Monograph:
Banks, S.P. and Harrison, R.F. (1989) Can Perceptrons find Lyapunov's Functions?-An Algorithmic Approach to Systems Stability. Research Report. Acse Report 365 . Dept of Automatic Control and System Engineering. University of Sheffield

Can Perceptrons find Lyapunov functions?—An algorithmic approach to systems stability.

S P Banks and R F Harrison
Department of Control Engineering, University of Sheffield
Mappin Street
Sheffield, S1 3JD

June 6, 1989

Research Report No. 365

1 Introduction

The stability theory of general systems has been a major topic of interest for control theorists and practitioners for many years—stability, of course, being a fundamental requirement of all control systems and indeed most systems if they are to serve any useful purpose. The recent resurgence of interest in self-organizing or adaptive systems has therefore naturally led people to ask questions about the stability of artificial neural networks, which are, for the systems theorist, simply non-linear dynamic systems.

A great variety of techniques for the study and prediction of stability have been developed, but the most important approach to this problem remains that introduced by A M Lyapunov [1] in 1907. The basis of this method is the association a Lyapunov function, V, to a particular system of the form:

$$\dot{x} = f(x) \;\; ; \;\; x \in R^n \tag{1.1}$$

such that

$$\begin{aligned} V(x) &> 0 \, , \; x \neq 0 \, , \; V(0) = 0 \\ &\text{and} \\ \dot{V}(x) &\triangleq \nabla_x V \dot{x} = \nabla_x V f(x) < 0 \end{aligned} \tag{1.2}$$

The famous theorem of Lyapunov then states (roughly) that the system (1.1) is (asymptotically) stable if and only if there exists a function V with the properties specified by (1.2).

The stability of linear (time-invariant) systems has been completely characterized by this method; a Lyapunov function being given by a simple, quadratic form. Several, special types of non-linear systems have also been considered and Lyapunov functions have been found using the specific properties of $f(x)$ which the system may possess. The general problem of finding a Lyapunov function for any (non-linear) system, however, remains open and to date no algorithm exists for the determination of a Lyapunov function for a given stable system.

It is interesting to note that Lyapunov's name has already entered the currency of neural network theory through such workers as Cohen and Grossberg [2] and Kosko [3] among others, who have successfully investigated the stability of certain classes of neural networks using Lyapunov theory.

In this paper we shall show that the problem of finding a Lyapunov function, if one exists, can be cast in a form suitable for solution by a simple neural network—namely Rosenblatt's single layer Perceptron [4]. Needless to say, the complete machine implementation of an algorithm which determines Lyapunov functions is of enormous importance in the field of systems theory and another, long standing problem has yielded to machine mathematics.

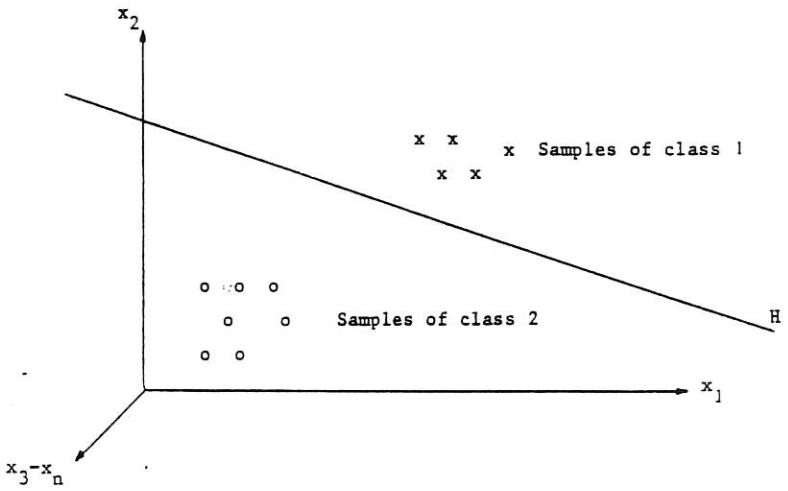

Figure 1: Sample distribution in pattern space

2 The Perceptron Algorithm

We shall begin by recalling the Perceptron algorithm in the form in which it will be applied in section 3, below. The algorithm was developed as a means of training pattern classifiers which form the basis of the decision-theoretic approach to pattern recognition. Each pattern sample is represented by a point in n-dimensional feature (or pattern) space, R^n. It is assumed that "neighbouring" samples represent the same pattern class, so that in a system with two pattern classes, we may have a set of samples as shown in figure 1. The classes are separated by a *decision boundary* which may be taken to be linear (in a sufficiently high dimensional space) given by the equation:

$$\begin{aligned} \sum_{i=1}^{n} w_i x_i + w_{n+1} &= 0 \\ \text{or} & \\ w \cdot x &= 0 \end{aligned} \tag{2.1}$$

where $w = (w_1, \ldots, w_{n+1})$ and x is the augmented vector $x = (x_1, \ldots, x_n, 1)^t$.

A pattern classification system is trained by presenting the samples to it in some sequence and adjusting the weight vector w recursively, until all samples are correctly classified. This is often implemented by means of the *Perceptron algorithm* [4] which may be written in the form:

Let $x^1, x^2, \ldots$ be the samples, in some order (if there are N samples then each sample is present in $\{x^1, x^2, \ldots, x^N\}$ and in $\{x^{N+1}, x^{N+2}, \ldots, x^{2N}\}$, *etc.*).

Define recursively:

$$\begin{aligned} w^{k+1} &= w^k + x^k \text{ if } x^k \text{ is in class 1 and } w^k \cdot x \text{ misclassifies } x^k \\ w^{k+1} &= w^k - x^k \text{ if } x^k \text{ is in class 2 and } w^k \cdot x \text{ misclassifies } x^k \\ w^{k+1} &= w^k \text{ if } x^k \text{ is correctly classified} \end{aligned}$$

Thus, the hyperplane H is moved about in R^n space until all samples are correctly classified. The celebrated perceptron algorithm convergence theorem [4] states that, if a separating hyperplane exists, then the algorithm converges in a finite number of steps.

3 Global Linearization of Lyapunov Theory

A full exposition of the following theory is given by Banks in [5].

Consider a stable polynomial system of the form:

$$\dot{x} = p_r(x) \;\; ; \;\; x \in R^n \tag{3.1}$$

where

$$p_r(x) = \sum_{i_1=0}^{m_1} \cdots \sum_{i_n=0}^{m_n} \alpha^r_{i_1 \ldots i_n} x_1^{i_1} \ldots x_n^{i_n}$$

Define the functions:

$$\phi_{i_1 \ldots i_n} = x_1^{i_1} \ldots x_n^{i_n} \tag{3.2}$$

and assume that a Lyapunov function, V exists in the form:

$$\begin{aligned} V &= \sum_{i_1=0}^{k_1} \cdots \sum_{i_n=0}^{k_n} v_{i_1 \ldots i_n} x_1^{i_1} \ldots x_n^{i_n} \\ &= \sum_{i_1=0}^{k_1} \cdots \sum_{i_n=0}^{k_n} v_{i_1 \ldots i_n} \phi_{i_1 \ldots i_n} \end{aligned} \tag{3.3}$$

From (3.1), (3.2) and (3.3) we have:

$$\dot{V} = \sum_{i_1=0}^{k_1} \cdots \sum_{i_n=0}^{k_n} v_{i_1 \ldots i_n} \dot{\phi}_{i_1 \ldots i_n} \tag{3.4}$$

where $\dot{\phi}_{i_1 \ldots i_n}$ is given by:

$$\dot{\phi}_{i_1 \ldots i_n} = \sum_{\ell=1}^{n} i_\ell x_1^{i_1} \ldots x_\ell^{i_\ell - 1} \ldots x_n^{i_n} \times \sum_{j_1=0}^{m_1} \cdots \sum_{j_n=0}^{m_n} \alpha^\ell_{j_1 \ldots j_n} x_1^{j_1} \ldots x_n^{j_n}$$

$$= \sum_{\ell=1}^{n} \sum_{j_1=0}^{m_1} \cdots \sum_{j_n=0}^{m_n} i_\ell \alpha^{\ell}_{j_1 \ldots j_n} x_1^{i_1+j_1} \ldots x_\ell^{i_\ell+j_\ell-1} \ldots x_n^{j_n}$$

$$= \sum_{\ell=1}^{n} \sum_{j_1=0}^{m_1} \cdots \sum_{j_n=0}^{m_n} i_\ell \alpha^{\ell}_{j_1 \ldots j_n} \phi_{i_1+j_1,\ldots,i_\ell+j_\ell-1,\ldots,j_n} \tag{3.5}$$

Note that (3.3) and (3.4) are linear forms in $(k_1+1) \times (k_2+1) \times \ldots \times (k_n+1)$-dimensional ϕ-space.

In order to simplify the notation we shall write

$$\begin{aligned} i &= (i_1, \ldots i_n) \\ k &= (k_1, \ldots k_n) \\ v_i &= v_{i_1 \ldots i_n} \\ \phi_i &= \phi_{i_1 \ldots i_n} \end{aligned}$$

In this notation (3.3) and (3.4) become:

$$V = \sum_{i=0}^{k} v_i \phi_i \tag{3.6}$$

$$\dot{V} = \sum_{i=0}^{k} v_i \dot{\phi}_i \tag{3.7}$$

respectively, where

$$\sum_{i=0}^{k} = \sum_{i_1=0}^{k_1} \cdots \sum_{i_n=0}^{k_n}$$

It should now be recalled that the coefficients v_i are unknown and so will correspond to the weight vector w in (2.1). In order to interpret the problem of finding suitable coefficients v_i using the perceptron, we must provide the algorithm with samples of ϕ_i and $\dot{\phi}_i$. This can be done by choosing N "well-spaced" points, $x(k); (1 \leq k \leq N)$ in $\Omega \in R^N$, where Ω is contained in the domain of attraction of $0 \in R^n$. Then we define:

$$\psi_i(k) = \phi_i(x(k)) = x_1^{i_1}(k) \ldots x_n^{i_n}(k) \tag{3.8}$$

$$\xi_i(k) = \dot{\phi}_i(x(k)) \tag{3.9}$$

for $1 \leq k \leq N$, where the right hand side of (3.9) is given by (3.5). Let H_v be the hyperplane in R^L, defined by:

$$H_v = \{y \in R^L : \sum_{i=0}^{k} v_i y_i = 0\}$$

where $L = \Pi_{i=1}^{n}(k_i + 1)$. Also, let

$$H_v^+ = \{y \in R^L : \sum_{i=0}^{k} v_i y_i > 0\}$$

$$H_v^- = \{y \in R^L : \sum_{i=0}^{k} v_i y_i < 0\}$$

Then we wish to find v so that the samples $\{\psi(1), \ldots, \psi(N)\}$ lie in H_v^+ while the samples $\{\xi(1), \ldots, \xi(N)\}$ lie in H_v^-, where

$$\psi(j) = (\psi_i(j)) \in R^L$$
$$\xi(j) = (\xi_i(j)) \in R^L$$

This leads to the following Perceptron algorithm for determining Lyapunov functions:

$$\begin{aligned} v^{k+1} &= v^k + \eta^k \text{ if } \eta^k = \psi(\ell) \in H_v^- \\ v^{k+1} &= v^k - \eta^k \text{ if } \eta^k = \xi(m) \in H_v^+ \\ v^{k+1} &= v^k \text{ if } (\eta^k = \psi(\ell) \in H_v^+) \text{ or } (\eta^k = \xi(m) \in H_v^-) \end{aligned}$$

(for some ℓ and m).

Here, η^k is the k^{th} training sample and is either a member of class 1, *ie* $\psi(\ell)$ for some ℓ or a member of class 2, *ie* $\xi(m)$ for some m. By the general Perceptron convergence theorem, this algorithm will converge in finite time if the original system is stable, since by Lyapunov's main stability theorem citelyapunov, a Lyapunov function must exist and hence a hyperplane H_v also exists, which separates the samples. The value

$$v = \lim_{k \to \infty} v^k$$

is the required parameter vector which will completely specify V as in (3.3).

4 Examples

In this section we shall give three simple examples of well-known systems and compare the results with known Lyapunov functions. We shall assume in all cases that V can be written in the form:

$$V = \sum_{i=0}^{4} \sum_{j=0}^{2} v_{ij} x_1^i x_2^j$$

with $v_{00} = 0$. In general, of course, we must use higher order terms in order to obtain a Lyapunov function.

i,j	v_{ij}
0,1	-1.182×10^{-11}
0,2	$+9.200 \times 10^{-1}$
1,0	-1.000×10^{-1}
1,1	$+4.700 \times 10^{-1}$
1,2	-1.010×10^{-1}
2,0	$+8.600 \times 10^{-1}$
2,1	$+1.940 \times 10^{-1}$
2,2	$+1.226 \times 10^{-1}$
3,0	-1.790×10^{-1}
3,1	$+8.370 \times 10^{-2}$
3,2	-4.407×10^{-2}
4,0	$+2.020 \times 10^{-1}$
4,1	-3.140×10^{-2}
4,2	$+3.110 \times 10^{-2}$

Table 4.1

i,j	v_{ij}
0,1	-2.700×10^{-2}
0,2	$+9.154 \times 10^{-1}$
1,0	-1.819×10^{-12}
1,1	$+1.945 \times 10^{-1}$
1,2	-2.834×10^{-2}
2,0	$+4.300 \times 10^{-1}$
2,1	-1.244×10^{-1}
2,2	$+1.010 \times 10^{-1}$
3,0	-1.020×10^{-1}
3,1	$+5.329 \times 10^{-2}$
3,2	-2.277×10^{-2}
4,0	$+6.850 \times 10^{-2}$
4,1	-3.114×10^{-2}
4,2	$+1.883 \times 10^{-2}$

Table 4.2

i,j	v_{ij}
0,1	-4.547×10^{-12}
0,2	$+7.600 \times 10^{-1}$
1,0	$+2.074 \times 10^{-2}$
1,1	$+7.792 \times 10^{-2}$
1,2	$+4.207 \times 10^{-3}$
2,0	$+8.594 \times 10^{-1}$
2,1	-1.019×10^{-1}
2,2	$+8.419 \times 10^{-2}$
3,0	-3.526×10^{-1}
3,1	$+4.806 \times 10^{-2}$
3,2	-1.740×10^{-2}
4,0	$+2.689 \times 10^{-1}$
4,1	-3.305×10^{-2}
4,2	$+1.655 \times 10^{-2}$

Table 4.3

4.1 A damped linear oscillator

Consider first the simple linear system

$$\ddot{x} + \dot{x} + x = 0$$

Written in phase-variable form this becomes

$$\dot{x}_1 = x_2 \quad , \quad \dot{x}_2 = -x_2 - x_1$$

and represents a damped harmonic oscillator. We can choose the Lyapunov function $V = x_1^2 + x_2^2$ since then

$$\begin{aligned} \dot{V} &= 2x_1\dot{x}_1 + 2x_2\dot{x}_2 \\ &= 2x_1x_2 + 2x_2(-x_2 - x_1) \\ &= -2x_2^2 \end{aligned}$$

Note, however, that $\dot{V} \not< 0$ away from the origin and so we must use LaSalle's invariance principle [6] in connection with this Lyapunov function. The algorithm derived in Section 3 was run for (x_1, x_2) in the square $[-0.5, 0.5] \times [-0.5, 0.5]$ with 36 evenly spaced points $(x_1(k), x_2(k))$. The coefficients v_{ij} were found and are given in table 4.1. The resulting functions V and $\dot{V}$ are shown in figure 2. Note that the Lyapunov function and its derivative are not symmetric in this case (largely because of the order in which the samples are presented).

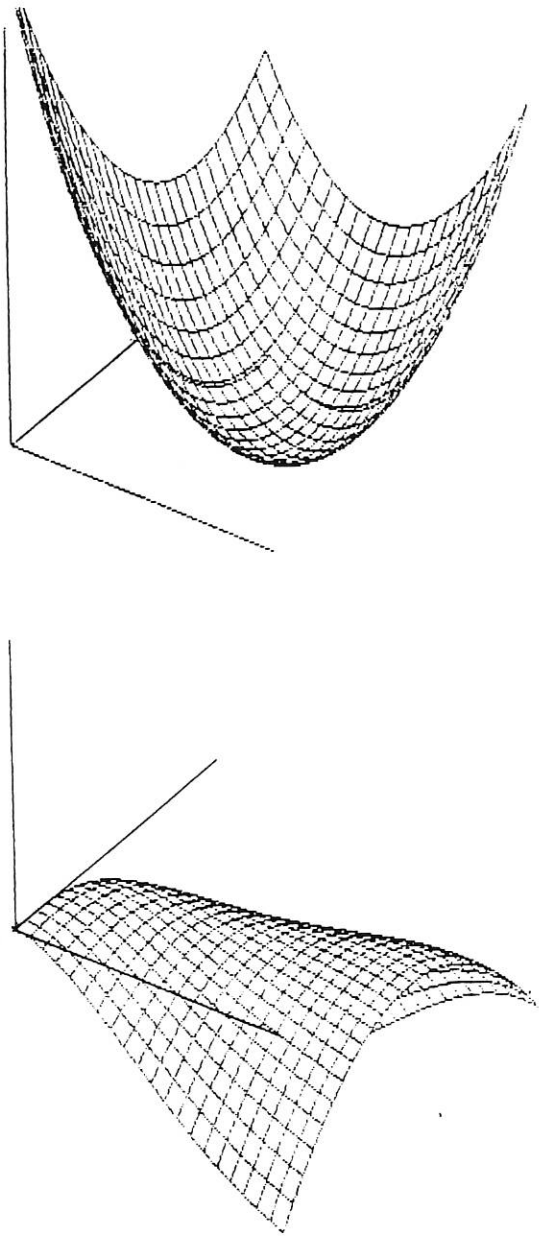

Figure 2: $V(\boldsymbol{x})$ and $\dot{V}(\boldsymbol{x})$ for the linear oscillator

4.2 A non-linear oscillator

An example which has been used to illustrate the variable gradient method of Schultz and Gibson [7] is given by:

$$\dot{x}_1 = x_2 \ , \ \dot{x}_2 = -x_1^3 - x_2$$

A Lyapunov function is found by this method to be:

$$V = \frac{x_1^2 + x_1^4}{2} + x_1 x_2 + x_2^2$$

Using our algorithm we obtain the coefficients v_{ij} as in the previous example, which are given in table 4.2. The functions V and $\dot{V}$ are shown in figure 3.

4.3 A non-polynomial system

The method will work even if $f(x)$ does not have a polynomial form. Thus, consider the system:

$$\begin{aligned} \dot{x}_1 &= -\sin x_1 + x_2 \triangleq f_1 \\ \dot{x}_2 &= x_1 - 2x_2 \triangleq f_2 \end{aligned}$$

By Krasovskii's theorem [8], this system has the Lyapunov function $V = f_1^2 + f_2^2$, provided that $\cos x_1 > 1/2$.

As before the algorithm determines the values shown in table 4.3, and the functions V and $\dot{V}$ are shown in figure 4. The Lyapunov function given by our method, *viz*

$$V = \sum_{i=0}^{4} \sum_{j=0}^{2} v_{ij} x_1^i x_2^j$$

where the coefficients are as in table 4.3, should be compared with that given by Krasovskii's method, *ie*

$$V_K = (-\sin x_1 + x_2)^2 + (x_1 - 2x_2)^2$$

5 Conclusions

In this paper we have developed a machine implementable algorithm for determining Lyapunov functions, which solves a long standing problem in systems theory. The method is based on the Perceptron algorithm and so is easily implemented by a neural network.

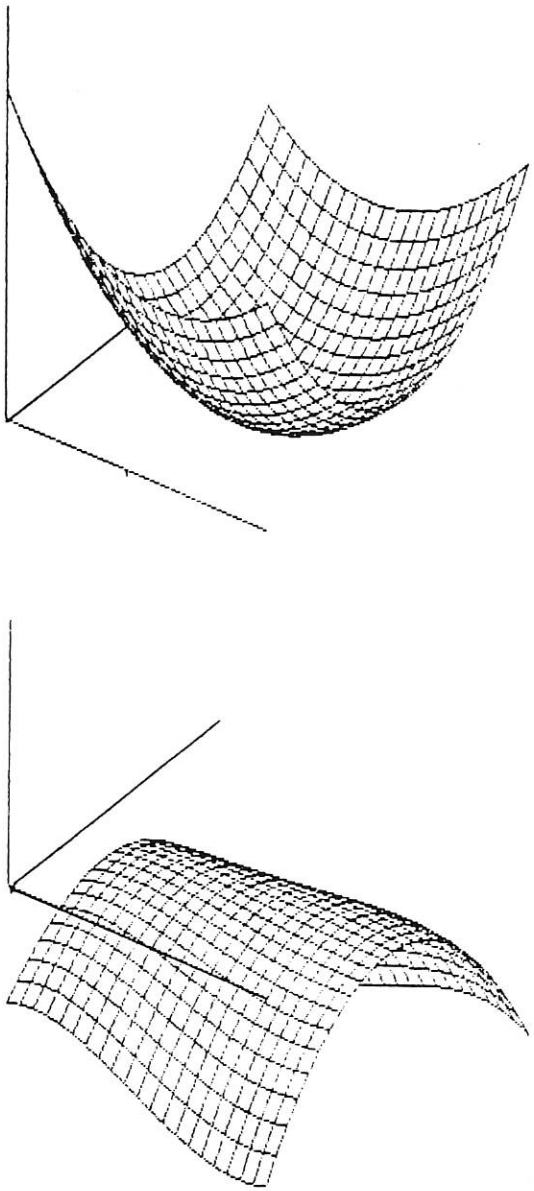

Figure 3: $V(\boldsymbol{x})$ and $\dot{V}(\boldsymbol{x})$ for the non-linear oscillator

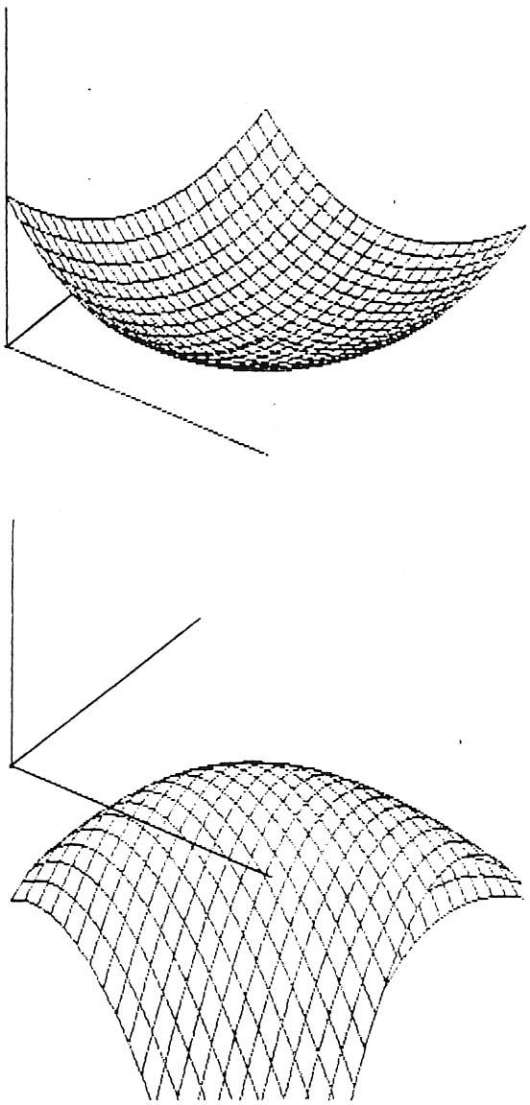

Figure 4: $V(\boldsymbol{x})$ and $\dot{V}(\boldsymbol{x})$ for the non-polynomial system

References

[1] A M Lyapunov, 1907, "Le problème général de la stabilité du mouvement" *Ann Fac Sci Toulouse*, vol 9, pp 203–474.

[2] M A Cohen and S Grossberg, 1983, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks" *IEEE Trans Syst Man Cybern*, vol SMC-13, pp 815–826.

[3] B Kosko, 1987, "Adaptive bi-directional associative memories" *Appl Opt*, vol 26, no 23, pp 4947–4860.

[4] F Rosenblatt, 1962, "Principles of neurodynamics" Spartan Books, New York.

[5] S P Banks, 5. S P Banks, 1988, "Mathematical theories of non-linear systems" Prentice Hall, England.

[6] J P LaSalle, 1960, "Some extensions of Lyapunov's second method" *Trans IRE*, vol CT-7, no 4, pp 520–527.

[7] D G Schultz and J E Gibson, 1962, "The variable gradient method for generating Lyapunov functions" *Trans AIEE pt 11*, vol 81, p 203.

[8] N N Krasovskii, 1954, "On the stability in the large of a solution of a system of non-linear differential equations" *Prikl Mat Mekh* vol 18, pp 735–737.