and Hafstein, 2015) has brought different methods and described the state of art of the vaste variety of methods to compute Lyapunov functions of various kinds of systems. It proposes conservative methods when the system is complex and highly non-linear.

However, in our opinion, the emergence of Artificial Intelligence tool such as Machine Learning and Neural Networks seems to be a good and powerful alternative for industrials to justify and then certificate quickly complex and intelligent systems that we can find in aero for instance. One of the first paper 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. (Banks, 2002) suggests a Genetic Programming for computing Lyapunov functions. However, the Lyapunov functions computed may have locally a conservative behavior. We overcome all these limits using Neural Networks. Neural Networks are widely used in a variety of applications, such as in image classification and in natural language processing. In general, neural networks are powerful regressors, and thus lend themselves to the approximation of Lyapunov function. In the literature, one can find other works using neural network to construct or approximate a Lyapunov function (Serpen, 2005; Long and Bayoumi, 1993) and the paper (Petridis and Petridis, 2006) proposes an interesting and promising approach for the con-

Bocquillon, B., Feyel, P., Sandou, G. and Rodriguez-Ayerbe, P.
Efficient Construction of Neural Networks Lyapunov Functions with Domain of Attraction Maximization.
DOI: 10.5220/0009883401740180
In *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2020)*, pages 174-180
ISBN: 978-989-758-442-8

struction of Lyapunov functions represented by neural networks. In this last paper, the author presents an approach for constructing a Lyapunov function which is modelled by a neural network using the well-know universal approximation capability of neural networks. Although effective, the main difficulty of this approach is related to the absence of the estimation of the DOA and the use of barrier functions, which do not ensure that the final result is a Lyapunov function : a post verification is needed.

To enhance the performance and usability of the Lyapunov function based neural network approach, we 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. Thanks to Chetaev's Instability Theory (Chetaev, 1934), instability is easy to prove as well with a similar optimization scheme.

The paper is structured as follows. First, we introduce preliminaries related to Lyapunov theory and Chetaev's Instability Theory. In section 3, we present the proposed algorithm to compute a candidate Lyapunov function through optimal neural network weights calculation while maximizing the DOA. We use the same development to prove instability and find a Chetaev function. Finally, we illustrate our method by three examples, exhibiting much than satisfactory results.

2 THEORETICAL BACKGROUND

In this section, we introduce notations and definitions, and present some key results needed for developing the main contribution of this paper. 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:

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

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

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. Thus D has to be as large as possible.

Theorem 2.2 (Chetaev's Instability Theory)(Chetaev, 1934)**.** Let $X_e = 0$ be an equilibrium point for (1). Let $V: U \to \mathbb{R}$ be a continuously differentiable function:

$$V(0) = 0 \text{ and } V(X) > 0 \text{ in } U - \{0\} \tag{5}$$

$$\dot{V}(X) > 0 \text{ in } U - \{0\} \tag{6}$$

then $X_e = 0$ is unstable.

Crucial Condition: $\dot{V}$ must be positive in the entire set where $V > 0$.

3 PROPOSED ALGORITHM

3.1 Neural Network Formalism

We keep the same formalism than in (Petridis and Petridis, 2006) for the construction of Lyapunov functions represented by neural network.

Let us consider the autonomous system in (1), in which we assume that the equilibrium point, the stability of which we wish to investigate, is the point 0 (X=0). Therefore,

$$f(0) = 0 \tag{7}$$

Suppose $V(X)$ is a scalar, continuous and differentiable function and its derivative respect to time is given in (8).

$$G(X) = \frac{dV}{dt} = \sum_{j=1}^{n} \frac{\partial V}{\partial x_j} f_j(X) \tag{8}$$

with $X = [x_1,...,x_n]^T$.
The Lyapunov function is modelled by a neural network whose weights are calculated in such a way that is proven mathematically that the resulting neural network implements indeed a Lyapunov function, showing the stability in the neighbourhood of 0, which is assumed to be the equilibrium point of f. 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 and 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.

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

$$V(X) = \sum_{i=1}^{K} a_i \sigma(\nu_i) + \theta \tag{9}$$

$$\nu_i = \sum_{j=1}^{n} w_{ij} x_j + h_i \tag{10}$$

From (2) and (3), sufficient conditions for the point 0 of system (1) to be 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:

$$\begin{aligned} V_{qr} &= \left.\frac{\partial^2 V}{\partial x_q \partial x_r}\right|_{X=0} \\ &= \sum_{i=1}^{K} a_i \left.\frac{d^2\sigma(\nu_i)}{d\nu_i^2}\right|_{X=0} \left.\frac{\partial \nu_i}{\partial x_r}\right|_{X=0} w_{qi} \\ &= \sum_{i=1}^{K} a_i \left.\frac{d^2\sigma(\nu_i)}{d\nu_i^2}\right|_{X=0} w_{ri} w_{qi} \end{aligned} \tag{11}$$

$$\begin{aligned} G_{lp} = &\sum_{j=1}^{n} \left(\sum_{i=1}^{K} a_i \left.\frac{d^2\sigma(\nu_i)}{d\nu_i^2}\right|_{X=0} w_{ji} w_{li} \right) J_{jp} + \\ &+ \sum_{j=1}^{n} \left(\sum_{i=1}^{K} a_i \left.\frac{d^2\sigma(\nu_i)}{d\nu_i^2}\right|_{X=0} w_{ji} w_{pi} \right) J_{jl} \end{aligned} \tag{12}$$

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,

$$H^{\mathrm{V}} = [V_{qr}(X=0)] \quad V_{qr} \text{ is given by (10)} \tag{13}$$

$$H^{\mathrm{G}} = [G_{lp}(X=0)] \quad G_{lp} \text{ is given by (11)} \tag{14}$$

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

$$(t1) \sum_{i=1}^{K} a_i \sigma(h_i) + \theta = 0. \tag{15}$$

$$(t2) \sum_{i=1}^{K} a_i (1 - tanh^2(h_i)) w_{qi} = 0 \quad \text{for} \quad q=1,...,n. \tag{16}$$

(t3) H^V as given by eq. (12) is positive definite.

(t4) H^G as given by eq. (13) is negative definite.

3.2 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 fitness function, Q, should be selected so that positivity and negativity respectively of H^V and H^G are constrained. The symetric 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 the eigenvalues of H^G.

3.2.1 DOA Maximization Problem

Conditions (a2) and (a4) prove the 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 to make D as large as possible. The idea behind the problem is to find the maximizing space D for which $V(X)$ is fully contained in the region of negative definiteness of $\dot{V}(X)$. In other words, the problem is to find the minimum level set where $V(X) < 0$ or $\dot{V}(X) > 0$.

Denote P = max ($ratio_v$, $ratio_{dv}$)
where $ratio_v$ are the number of points X where $V(X)<0$ and $ratio_{dv}$ are the number of points X where $\dot{V}(X)> 0$.

In order to determine P, we evaluate $V(X)$ and $\dot{V}(X)$ in a set of points to maximize the domain of attraction D. The set of points is constituted of a hypercube whose faces are gridded in order to cover a sufficiently large enough domain $\subset \mathbb{X}$, which contained the searched D. 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 Lyapunov during the optimization. An idea could be to analyze the first run of the optimization algorithm and larger the gradient is, the thinner the gridding has to be in the next runs.

3.2.2 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 post 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 Q needs to be minimized.

To this purpose, we set down:
$H^{V\prime} = H^{V} \times$ -1

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

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

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

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

We assume that α is the decision variables where:

$$\alpha = [w_{ji}, a_i, h_i, \theta] \tag{17}$$

Then, the fitness function to be minimised has the following form:

$$\min_{\alpha} Q$$
If $\bar{\lambda} > 0$
$Q = \bar{\lambda}$
Else
$$Q = -\frac{1}{P+1}$$

which is a similar formulation of the cost function that can be found can find in (Feyel, 2017) and has proven its efficiency. According to the definition of problem Q, a neural network candidate is a Lyapunov function if $Q < 0$. In this case, we have the best domain D if Q = - 1. 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 do have the certification that the eigenvalues are really positive and negative for H^V and H^G respectively. Finally, no parameter is needed.

3.3 Instability

In this section, we try to adapt Lyapunov methods for stability to an instability algorithm. Hence, it suffices to identify a suitable region in which both $V(X)$ and $\dot{V}(X)$ have the same sign by definition of (5) and (6). We can use the development in the previous sections considering H^V and H^G have to be negative definite. It requires opposite signs with respect to (5) and (6) but this convention has been accepted for computational simplicity. Indeed, using our method described previously, we can express this problem in the general form of an optimization problem in which the cost function $\mathcal{J}$ needs to be minimised, where $\mathcal{J}$ is defined below.

In order to have H^V and H^G negative, we replace (λ^v) by (- λ^v). Therefore, for an appropriate Chetaev function to be determined, values of the weights of the neural network should be calculated such that the conditions (t1)-(t2) are satisfied + H^V and H^G are negative definite.

This method, using an optimization problem whose decision variables are the same α vector as before, has the following form:

$$\min_{\alpha} \mathcal{J}$$
If $\bar{\lambda} > 0$
$\mathcal{J} = \bar{\lambda}$
Else
$$\mathcal{J} = -\frac{1}{P'+1}$$
where
$$\bar{\lambda} = \max\,(\bar{\lambda}^{v\prime},\bar{\lambda}^{g})$$

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

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

$$P' = \max\,(ratio_v', ratio_{dv}')$$

where $ratio_v{}'$ are the number of points X where $V(X)>0$ and $ratio_{dv}{}'$ are the number of points X where $\dot{V}(X)>0$.

According to the definition of problem $\mathcal{J}$, a neural network candidate is a Chetaev function if $\mathcal{J} < 0$. In this case, we have the best domain U if $\mathcal{J}$ = - 1.

4 SIMULATION RESULTS

In this section, we apply our approach to two different nonlinear systems: a damped pendulum and a nonlinear system containing the tangent function. Both of them are two-dimensional, so that the returned function can be plotted.

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

4.1 Parameter Settings

The parameters settings used in the tests were as follows:

- We consider 1 hidden layer and the number of neurons of this hidden layer was: $K = n \times (n+1) \times 2 = 12$.
- The number of variables was: $K \times (2n) + 1 = 49$.
- We evaluated $V(X)$ and $\dot{V}(X)$ to determine P at many points in order to maximize the domain of attraction D. The points consist in the grid set covering the domain $\mathbb{X}$: a rectangular of 21 $\times$ 21. Therefore, $V(X)$ and $\dot{V}(X)$ are evaluated in 441 points in the range of each system.
- We use [-4; 4] for the lower and upper bounds.
- The optimization method to calculate the weights of the neural network to compute a Lyapunov function is the GA from the Global Optimization Toolbox in Matlab, used, for instance, in the paper (Krishna et al., 2019). Other parameters whose are not mentioned are the default value in GA, like the mutation rate for instance.

4.2 Stability

4.2.1 Damped Pendulum

We start with a very simple problem for which the stability / instability region can be analytically defined. Indeed, for this example we know that the equilibrium point x_e = [0; 0] is stable and x_e = [π; 0] is unstable.

Thus, let us consider the following system :

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -sin\, x_1 - x_2 \end{cases}$$

The ranges for $\dot{x}_1$ and $\dot{x}_2$ are $\dot{x}_1 \in [-\pi/2, \pi/2]$ and $\dot{x}_2 \in [-1, 1]$. The stability of the origin is considered and the figures 1 and 2 show the result.

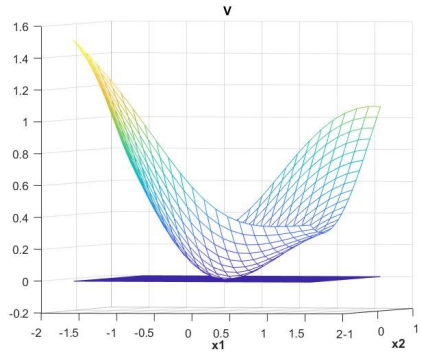

Figure 1: The constructed Lyapunov function for this system.

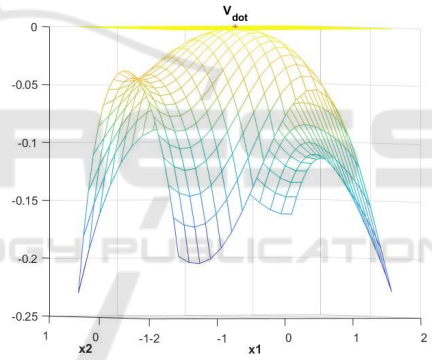

Figure 2: The time derivative of the constructed Lyapunov function.

We can easily check that the neural network is a Lyapunov function for this system. Therefore, the origin of the system is stable.

4.2.2 Nonlinear System Containing the Tangent Function

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 $\dot{x}_1$ and $\dot{x}_2$ are $\dot{x}_1 \in [-1, 1]$ and $\dot{x}_2 \in [-1, 1]$. The stability of the origin is considered and the figures 3 and 4 show the result. We can easily check that the neural network is a Lyapunov function for this system. Therefore, the origin of the system is stable.

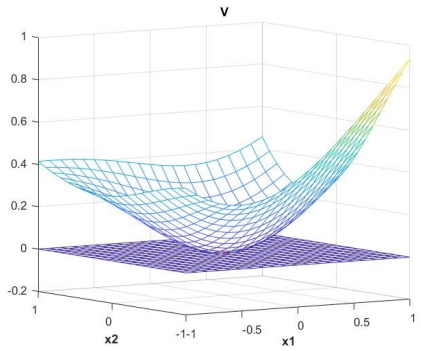

Figure 3: The constructed Lyapunov function for this system.

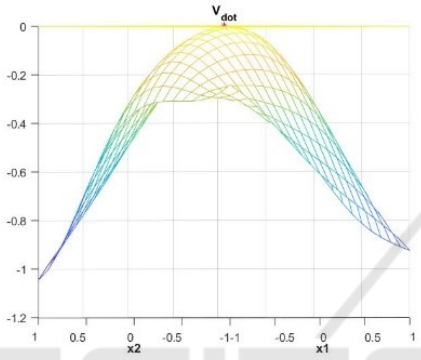

Figure 4: The time derivative of the constructed Lyapunov function.

4.3 Instability

4.3.1 Damped Pendulum

We use the same system than the first example but instead of considering the stability of the origin, we will consider the instability of the equilibrium point $x_e = [\pi; 0]$.

Let us consider the following system:

$$\begin{cases} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -sin\, x_1 - x_2 \end{cases}$$

The ranges for $\dot{x}_1$ and $\dot{x}_2$ are $\dot{x}_1 \in [\pi/2, -\pi/2]$ and $\dot{x}_2 \in [-1, 1]$. The instability of $x_e = [\pi; 0]$ is considered and the figures 5 and 6 show the result:

We can easily check that the neural network is a Chetaev function for this system. Therefore, the equilibrium point $x_e = [\pi; 0]$ of the system is unstable.

4.4 Performance Measurement

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

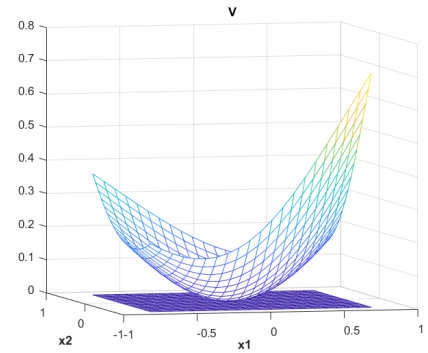

Figure 5: The constructed Chetaev function for this system.

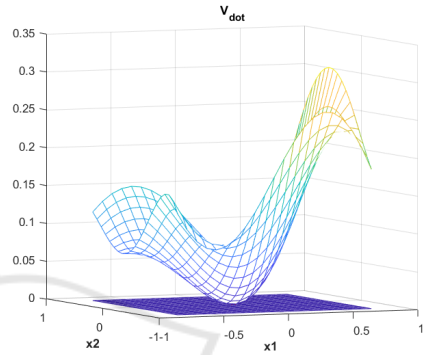

Figure 6: The time derivative of the constructed Chetaev function.

The cost function Q is defined in 3.2 for the stability cases and $\mathcal{J}$ in 3.3 for the instability case. We are going to use only the notation Q for the two costs functions to simplify the notation in this section. 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.

$$\begin{cases} Q_{min} = \min\limits_{i=1,\ldots,nruns} Q_i \\ Q_{mean} = \dfrac{1}{n} \sum\limits_{i=1}^{nruns} Q_i \\ Q_{std} = \sqrt{\dfrac{1}{n} \sum\limits_{i=1}^{n\ runs} (Q_i - Q_{mean})^2} \end{cases} \tag{18}$$

Table 1: Algorithm Performance Measurement.

	Q_{min}	Q_{mean}	Q_{std}	$t_{cpu/run(mn)}$
Stab_4.1.1	- 1	- 0.95	0.16	68.3
Stab_4.1.2	- 1	-1	0	52.6
Instability	- 1	-1	0	47.5

According to the definition of problems Q and $\mathcal{J}$, a neural network candidate is a Lyapunov function or a Chetaev function if $Q < 0$ in the table 1. In these cases, we have the best domain if Q = - 1.Therefore, we see that our approach has very good results. The optimization algorithm finds a Lyapunov function 29 times out of 30 runs with these 2 examples. The best runs lead to the figures presented in the section Simulation Results.

5 CONCLUSIONS

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

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*.

Chetaev, N. (1934). Un théoreme sur l'instabilité. In *Dokl. Akad. Nauk SSSR*, volume 2, pages 529–534.

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

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.

Long, Y. and Bayoumi, M. (1993). Feedback stabilization: Control lyapunov functions modelled by neural networks. In *Proceedings of 32nd IEEE Conference on Decision and Control*, pages 2812–2814. IEEE.

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

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 (ICNN'94)*, 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, 2005.*, volume 2, pages 735–740. IEEE.