Technical University of Crete

Electronics and Computer Engineering Department

Diploma thesis

‘’Applications of Neural Networks in Optimization problems with constraints for Lyapunov Based stability’’

Kagkarakis Leonidas

Advisor

Prof. Manolis Christodoulou

Examination Committee

Prof. Manolis Christodoulou

Assoc. Prof. Dionisios Pnevmatikatos

Prof. Boutalis Ioannis

Acknowledgments

First of all I would like to thank my advisor Prof. Manolis Christodoulou for all his support and useful advices during the project and the other two members of the examination committee Prof. Dionisios Pnevmatikatos and Prof. Ioannis Boutalis.

Also, in that respect, I would like to thank all of my friends for what we have been through all these years. They stood by my side all these years and I shared with them the best seven years of my life so far.

Malafouris Zafeiris, Karakasiliotis Kostas, Skourtis Vasilis, Spanos Nikolaos, Zotos Alexandros, Koufidakis Manolis, Afratis Panagiotis

To my family

Contents

Table of contents

Introduction

There are many techniques for studying and analyzing stability of linear and non linear control systems. In this study we shall focus on linear systems but we can show that nonlinear, when are amenable to local linearization, can also be studied as if they were linear systems. Over the last ten years many techniques have been used for analyzing stability.

When the state space representation of linear systems was established in the late 1960s, the study of stability focused on more modern techniques as it is the Lyapunov Criterion, which we shall see later.

The Lyapunov criterion, however, leads to a solution of matrix equations which in many cases as in the case of discrete time linear system is nonlinear. Solving nonlinear matrix equations require the usage of .sophisticated and considerable computational processing and must be done before hand if the knowledge of stability of the system is of importance in the problem under consideration. We are, therefore, tempted to seek techniques which could lead to on line analyses. Those are a class of Neural networks which are based on various minimization methods [1] , [12] – [21]. In this study we shall show that these types of Neural networks can lead to an acceptable analyses for the case of Lyapunov based stability [1], [2], [4] – [11], [22], [23]. Chapter 1 analyzes the analog Neural Networks, Chapter 2 covers the theoretical basis of Lyapunov stability and the chapter 3 includes implementations of some models according to Lyapunov stability theory using Neural Networks.

Chapter 1

Unconstrained Optimization Algorithms

And

Neural Network Model

1.1 Necessary and Sufficient Conditions For an Extremum

Let us consider the following unconstrained optimization problem [1]:

Find a vector $x \in R^n$ that minimizes the real valued scalar function

$E = E(x)$

The function above is called the cost, objective or energy function and x is a non-dimensional vector called the design vector. Minimizing a function, is the same as maximizing the negative of the function, so there is no loss of generality in our considerations.

The point x^* is a global minimum for $E(x)$ if $E(x^*) \leq E(x)$ for all $x \in R^n$, and a strict local minimum if the relation $E(x^*) \leq E(x)$ holds for a ball $B(x^*; \varepsilon)$.

Assuming that the first and second derivatives of $E(x)$ exist, a point x^* is a strict local minimum of $E(x)$ if the gradient is zero ($\nabla E(x^*) = 0$) and the Hessian matrix is positive definite.

The above statement can be formulated as a theorem on necessary and sufficient conditions for a strict local minimum.

Theorem 1*: Let* $\nabla^2 E(x)$ *be nonsingular for point* x^**. Then we have* $E(x^*) < E(x)$ *for every x in neighborhood* $0 < \| x - x^* \| < \varepsilon$ *with some* $\varepsilon > 0$ *if* $(\nabla E(x^*) = 0)$ *and* $\nabla^2 E(x^*)$ *is symmetric and positive definite.*

1.2 Dynamic Gradient Systems

A very broad and perhaps the most important class of methods for unconstrained optimization is that which is based on the so called gradient descent methods. Virtually all of the gradient descent methods have their origin in standard methods known as methods of steepest descent (also called dynamic gradient systems) and Newton's methods. These methods transform the minimization problem into an associated system of first order ordinary differential equations

$$\frac{dx_j}{dt} = -\sum_{i=1}^{n} \mu_{ji} \frac{\partial E}{\partial x_i}, \tag{1.2-1}$$

With initial conditions $x_j(0) = x^{(0)}$ (starting point) which can be written in the compact matrix form

$$\frac{d\boldsymbol{x}}{dt} = -\boldsymbol{\mu}(\boldsymbol{x}, t)\nabla_{\boldsymbol{x}} E(\boldsymbol{x}), \tag{1.2-2}$$

where

$$\frac{d\boldsymbol{x}}{dt} = \left[\frac{dx_1}{dt}, \frac{dx_2}{dt}, \ldots, \frac{dx_n}{dt}\right]^T,$$

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$$

And $\mu(x,t)$ is a symmetric positive definite matrix on the time t and the vector $x(t)$. In order to find the desired vector x^* that minimizes the function $E(x)$ we need to solve the system of ordinary differential equations with initial conditions. This means

that the minimum of the energy function is determined by the following solution curve of the gradient system with

$$x^* = \lim_{t \to \infty} x(t).$$

In order to show that the above system of differential equations(1.2-1,1.2-2) leads to stable solutions, we will determine the time derivative of the energy (Lyapunov) function which has to be less or equal to zero.

$$\frac{dE}{dt} = \sum_{j=1}^{n} \frac{\partial E}{\partial x_j} \frac{dx_j}{dt} = - [\nabla_x E(x)]^T \mu(x, t) \nabla_x E(x) \leqq 0$$

(1.2-3)

The above relation guarantees that the energy function *E(x)* decreases in time and converges to a local minimum as $t \to \infty$. When the vector dx / dt = 0, then this implies that $\nabla E(x) = 0$ for the system of differential equations. Therefore, it follows that the equilibrium point coincides either with the minimum or with the inflection point of energy function *E(x).* Obviously the speed of convergence to the minimum depends on the choice of the entries of the matrix μ(*x, t*). In the simplest and best known form of this method (steepest descent method) the matrix μ(*x, t*) is reduced to the unity matrix multiplied by a positive constant μ_0. In this case the system simplifies to the form

$$\frac{dx_j}{dt} = - \mu_0 \frac{\partial E(x)}{\partial x_j}, \quad x_j(0) = x_j^{(0)}$$
$$(j = 1, 2, \ldots, n),$$

(1.2-4)

Where the positive coefficient μ_0 is called the learning parameter. It is interesting to note that the vectors dx/dt and $\nabla E(x)$ are opposite. Thus the time evaluation of *x(t)* will cause the minimization of *E(x)* as time goes on. The trajectory *x (t)* moves along the direction which has the sharpest rate of decrease and is called the direction of

steepest-descent. This concept is used to derive algorithms on which we can base Neural Networks designs.

1.3 Newton's Methods

The idea behind Newton's methods is that the energy function *E(x)* is approximated locally by a quadratic function when the former tends to be minimized.

The function *E(x)* near the point $x^{(k)}$ (k = 0,1, 2, ….) can be approximated by the Taylor series:

$$E(\boldsymbol{x}) \cong E(\boldsymbol{x}^{(k)}) + (\boldsymbol{x} - \boldsymbol{x}^{(k)})^{\mathrm{T}} \nabla E(\boldsymbol{x}^{(k)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(k)})^{\mathrm{T}} \nabla^2 E(\boldsymbol{x}^{(k)})(\boldsymbol{x} - \boldsymbol{x}^{(k)}). \tag{1.3-1}$$

The point $x^{(k+1)}$ that minimizes this series must satisfy the equation:

$$\nabla E(\boldsymbol{x}^{(k)}) + \nabla^2 E(\boldsymbol{x}^{(k)})(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}) = \mathbf{0} \tag{1.3-2}$$

Since $\theta E(x) / \theta x$ ($\nabla E(x)$) at a minimum point is equal to zero. If the inverse matrix of Hessian exists then the above equation can be written as follows:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - [\nabla^2 E(\boldsymbol{x}^{(k)})]^{-1} \nabla E(\boldsymbol{x}^{(k)}). \tag{1.3-3}$$

This equation is the standard form of the discrete-time Newton method.

The continuous time dynamical system corresponding to the above discrete time algorithm takes the following form as compared to (1.3-4):

$$\frac{d\boldsymbol{x}}{dt} = -\mu_0[\nabla^2 E(\boldsymbol{x})]^{-1}\nabla E(\boldsymbol{x}), \tag{1.3-4}$$

Where $\mu_0 > 0$.

We observe that in 1.3-4,we require that the inverse of the Hessian exists.

1.4 The Quasi-Newton Methods

In some problems only the gradient of the objective function *E(x)* can explicitly be determined. For such problems quasi-Newton methods can be used. The sequential quasi-Newton methods elaborate the differences of two successive iteration points and the difference of the corresponding gradients to approximate the inverse Hessian matrix. Thus these methods take advantage of the Newton method while using only first order information about the objective function. One of the most well known as well as powerful quasi-Newton methods is the *Broyden-Fletcher-Goldfrab-Shanno(BFGS)* algorithm, which can be formulated as:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \eta^{(k)}\boldsymbol{d}_k, \tag{1.4-1α}$$

$$\boldsymbol{d}_k \simeq \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} = -\boldsymbol{H}_k\nabla E(\boldsymbol{x}^{(k)}), \tag{1.4-1b}$$

$$\boldsymbol{y}_k := \nabla E(\boldsymbol{x}^{(k+1)}) - \nabla E(\boldsymbol{x}^{(k)}), \tag{1.4-1c}$$

$$\boldsymbol{H}_{k+1} = \left(\boldsymbol{I} - \frac{\boldsymbol{d}_k\boldsymbol{y}_k^{\mathrm{T}}}{\boldsymbol{d}_k^{\mathrm{T}}\boldsymbol{y}_k}\right)\boldsymbol{H}_k\left(\boldsymbol{I} - \frac{\boldsymbol{y}_k\boldsymbol{d}_k^{\mathrm{T}}}{\boldsymbol{d}_k^{\mathrm{T}}\boldsymbol{y}_k}\right) + \frac{\boldsymbol{d}_k\boldsymbol{d}_k^{\mathrm{T}}}{\boldsymbol{d}_k^{\mathrm{T}}\boldsymbol{y}_k}, \tag{1.4-1d}$$

Where the learning rate $\eta^{(k)} \geqq 0$ is determined from one dimensional line search:

$$\eta^{(k)} = \arg \min_{\eta \geq 0} E[\boldsymbol{x}^{(k)} - \eta \boldsymbol{H}_k \nabla E(\boldsymbol{x}^{(k)})]$$

And H_k denotes the current approximation to $[\nabla^2 E(\boldsymbol{x})]^{-1}$ and the iterative procedure starts at an arbitrary point $x^{(0)}$ with an initial approximation H_0, usually with the identity matrix I.

In such a case, we can present another formula for the updating matrix H_k, which is the *Barnes-Rosen* formula [1].

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \frac{(\boldsymbol{d}_k - \boldsymbol{H}_k \boldsymbol{y}_k)(\boldsymbol{d}_k - \boldsymbol{H}_k \boldsymbol{y}_k)^{\mathrm{T}}}{(\boldsymbol{d}_k - \boldsymbol{H}_k \boldsymbol{y}_k)^{\mathrm{T}} \boldsymbol{y}_k} .$$

(1.4-2)

The variable metric methods associated with Quasi-Newton method provide many advantages over the Newton method by saving us from the trouble of deriving second order derivatives.

1.5 The conjugate Gradient Method

The conjugate gradient algorithm of *Fletcher* and *Reeves* requires a relatively simple modification of the discrete time steepest descent method and often enables us to dramatically improve the convergence rate. The conjugate gradient algorithm in its simplest form can be formulated as [1]:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \eta^{(k)} \boldsymbol{d}_k ,$$

$$\boldsymbol{d}_k := \beta_k \boldsymbol{d}_{k-1} - \nabla E(\boldsymbol{x}^{(k)}) ,$$

(1.5-1)

Where:

$$\beta_k = \frac{\| \nabla E(\boldsymbol{x}^{(k)}) \|_2^2}{\| \nabla E(\boldsymbol{x}^{(k-1)}) \|_2^2}$$

And

$$\eta^{(k)} = \arg \min_{\eta \geq 0} E(\boldsymbol{x}^{(k)} + \eta \boldsymbol{d}_k).$$

It can be noticed that the algorithm above utilizes information about the direction search d_{k-1} from the previous iteration in order to accelerate the convergence, and each search direction is conjugate if the objective function is quadratic. Theoretically the algorithm above will minimize a quadratic function in n or less iterations[1]. Using the theoretical background presented so far, we can show that these algorithms can be used to develop Neural Networks, which in terms can be used to analyze and solve Lyapunov Stability problem. Based on the above algorithm we can implement Neural Networks to solve complex problems on a quasi-on-time basis, which otherwise would require sophisticated computational techniques.

1.6 Basic Neuron Model -- McCulloch-Pitts Model

The basic artificial neuron can be modeled as a multi-input nonlinear device with weighted interconnections w_{ji} , also called synaptic weights or strengths (cf.fig 1.6a).

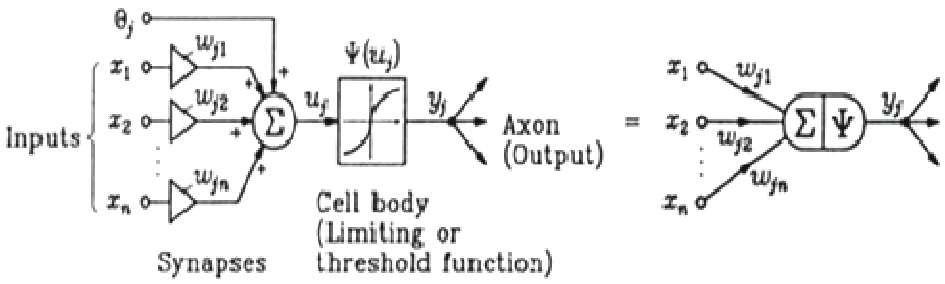


Fig. 1.6.a Simplified functional model of an artificial basic neuron cell and its symbol

The cell body is represented by a non linear limiting or threshold function $\psi(u_j)$. The simplest model of an artificial neuron sums the n weighted inputs and passes the result through a nonlinearity according to the equation

$$y_j = \Psi\left(\sum_{i=1}^{n} w_{ji}\, x_i + \Theta_j \right), \tag{1.6-1}$$

and

$$u_j = \left(\sum_{i=1}^{n} w_{ji}\, x_i + \Theta_j \right),$$

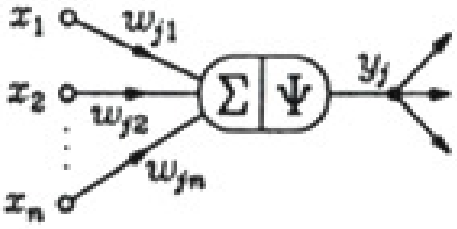
Where ψ is a limiting or threshold function, called an activation function, θ_j ($\in R^n$) is the external threshold also called an offset or bias, w_{ji} ,are the synaptics weights or strengths, x_i are the inputs (i = 1,2,3,…,n) is the number of inputs and y_j represents the output. Note that a threshold value θ_j may be introduced by employing an additional input x_0 equal *to ' 1'* and the corresponding weight w_{j0} equal to the threshold value. So we can write the above equation as:

$$y_j = \Psi\left(\sum_{i=0}^{n} w_{ji}\, x_i \right),$$

where

$$w_{j0} = \Theta_j , \quad x_0 = 1 . \tag{1.6-2}$$

The graphical implementation of the above equation is shown in the figure (1.6.a) below:



(1.6.a)

The basic artificial neuron is characterized by its nonlinearity and the threshold θ_j.

For example the early McCalloch-Pitts model of the neuron used only the binary function (cf. 1.6b) where a weighted sum of all inputs is compared with a threshold θ_j . If this sum exceeds the threshold, the neuron output is set to the high value "1", otherwise is set to the low value or logic "0".

Generally the threshold (step) function can be replaced by a more nonlinear function (cf. fig 1.6b and fig 1.6c) [1]. and consequently the output of the neuron y_i can either assume a value of a discrete set (-1 ,1) or vary continuously (between -1 and 1 or generally between y_{min} *and* y_{max}>y_{min}). The activation level or the state of the neuron is measured by the output signal y_j (if y_j = 1, then the neuron is firing (activate), if y_j = 0 then the neuron is quiescent). Various non linear …. Have been used over the year depending on the problem under consideration as shown in figures (1.6.b) and (1.6.c).

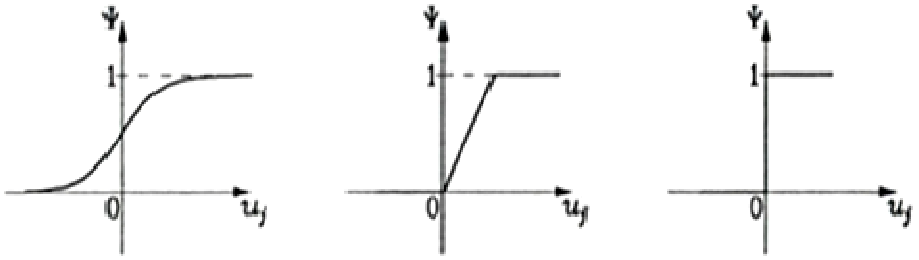


Fig. 1.6.b Three possible unipolar transfer characteristics of the amplifier: sigmoid transfer function, ramping function, hard limiter

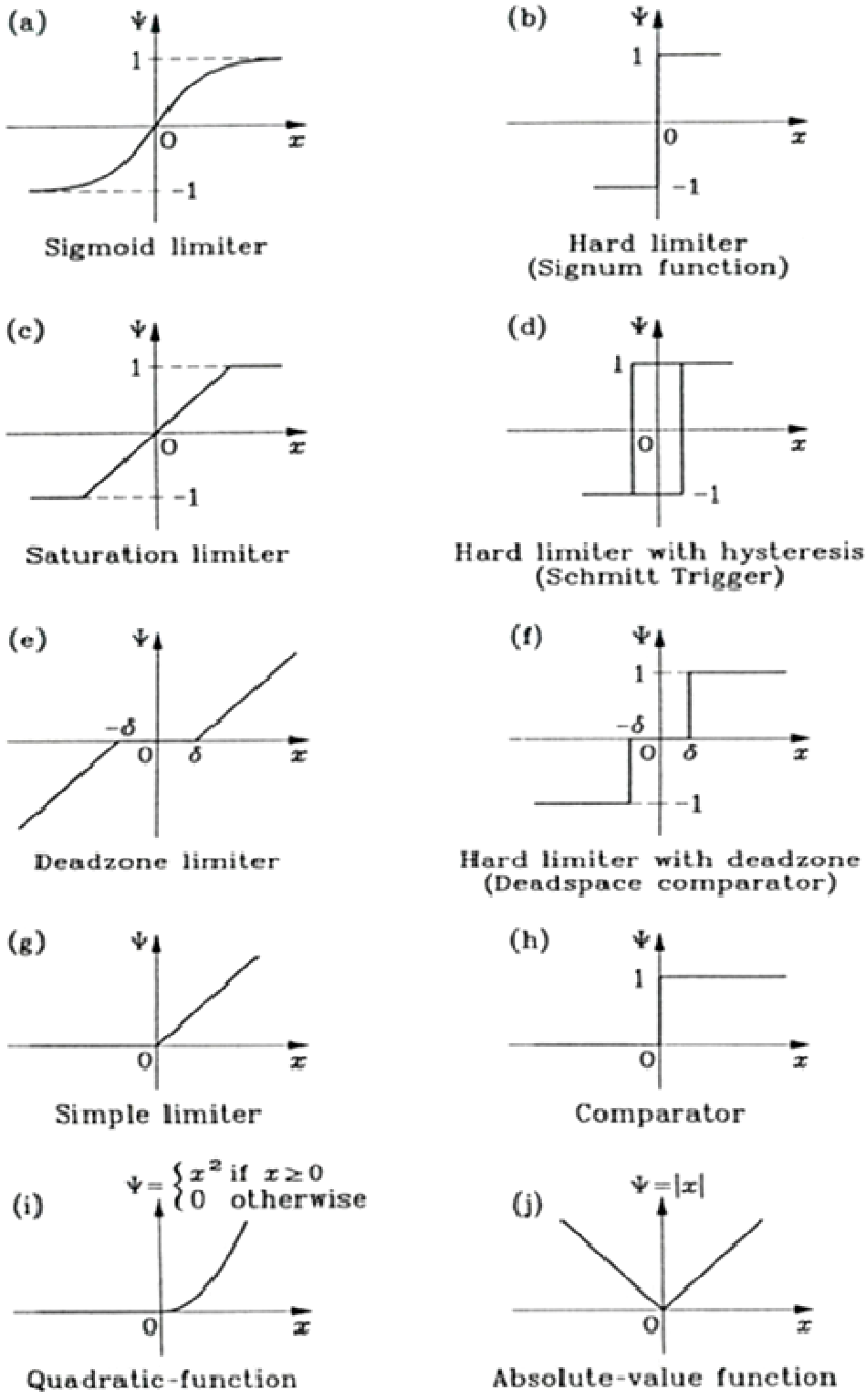


Fig 1.6.c Typical nonlinearity functions used in artificial neural networks

In the basic neural model the output signal is usually determined by a monotonically increasing sigmoid function of a weighted sum of the input signals. Such a sigmoid function can be mathematically described by

$$y_j = \tanh \gamma u_j = \frac{1 - e^{-2\gamma u_j}}{1 + e^{-2\gamma u_j}} \tag{1.6-3}$$

For a symmetrical (bipolar) representation or

$$y_j = \frac{1}{1 + e^{-\gamma u_j}} \tag{1.6-4}$$

For an unsymmetrical unipolar representation with :

$$u_j = \sum_{i=0}^{n} w_{ji}\, x_i \tag{1.6-5}$$

Where γ is a positive constant or variable which controls the ‘’steepness ‘’ (slope) of the sigmoidal function.

1.7 Widrow-Hoff Delta Rule or LMS Algorithm

Having defined the general architecture of the Neural Network that fits our problem under consideration, we then need a way to determine the weights w_{ji}. Various tools have been used, but the most useful especially when we deal with nonlinear problems, are adaptive ones. These adaptive algorithms as we will show below, lead to formulations which are amenable to gradient methods applications. Many possible adaptive algorithms, represent a mechanism-procedure by which the synaptic weights can properly be adjusted to achieve correct values. The most popular one is the *Widrow-Hoff Delta Rule*, also called the *LMS* (least mean squares) algorithm. The purpose of this algorithm is to adjust the synaptic weights w_{ji} (i=o,1,2,…,n) to assure a minimization of the error function.

$$E_j := \frac{1}{2} e_j^2(t) = \frac{1}{2} (d_j - u_j)^2 , \tag{1.7-1}$$

Which is an instantaneous estimate of the mean-square error. Applying the gradient descent approach, (mentioned in previous chapter), we obtain the system of differential equations

$$\frac{\mathrm{d}w_{ji}}{\mathrm{d}t} = -\mu \frac{\partial E_j}{\partial w_{ji}} = -\mu \frac{\partial E_j}{\partial u_j} \frac{\partial u_j}{\partial w_{ji}} , \tag{1.7-2a}$$

where $\mu > 0$. Hence, taking into account that

$$u_j = \sum_{i=0}^{n} w_{ji} x_i$$

We get

$$\frac{\mathrm{d}w_{ji}(t)}{\mathrm{d}t} = \mu e_j(t) x_i(t) = \mu \left(d_j - \sum_{p=0}^{n} w_{jp} x_p \right) x_i$$
$$(i = 0, 1, \ldots, n), \tag{1.7-2b}$$

Where $\mu > 0$ is the adaptive gain, x_i is the *i-th* input associated with the weight $w_{ji,}$ d_j is the desired output and e_j, is the output error (cf. fig 1.7.1) [1]. Equation (1.7-2b) can be implemented as a continuous-time LMS algorithm employing analog multipliers as shown below. This algorithm will be used in the sequel to study Lyapunov-based stability problem,.

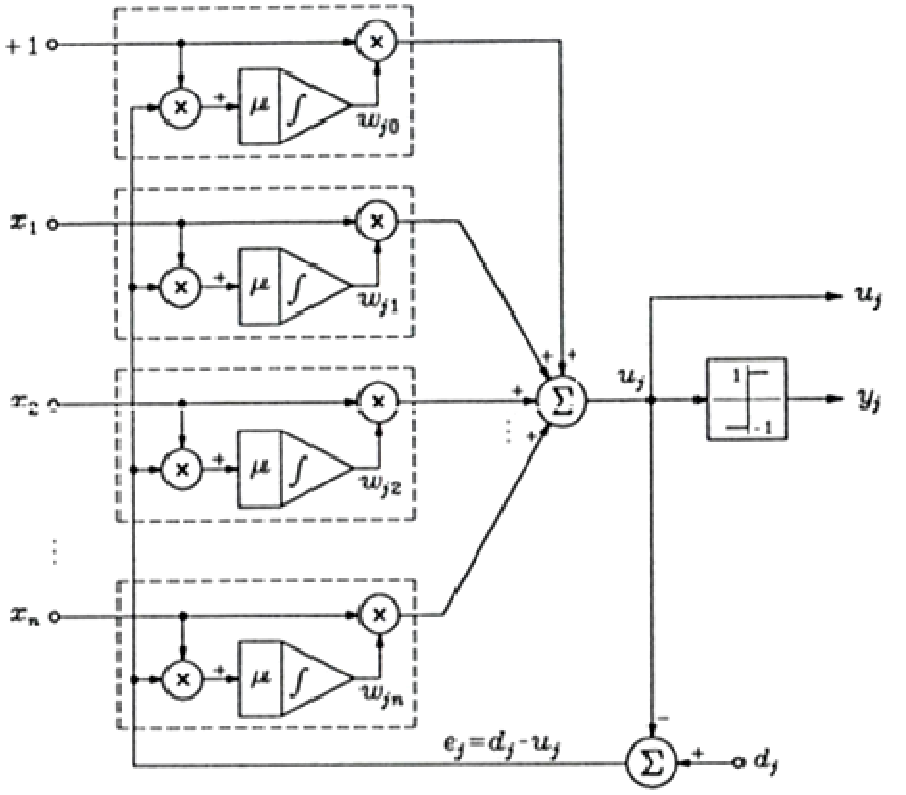


Fig. 1.7.1 Implementation of continuous-time LMS (Widrow-Hof) algorithm employing analog multipliers and integrators

Chapter 2

Lyapunov Stability Theory

Lyapunov's contribution to the study of stability systems is of great importance, especially for the stability of non linear systems. Lyaponov's methodology is analytical (not graphical), and is based on the differential equation which describes the system and provides information about the stability of the system without requiring a solution of the differential equation that describes the mathematical model of the system under study. Results of the lyapunov theory can be separated in two (2)

techniques (methods): the first method of Lyapunov and the second method of lyapunov (direct method).

2.1 Introduction-Definitions

2.1.1 Basic

Consider a dynamical system which satisfies

$$\dot{x} = f(x, t) \quad x(t_0) = x_0 \quad x \in \mathbb{R}^n \tag{2.1-1}$$

We will assume that $f(x, t)$ satisfies the standard conditions for the existence and uniqueness of solutions. Such conditions are, for instance, that $f(x, t)$ is Lipschitz continuous with respect to x, uniformly in t, and piecewise continuous in t. A point $x^* \in \mathbb{R}^n$ is an equilibrium point of (2.1-1) if $f(x^*, t) \equiv 0$

Intuitively, we say an equilibrium point is locally stable if all solutions which start near x^* (meaning that the initial conditions are in a neighborhood of x^*) remain near x^* for all time. The equilibrium point x^* is said to be locally asymptotically stable if x^* is locally stable and, furthermore, all solutions starting near x^* tend towards x^* as $t \to \infty$.

The time-varying nature of equation (2.1-1), however, introduces all kinds of additional subtleties. Nonetheless, it is intuitive that a pendulum has a locally stable equilibrium point when the pendulum is hanging straight down and an unstable equilibrium point when it is pointing straight up. If the pendulum is damped, the stable equilibrium point is locally asymptotically stable.

By shifting the origin of the system, we may assume that the equilibrium point of interest occurs at $x^* = 0$. If multiple equilibrium points exist, we will need to study the stability of each by appropriately shifting the origin.

2.1.2 Stability in the sense of Lyapunov

The equilibrium point $x^* = 0$ of *(2.1-1)* is stable (in the sense of Lyapunov) at $t = t_0$ if for any $\varepsilon > 0$ there exists a $\delta(\varepsilon, t_0) > 0$ such that

$$\|x(t_0)\| < \delta \quad \Longrightarrow \quad \|x(t)\| < \epsilon, \quad \forall t \geq t_0. \tag{2.1-2}$$

Lyapunov stability is a very mild requirement on equilibrium points. In particular, it does not require that trajectories starting close to the origin tend to the origin asymptotically.

Also, stability is defined at time instant t_0. Uniform stability is a concept which guarantees that the equilibrium point is not losing stability. We insist that for a uniformly stable equilibrium point x^*, δ not be a function of t_0, so that equation (2.1-2) may hold for all t_0.

2.1.3 Asymptotic stability

An equilibrium point $x^* = 0$ of (2.1-1) is asymptotically stable at $t = t_0$ if

1. $x^* = 0$ is stable, and
2. $x^* = 0$ is locally attractive; i.e., there exists $\delta(t_0)$ such that

$$\|x(t_0)\| < \delta \quad \Longrightarrow \quad \lim_{t \to \infty} x(t) = 0. \tag{2.1-3}$$

As in the previous definition, asymptotic stability is defined at t_0.

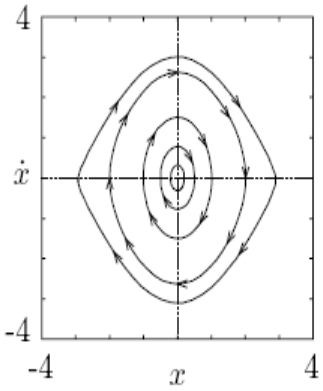
Uniform asymptotic stability requires:

1. $x^* = 0$ is uniformly stable, and

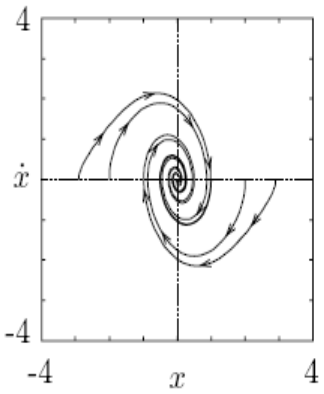
2. $x^* = 0$ is uniformly locally attractive; i.e., there exists δ independent of t_0 for which equation (2.1-3) holds.

Further, it is required that the convergence as defined by equation (2.1-3) is uniform. .

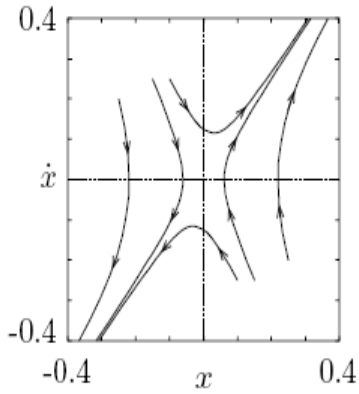
Finally, we say that an equilibrium point is unstable if it is not stable. This is less of a tautology than it sounds and we should be sure if we can negate the definition of stability in the sense of Lyapunov to get a definition of instability. [22] Figure (2.1) illustrates the difference between stability in the sense of Lyapunov and asymptotic stability.



(a) Stable in the sense of Lyapunov



(b) Asymptotically stable



(c) Unstable (saddle)

Figure 2.1: Phase portraits for stable and unstable equilibrium points.

Definitions 2.1.2 and 2.1.3 are local definitions. They describe the behavior of a system near an equilibrium point. We say an equilibrium point x^* is globally stable if it is stable for all initial conditions $x_0 \in \mathrm{R}^n$. Global stability is very desirable, but in many applications it can be difficult to achieve. We will concentrate on local stability theorems and indicate where it is possible to extend the results to the global case.

Notions of uniformity are only important for time-varying systems. Thus, for time-invarying systems, stability implies uniform stability and asymptotic stability implies uniform asymptotic stability. It is important to note that the definitions of asymptotic stability do not quantify the rate of convergence.

2.2 Lyapunov's First Method

The first method of Lyapunov [2] is based on the linearization and replacement of the non linear differential equation, which describes the system, with a linear differential equation. This approximation takes place for every equilibrium condition separately and the stability results of the non linear system respond only for a small area around of the equilibrium condition.

Consider a linear system:

$$x^{\cdot} = f(x) \tag{2.2-1}$$

and x_e an equilibrium condition. We analyze the above equation (2.2-1) to a Taylor series in the point $x=x_e$, so we take the follow:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_e) + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]^T_{\mathbf{x}=\mathbf{x}_e}(\mathbf{x}-\mathbf{x}_e) + \frac{1}{2}(\mathbf{x}-\mathbf{x}_e)^T\left[\frac{\partial}{\partial \mathbf{x}}\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]^T_{\mathbf{x}=\mathbf{x}_e}(\mathbf{x}-\mathbf{x}_e)+\ldots$$

$$= \mathbf{f}(\mathbf{x}_e) + \mathbf{A}(\mathbf{x}-\mathbf{x}_e) + \mathbf{B}(\mathbf{x}-\mathbf{x}_e)](\mathbf{x}-\mathbf{x}_e)+\ldots$$

where

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix}, \quad \mathbf{A} = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right]^T_{\mathbf{x}=\mathbf{x}_e} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}_{\mathbf{x}=\mathbf{x}_e}$$

The matrix $B(x\text{-}x_e)$ contains terms which come from bigger rank derivatives than the first. Because of the fact that the x_e is an equilibrium point, it means that :

$f(x_e) = 0$

If we now set $z = x\text{-}x_e$ then the above equation can be written as

$$z^{\cdot} = Az + B(z)z + \ldots \tag{2.2-3}$$

The first approach, which means the linear part of (2.2-3) will be

$$\dot{z} = Az \tag{2.2-4}$$

Generally the lyapunov's first approximation is based on the follow theorem:

Theorem 2: If every factor of matrix A has a not zero(0) real part, then the conclusions concerning the stability of the non linear system in the area of x_e can be exported by the study of the linear system stability (2.2-4).

2.3 Lyapunov's Second Method (Direct Method)

Lyapunov's second method which is also called as Lyapunov's direct method allows us to determine the stability of a system without explicitly integrating the differential equation (2.2-1). The method is a generalization of the idea that if there is some "*measure of energy*" in a system, then we can study the rate of change of the energy of the system to ascertain stability. To make this precise, we need to define exactly what one means by a "measure of energy." Let B_ε be a ball of size (ε) around the origin,

$$B_\epsilon = \{x \in \mathbb{R}^n : \|x\| < \epsilon\}.$$

2.3.1 Basics of Direct Method

A Lyapunov function is not necessarily the energy of a dynamical system. However, the term Lyapunov function is often used instead of the energy function, cost function or objective function and vice versa in the neural networks community. Strictly speaking, the Lyapunov function denoted by $V(x)$ is positive definite. The energy function is always bounded from below, so the relation holds $V(x) = E(x) - E(x^*)$.

Let us consider a dynamical system, described by a set of coupled differential equations

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\theta}) \tag{2.3-1}$$

$$\frac{d\boldsymbol{W}}{dt} = \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\theta}),$$

(2.3-2)

Where $x = [x_1(t), x_2(t),\ldots, x_n(t)]^T$ is the activation state space vector, $W(x) = [w_{ij}(t)]_{n \times n}$ is the weight matrix and $\Theta = [\theta_1, \theta_{2,\ldots,} \theta_n]^T$ is the external time independent pattern vector. Such systems of differential equations describe a general model of the neural network. The first of the above equations is called the generalized neural network state equation and the second differential equation is the general learning equation.

The energy function $E = E(x, W, \Theta)$ is a function defined on the state space which is non increasing along the trajectories and it is bounded from below. The function E will be called a Lyapunov function if it is decreasing on all non constant trajectories and thus the Lyapunov function forces every trajectory to a stable equilibrium point.

The Lyapunov theorem can be stated as follows:

Theorem 3*: If for a given system of differential equations an energy function E exists, called a Lyapunov function satisfying the following relation*:

$$\frac{dE}{dt} = \sum_{j=1}^{n} \frac{\partial E}{\partial x_j} \frac{dx_j}{dt} + \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial E}{\partial w_{ij}} \frac{dw_{ij}}{dt} \leqq 0, \tag{2.3-3}$$

with

$$\frac{dE}{dt} = 0 \quad \text{only for} \quad \frac{dx}{dt} = \mathbf{0} \quad \text{and} \quad \frac{dW}{dt} = \mathbf{0},$$

Then the system is stable, the trajectories $x_j(t)$ *and* $w_{ij}(t)$ *converge to stationary points as* $t \to \infty$.

In such a case, it is of paramount importance to refer to the fact that since E is bounded monotonically decreasing function of time, it converges to a limit and its time derivative converges to (*0*). The time derivative *dE/dt* is strictly less than zero except at the equilibrium points where it vanishes.

Obviously the Lyapunov theorem(3) doesn't say anything about the form of the function E or how to construct it. It gives only a sufficient condition for the convergence for $t \to \infty$ and is not concerned with a finite time of convergence.

Unfortunately there is no general method for constructing Lyapunov functions. In the following we shall define some characteristics of the Lyapunov funtions and then show how we can construct them for linear systems.

2.3.2 Locally Positive Definite Functions

A continuous function $V : \mathbb{R}^n \times \mathbb{R}_+ \rightarrow \mathbb{R}$ is a locally positive definite function if for some $\varepsilon > 0$ and some continuous, strictly increasing function $\alpha : \mathbb{R}_+ \rightarrow \mathbb{R}$,

$$V(0,t) = 0 \quad \text{and} \quad V(x,t) \geq \alpha(\|x\|) \qquad \forall x \in B_\epsilon, \forall t \geq 0. \tag{2.3-4}$$

A locally positive definite function is locally like an energy function. Functions which behave globally as energy functions are called positive definite functions:

2.3.3 Positive Definite Functions

A continuous function $V : \mathbb{R}^n \times \mathbb{R}_+ \rightarrow \mathbb{R}$ is a positive definite function if it satisfies the conditions of Definition 2.3.1 and, additionally, $\alpha(p) \rightarrow \infty$ as $p \rightarrow \infty$.

2.3.4 Decreasing Functions

A continuous function $V : \mathbb{R}^n \times \mathbb{R}_+ \rightarrow \mathbb{R}$ is a decreasing function if for some $\varepsilon > 0$ and some continuous, strictly increasing function $\beta : \mathbb{R}_+ \rightarrow \mathbb{R}$,

$$V(x,t) \leq \beta(\|x\|) \qquad \forall x \in B_\epsilon, \forall t \geq 0 \tag{2.3-5}$$

Using these definitions, the following theorem allows us to determine stability for a system by studying an appropriate energy function.

(Theorem) :Roughly, this theorem states that when V (x, t) is a locally positive definite function and $\dot{V}(x, t) \leq 0$ *then we can conclude stability of the equilibrium point. The time derivative of V is taken along the trajectories of the system:*

$$\dot{V}\Big|_{\dot{x}=f(x,t)} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f.$$

Table 2.2 : Summary of the basic theorem of Lyapunov.

	Conditions on $V(x,t)$	Conditions on $-\dot{V}(x,t)$	Conclusions
1	lpdf	≥ 0 locally	Stable
2	lpdf, decrescent	≥ 0 locally	Uniformly stable
3	lpdf, decrescent	lpdf	Uniformly asymptotically stable
4	pdf, decrescent	pdf	Globally uniformly asymptotically stable

Where **lpdf** means locally positive definite function.

2.3.5 Basic Theorem Of Lyapunov

Theorem 4*: Let V(x,t) be a non negative function with derivative* $\dot{V}$ *along the trajectories of a system:*

1. *If V(x,t) is locally positive definite and* $\dot{V}(x, t) \leq 0$ *locally in x and for all t, then the origin of the system is locally stable*

2. *If V(x,t) is locally positive definite and decreasing, and* $\dot{V}(x, t) \leq 0$ *locally in x and for all t, then the origin of the system is uniformly locally stable.*

3. *If V(x,t) is locally positive definite and decreasing, and* $-\dot{V}(x, t) \leq 0$ *is locally positive definite , then the origin of the system is uniformly locally asymptotically stable.*

4. *If V(x,t) is positive definite and decreasing, and $-\dot{V}(x, t) \leq 0$ is positive definite , then the origin of the system is globally uniformly asymptotically stable.*

The conditions in the theorem are summarized in Table 2.2

Theorem 2.3.4 gives sufficient conditions for the stability of the origin of a system. However, the former doesn't give a prescription for determining the Lyapunov function $V(x, t)$. Since the theorem only gives sufficient conditions, the search for a Lyapunov function establishing stability of an equilibrium point could be difficult. However, it is a remarkable fact that the converse of Theorem 2.3.4 also exists. If an equilibrium point is stable, then there is a function $V(x, t)$ satisfying the conditions of the theorem. However, the utility of this and other converse theorems is limited by the lack of a computable technique for generating Lyapunov functions.

2.4 Finding Lyapunov Functions

From the text above comes to surface the fact that the problem of studying the stability of a system through the second method of Lyapunov consists of finding a Lyapunov's function for the specific system under consideration. That function may not be the only one and simultaneously can be difficult to be found. So far we haven't invented a systematical way of finding a Lyapunov function for all the specific categories of systems. *It must be noticed that when there is no possibility of finding a Lyapunov function for a system, it means that we just can't conclude about the stability of a system. It doesn't mean that the system is unstable.*

There are many different types of Lyapunov theorems. The key in all cases is to find a Lyapunov function and verify that it has the required properties.

2.4.1 Theorem

For the case of the time invarying linear systems we have the following theorem:

Theorem 5: Consider the linear time invarying system:

$$\dot{x}=Ax \ , \ |A| \neq 0 \text{ and } x_e=0$$

Consider the Lyapunov function $V(x) = x^TPx$, where P is a positive definite real symmetric matrix.

Then the function $V(x) = x^TPx$ is the Lyapunov function of the system if only for a matrix P there is a positive definite real symmetric matrix Q satisfying the following

$$A^TP + PA = - Q \qquad (2.4\text{-}1)$$

Moreover for the time varying systems which are described by the differential equation

$$\dot{x}(t)=A(t)x$$

the equation that we take, is the following:

$$\dot{P}(t) + A^T(t)P(t) + P(t)A(t) = - Q(t) \qquad (2.4\text{-}2)$$

On the other hand for the non linear systems have already suggested some methods for finding the Lyapunov function. One of the dominants methods is either the Schultz-Gibson or the gradient method. The latter is going to be analyzed thoroughly in the following.

Consider a function $V(x)$ which is a possible Lyapunov function[2]. Then :

$$\dot{V}(\mathbf{x}) = (\nabla V)^T \dot{\mathbf{x}}$$

where

$$\nabla = \text{grad}_{\mathbf{x}}$$

and

$$\nabla V = \begin{bmatrix} \frac{\partial V}{\partial x_1} \\ \vdots \\ \frac{\partial V}{\partial x_n} \end{bmatrix} \tag{2.4-3}$$

If the function ∇V is known, then the function V is obtained by the integral:

$$V = \int_0^{\mathbf{x}} (\nabla V)^T d\mathbf{x} = \int_0^{x_1,(x_2=\ldots x_n=0)} \left[\frac{\partial V}{\partial x_1}\right] dx_1 + \int_0^{x_2,(x_1=x_1,x_3=\ldots x_n=0)} \left[\frac{\partial V}{\partial x_2}\right] dx_2$$

$$+\ldots+ \int_0^{x_n,(x_i=x_i,\, i=1,2,\ldots,n-1)} \left[\frac{\partial V}{\partial x_n}\right] dx_n \tag{2.4-4}$$

For the scalar function V to be unique, which means independent from the way it is obtained through integration, the equation (2.4-5) has to be satisfied[2]:

$$\nabla \times \nabla V = \mathbf{0} \tag{2.4-5}$$

Where ∇x means the operation '*curl*' , and for the three (3) dimension example for cartesian coordinates x_1, x_2, x_3 becomes :

$$\nabla \times \mathbf{A} = \left[\frac{\partial A_3}{\partial x_2} - \frac{\partial A_2}{\partial x_3}\right] i_1 + \left[\frac{\partial A_1}{\partial x_3} - \frac{\partial A_3}{\partial x_1}\right] i_2 + \left[\frac{\partial A_2}{\partial x_1} - \frac{\partial A_1}{\partial x_2}\right] i_3$$

Where $A = A_1\, i_1 + A_2\, i_2 + A_3\, i_3$, and i_1, i_2, i_3, are unary vectors to the direction of axis $x_1, x_2,$ *and* x_3.

Consider $A = \nabla V$. Then

$$\nabla \times \nabla V = \left[\frac{\partial^2 V}{\partial x_3 \partial x_2} - \frac{\partial^2 V}{\partial x_2 \partial x_3}\right] i_1 + \left[\frac{\partial^2 V}{\partial x_1 \partial x_3} - \frac{\partial^2 V}{\partial x_3 \partial x_1}\right] i_2 + \left[\frac{\partial^2 V}{\partial x_2 \partial x_1} - \frac{\partial^2 V}{\partial x_1 \partial x_2}\right] i_3$$

For the equation (2.4-5) to be satisfied we require the following conditions to exist .

$$\frac{\partial^2 V}{\partial x_3 \partial x_2} = \frac{\partial^2 V}{\partial x_2 \partial x_3}, \quad \frac{\partial^2 V}{\partial x_1 \partial x_3} = \frac{\partial^2 V}{\partial x_3 \partial x_1}, \quad \frac{\partial^2 V}{\partial x_2 \partial x_1} = \frac{\partial^2 V}{\partial x_1 \partial x_2} \tag{2.4-6}$$

The general form of the above relations must be put in the form of a matrix B as follows:

$$B = \begin{bmatrix} \frac{\partial^2 V}{\partial x_1^2} & \frac{\partial^2 V}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 V}{\partial x_1 \partial x_n} \\ \frac{\partial^2 V}{\partial x_2 \partial x_1} & \frac{\partial^2 V}{\partial x_2^2} & \cdots & \frac{\partial^2 V}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 V}{\partial x_n \partial x_1} & \frac{\partial^2 V}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 V}{\partial x_n^2} \end{bmatrix} \quad (2.4\text{-}7)$$

So far, we have converted the problem of finding a V Lyapunov function to a problem of finding a V , which is $\bar{\nabla} * \bar{\nabla} V = 0$ and of course V satisfies the $\|x_0 - x_e\| \leq \delta$.

The main steps of determing counting a Lyapunov function using the gradient method are:

1. we consider $\bar{\nabla} V = G(x,t)x$ where G is a steady matrix (2.4-8)
2. we calculate the $\dot{V}$, and we notice to either be positive definite or to be at least negative semi-definite
3. we bound the $\dot{V}$ to meet constraints (2.4-5)- (2.4-7)
4. we check if after applying step 3, V still satisfies the step 2
5. we derive the V according to relation (2.4-4)
6. we check if the V satisfies the $\|x_0 - x_e\| \leq \delta$

Here, we can highlight the fact that if we are not able to find the Lyapunov function with the method mentioned above, then it doesn't mean that the system is unstable. In this study we shall concentrate on how Neural Networks can be implemented to study Lyapunov based stability linear time invarying problems.

Chapter 3

Neural Networks for Matrix Algebra Problems

3.1 introduction

Researchers in modern scientific and engineering computing are actively searching for fast, efficient and robust algorithms which are parallel in nature. Neuron- like networks enable us to construct massively parallel algorithms in the sense that a system of differential equations describing specific algorithms is solved simultaneously by an associated circuit consisting of highly interconnected processing units.

The main purpose of statements mentioned above is to describe a class of such massively parallel algorithms for solving in real time a large variety of important matrix algebra problems[1] such as matrix inversion, *LU* decomposition, *QR* factorization, the eigenvalue problem, singular value decomposition (*SVD*), solving the Lyapunov equation, and principal component analysis (*PCA*). The algorithms are associated with neuron-like adaptive systems and are based mostly on error back-propagation techniques. In the following, we will be occupied with solving of Lyapunov based stability problem. As we mentioned before there exist many Lyapunov based stability problems, the analysis of which results in the solution of a matrix equation as shown in (2.4-1). In the following we shall show how we can solve these problems using Neural Networks

3.2 Neural Networks for Solving Generalized Matrix Equations

Especially Lyapunov's Equation

In some applications, especially in mathematical control theory, that involves Lyapunov based stability problems (2.4-1), it is required to find an unknown matrix X which is involved in linear matrix equations containing several matrices. Such equations are called generalized matrix equations:

$$A X + X B + C = 0 \qquad (3.2\text{-}1)$$

Where the matrices A, B, C are known and the matrix X is unknown (see equation 2.4-1).

In that point we will be occupying with the Lyapunov equation

$$A X + X A^T = - C$$

The problem is formulated as follows:

For given matrices $A \in R^{n \times n}$, and $C \in R^{n \times n}$ we will find a symmetric matrix X which satisfies the equation (3.2-1).

To solve the problem the first step will be to implement the latter equation by an appropriate network shown in figure (3.2.1) [1]. For such a network we can formulate the Error (cost) function as:

$$E = \frac{1}{2} \| e \|_2^2 = \frac{1}{2} \sum_{i=1}^{n} e_i^2 , \qquad (3.2\text{-}2)$$

Where

$$e = [e_1, e_2, \ldots, e_n]^T ,$$

$$y_i = r_i + p_i = \sum_{k=1}^{n} a_{ik} v_k + \sum_{k=1}^{n} x_{ik} u_k .$$

$$e_i = d_i - y_i = - \sum_{j=1}^{n} c_{ij} z_j - y_i ,$$

The equation $A X + X A^T = - C$ is shown in a block diagram form in figure (3.2.1) below.

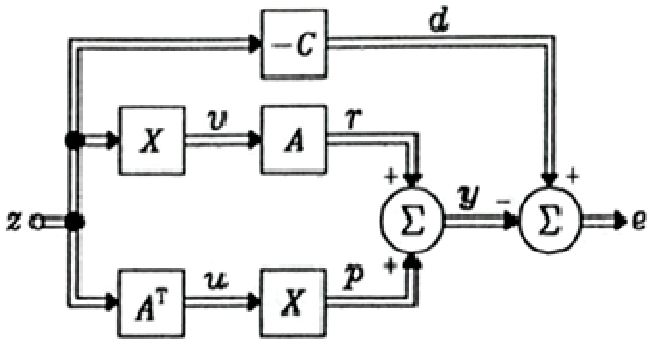


Fig. 3.2.1 Network architecture for solving the Lyapunov equation

A direct minimization of the error function $E(t)$ in equation (3.2-2) leads to the learning algorithm:

$$\frac{dx_{ij}}{dt} = \mu \left[e_i u_j + \left(\sum_{k=1}^{n} a_{ki} e_k \right) z_j \right]$$

$$(i, j = 1, 2, \ldots, n),$$

(3.2-3)

Where $\mu > 0$.

Unfortunately such an algorithm will not satisfy the symmetry condition of matrix X ($x_{ij} = x_{ji}$).

In order to meet the symmetry constraints the above algorithm (3.2-3) can be modified as:

$$\frac{dx_{ij}}{dt} = \frac{\mu}{2}\left[e_i u_j + e_j u_i + \left(\sum_{k=1}^{n} a_{ki} e_k\right) z_j + \left(\sum_{k=1}^{n} a_{kj} e_k\right) z_i\right]$$

$$(i, j = 1, 2, \ldots, n),$$

(3.2-4)

An analog implementation of the above algorithm is shown in figure (3.2.2) [1].

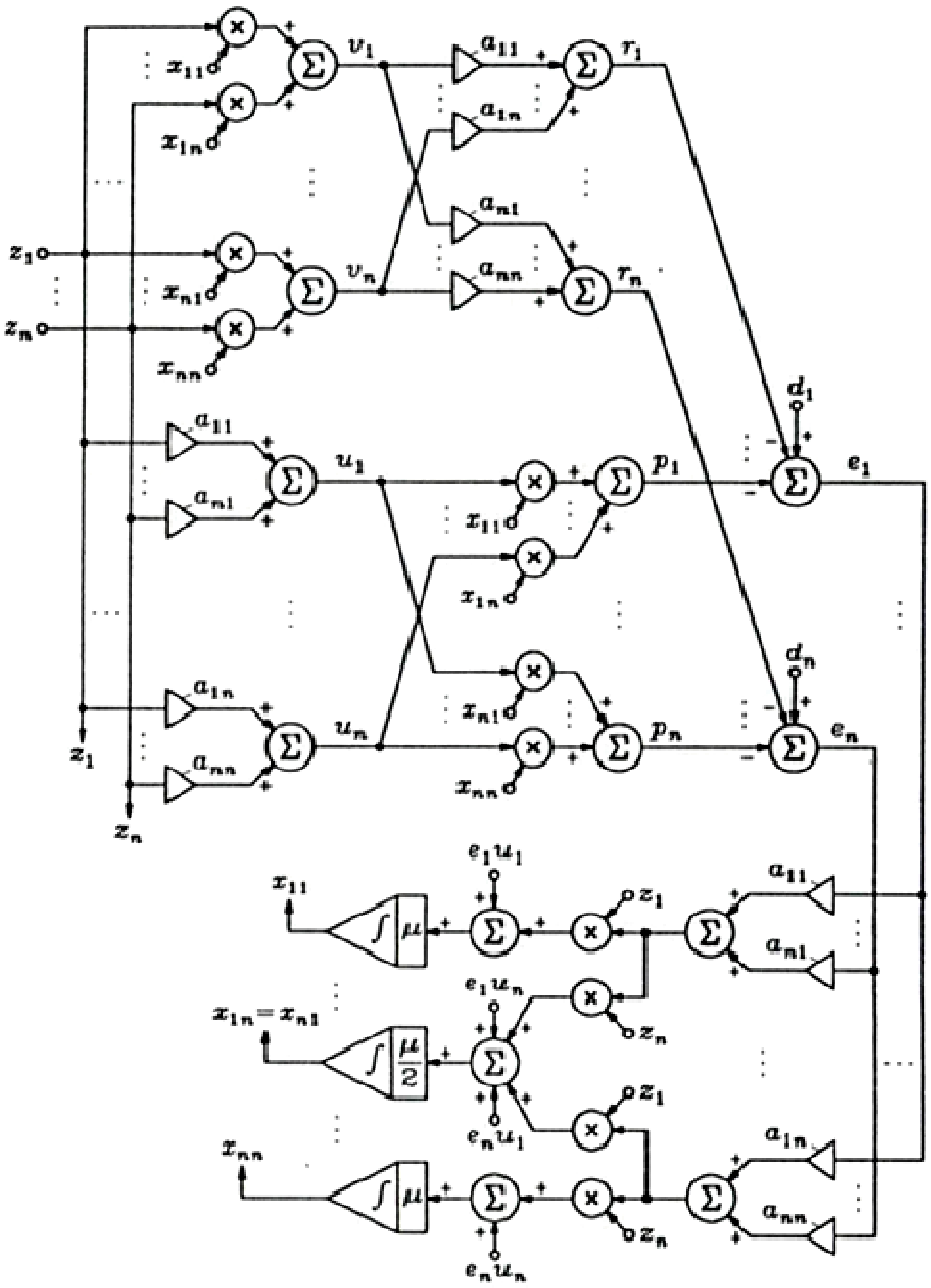


Fig. 3.2.2 Analog implementation of the learning algorithm

3.3 The Problem of Solving the Lyapunov Equation

Let us consider the problem of solving the Lyapunov equation [1].

$AX + XA^T = -C$, where the matrices A, C are known

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 0 & 8 \end{bmatrix} \qquad C = \begin{bmatrix} 7 & 2 & 4 \\ 2 & 5 & 1 \\ 4 & 1 & 6 \end{bmatrix}$$

Using the Neural Network of figure (3.2.2), in other words, we seek to find the symmetric matrix X knowing matrices A,C.

The network depicted in figure (3.2.2) has been simulated in Matlab in order to solve the problem. Actually the point is to approximate as much as possible the real values of X, something which will not be that easy because of the error possibility and the complexity of the algorithm presented above.

$$\frac{dx_{ij}}{dt} = \frac{\mu}{2}\left[e_i u_j + e_j u_i + \left(\sum_{k=1}^{n} a_{ki} e_k\right) z_j + \left(\sum_{k=1}^{n} a_{kj} e_k\right) z_i\right]$$

$$(i, j = 1, 2, \ldots, n),$$

$$E = \frac{1}{2} \| e \|_2^2 = \frac{1}{2} \sum_{i=1}^{n} e_i^2 ,$$

Through function eig() in Matlab, we were able to find the eigenvalues of matrix A in order to see if the system was asymptotically stable. The results are shown below:

λ = -2.0113

11.4534

4.5580

Hence with a quick look the system is not asymptotically stable, since the eigenvalues of matrix A are not in the closed left half plane. In order to succeed asymptotical stability the eigenvalues of the matrix A should be negative (in the closed left half plane).

Now we will apply the Lyapunov method taking into consideration our Matlab code. We chose a positive definite matrix C (all the eigenvalues of C are positive, λ= 2.3134, 4.4072, 11.2794), in order to apply our method.

The simulation results using Matlab for the matrices A and C given above yield the values of the elements x_{11}, $x_{12}=x_{21}$, x_{22} , $x_{23}=x_{32}$, x_{33}, $x_{13}=x_{31}$ of the matrix X. Our code was able to find the graphical representations of matrix *X (3*3)* and the *E(t)* which is the Error (cost) function, as they can be seen below in figures (3.3.1)-(3.3.2)

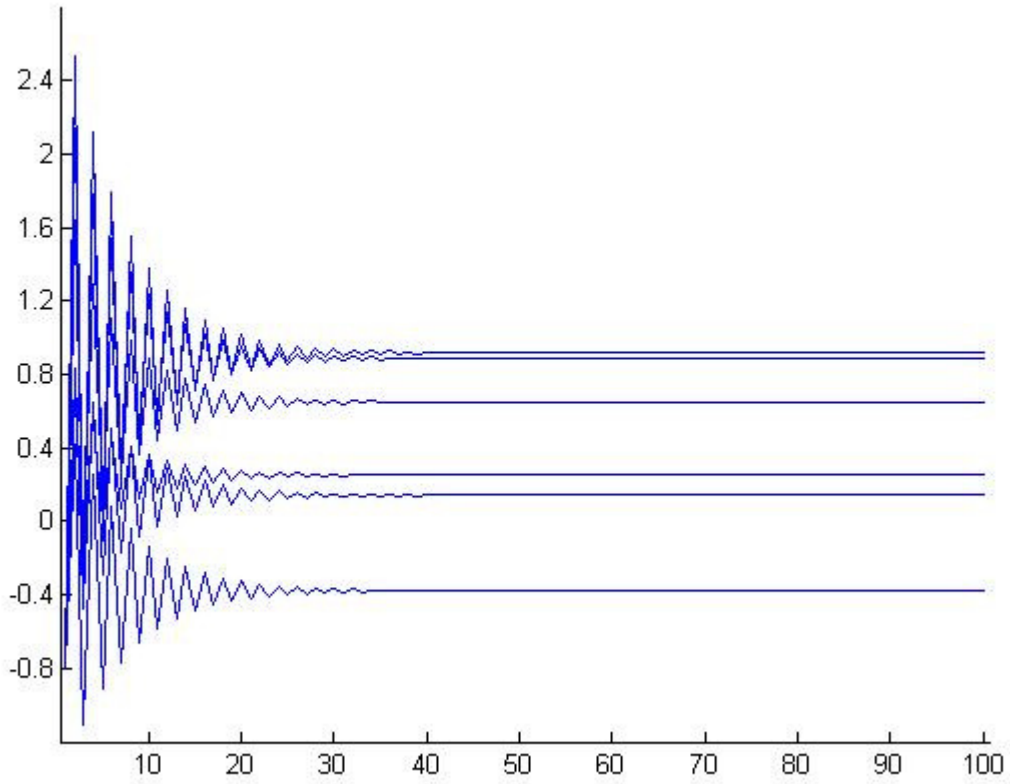


Fig .3.3.1: graphical representation of matrix X (3*3).

Where the X-axis presents the number of iterations and the Y-axis presents the matrix X-values. The first line from above is the value x_{33} , the second line is the x_{11} , the third line is the $x_{12} = x_{21}$,the fourth line is the x_{22} , the fifth line is the $x_{23} = x_{32}$, and finally the sixth line is the $x_{13} = x_{31}$.

In order for the system to be stable, all the eigenvalues of matrix X (values of X are shown in the above graphical representation) must be positive according to *theorem* 5. Matlab, through its function eig(), gave us the following results

$\lambda =$ -0,18157

-0,042606

0,01319

As we can see the *theorem* 5 is not confirmed, because all the eigenvalues of matrix X are not in the closed right half plane (which means that the eigenvalues are not all positives). Hence X is not positive definite and the Lyapunov test indicates that the system under consideration is not stable.

Concerning the Error (cost) function graphical representation, it takes the following form:

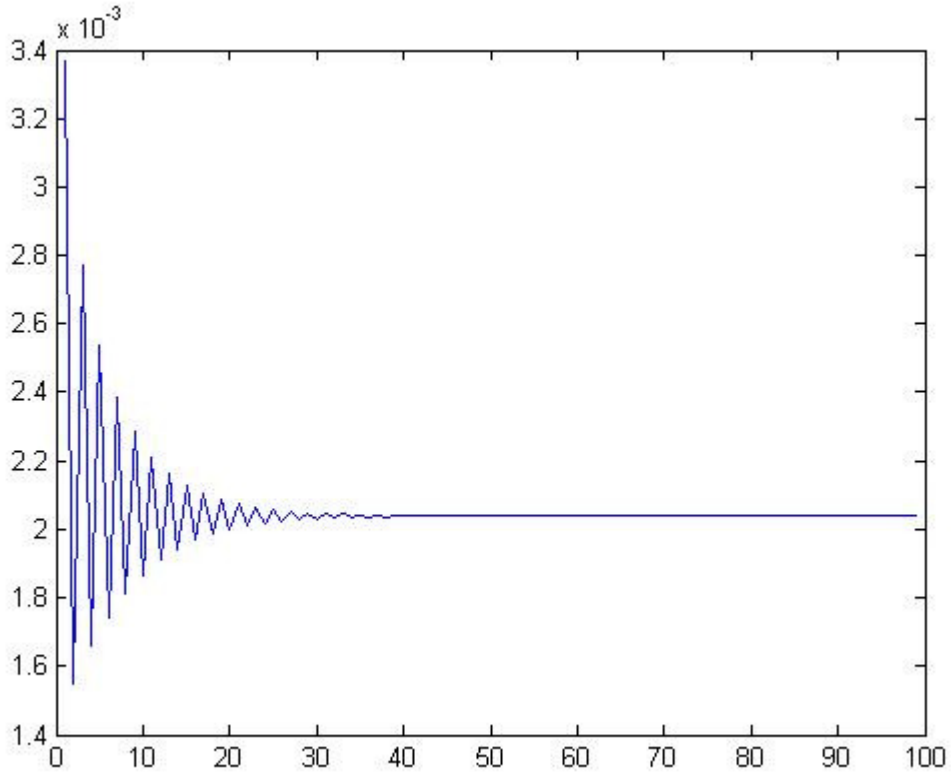


Fig. 3.3.2: graphical representation of error (cost) function.

Where the X-axis presents the number of iterations and the Y-axis presents the error.

What we see above is that the Error function E(t) decreases while *t* increasing until the stored energy reaches a value. The above Error function is very small and converges after some iterations (between 40-50 iteration we succeed the convergence).

Now we will present another example for Lyapunov stability. The example is a linearized model that describes the dynamics of an aircraft that possesses vertical take off and landing. The equation of the model is:

$$x' = Ax + B_1u_1 + B_2u_2 \qquad (3.3\text{-}1)$$

Where:

$$A = \begin{bmatrix} -0.0366 & 0.0271 & 0.0188 & -0.4555 \\ 0.0482 & -1.0100 & 0.0024 & -4.0208 \\ 0.1002 & 0.3681 & -0.7070 & 1.4200 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

And

$$B_1 = \begin{bmatrix} 0,4422 \\ 3.5446 \\ -5.5200 \\ 0 \end{bmatrix}$$

$$B_2 = \begin{bmatrix} 0.1761 \\ -7.5922 \\ 4.4900 \\ 0 \end{bmatrix}$$

The elements of the state vector are: i) x_1 which corresponds to the horizontal component of the velocity, ii) x_2 which corresponds to the vertical component of the velocity, iii) x_3 which corresponds to rate of change of the inclination angle (degrees/seconds), iv) x_4 which corresponds to the inclination angle (degrees/seconds). The input u_1 is used mainly for the motion control of vertical axis, while u_2 is used for the control of the motion of the horizontal axis.

The matrix C is positive definite, where C=I. So we have :

$$A = \begin{bmatrix} -0.0366 & 0.0271 & 0.0188 & -0.4555 \\ 0.0482 & -1.0100 & 0.0024 & -4.0208 \\ 0.1002 & 0.3681 & -0.7070 & 1.4200 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = I$$

Through function eig() in Matlab, we were able to find the eigenvalues of matrix A in order to see if the system was asymptotically stable. The results are shown below:

$\lambda =$ 0.2758 + 0.2576i

0.2758 - 0.2576i

-0.2325

-2.0727

Hence with a quick look the system is not asymptotically stable, since the eigenvalues of matrix A are not in the closed left half plane (the eigenvalues are in the right half plane). In order to achieve asymptotical stability the eigenvalues of the matrix A should be negative (in the closed left half plane). However, the system under consideration is controllable from either u_1 or u_2. If the aircraft loses somehow the control of the vertical motion through u_1, then the control u_2 can be used to control both vertical and horizontal motion and vice versa.

Now we will apply the Lyapunov method $(AX + XA^T = - C)$ taking into consideration our Matlab code, as we did in the previous example. We chose a positive definite matrix C=I as it can be seen above, in order to apply our method.

The simulation results using Matlab for the matrices A and C given above, yield the values of the elements $x_{11}, x_{22}, x_{33}, x_{44}, x_{12}=x_{21}, x_{13}=x_{31}, x_{14}= x_{41}, x_{24}= x_{42}, x_{23}= x_{32}, x_{43}= x_{34}$ of the matrix X. Our code was able to find the graphical representations of matrix *X (4*4)* and the *E(t)* which is the Error (cost) function, as they can be seen below in figures (3.3.3)-(3.3.4).

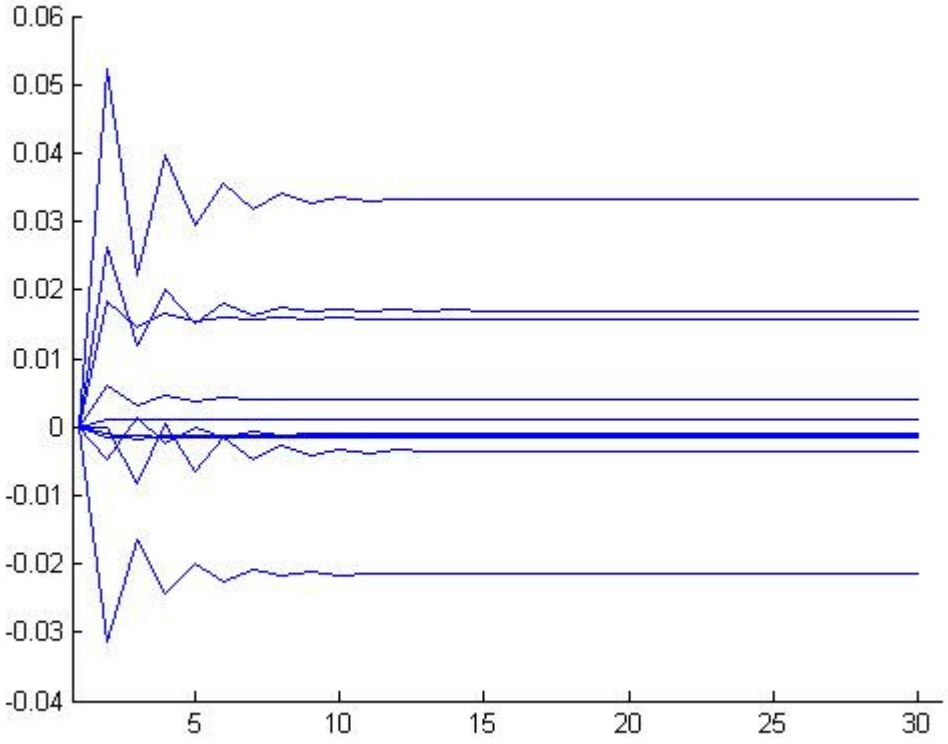


Fig .3.3.3: graphical representation of matrix X (4*4).

Where the X-axis presents the number of iterations and the Y-axis presents the matrix X-values.

In order for the system to be stable, all the eigenvalues of matrix X must be positive according to *theorem* 5. Matlab, through its function eig(), gave us the following results

$\lambda =$ -0,034316

0.00093076

0.015528

0.048074

As we can see the *theorem* 5 is not confirmed, because all the eigenvalues of matrix X are not in the closed right half plane (which means that they are not all positives). Hence X is not positive definite and the Lyapunov test indicates that the system under consideration is not stable. But as we mentioned above the system under consideration is controllable from either u_1 or u_2, something which is very important for the safety of the airplane and its passengers. Hence if the aircraft loses, for any

reason, its control of the vertical motion through u_1, then the control u_2 will be used to control both vertical and horizontal motion and vice versa.

In this point we will proceed with an analysis of the problem above, concerning its stability. The stability theorem in [23](which can be seen in Appendix) is so powerful, where someone would like to seek, depending on the same theorem, what it could happen, if the system was unstable in a specific sense as we explained below.

The above question is already been examined and proved by [23]. Hence if the solution of the equation $A^TP + PA = -C$ gives a matrix P to be positive definite as well as eigenvalues of matrix A are discrete and in the left half plane except for one, then the system is proved to be unstable. From that theorem we can conclude, however it is not the target of this dissertation, that the instability of the system is further confirmed when the matrix P is not positive definite, like in our case.

Although, even in that case where we study practical applications of the above theorem, like in this study, where we deal with an example for Lyapunov stability which contains a linearized model that describes the dynamics of an aircraft that possesses vertical take off and landing, the methodology that we have already developed can help us determine if and in what level our system is controllable. This controllability can be used for the safe operation of a system

In that case if the system is controllable, we guarantee its safe operation even if the mechanical, the operational and the economical design enforced us to select parameters which made the matrix A unstable. The alteration of the parameters could be the result of system deterioration due to time or other causes.

Concerning the Error (cost) function graphical representation, it takes the following form:

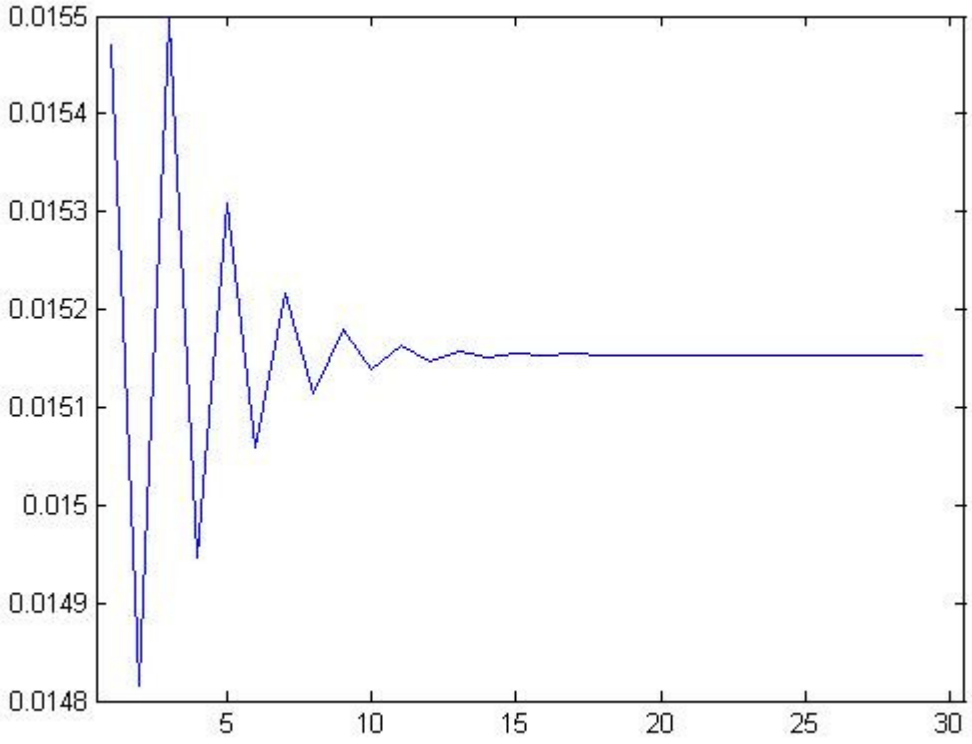


Fig. 3.3.4: graphical representation of error (cost) function.

Where the X-axis presents the number of iterations and the Y-axis presents the error.

What we see above is that the Error function E(t) decreases while *t* increasing until the stored energy reaches the minimum value. The above Error function is very small and converges after 20 iterations.

Now we have to analyze the parameters and the vectors of the algorithm (3.2-4).

$$\frac{\mathrm{d}x_{ij}}{\mathrm{d}t} = \frac{\mu}{2}\left[e_i u_j + e_j u_i + \left(\sum_{k=1}^{n} a_{ki}\, e_k\right) z_j + \left(\sum_{k=1}^{n} a_{kj} e_k\right) z_i\right]$$

$$(i, j = 1, 2, \ldots, n),$$

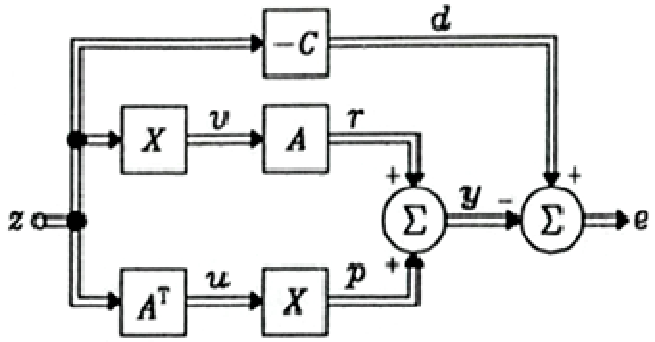


Fig. 3.2.1 Network architecture for solving the Lyapunov equation

e=d-y

y=r+p

r=A*v

p=x*u

v=x*z

$u=a^T*z$

d=-c*z

μ=learning rate=0,026

$z(t)=\sin(kw_0t)$ where k(1,2,3…) depending on the inputs

$w_0=5.00000182*10^6$

$t=10^{-1}$

The parameter t was chosen to be $t=10^{-1}$ cause we obtained quick and effective convergence.

Conclusion

In this study we have developed a methodology based on neural networks, with which we can compute the inverse of a matrix, eigenvalues of a matrix and solve linear and non linear matrix equations. This methodology can be applied to control systems which have been designed to operate in a stable manner and for some reason, through time, they turn into instability.

Our method gives us the means to determine the controllability and stability. If it is not possible to change the parameters in order to achieve stability, then we can examine in a very fast way if the system is controllable, so that we can maintain its safe operation.

This method can be applied to airplanes and nuclear reactors where stability is of paramount importance. In more revolutionary techniques the same methodology can be applied even in the case where the whole changing of all eigenvalues of the matrix A is either needed or desired.

References

[1] Neural Networks for optimization and signal processing, Andrzej Cichocki, R. Unbehauen

[2] An introduction to the automatic control theory and applications, P.N Paraskeuopoulos

[3] Synchronous systems of automatic control theory, Richard.C Dorf and Robert.H Bishop

[4] F. Albertini and E.D. Sontag. Continuous control-Lyapunov functions for asymptotically controllable time-varying systems

[5] B. Brogliato, R. Lozano, I.D. Landau, "New relationships between Lyapunov functions and the passivity theorem", Int. J. of Adaptive Control and Signal Processing

[6] http://www.stanford.edu/class/ee363/lyap.pdf

[7] http://www.unipa.it/~giarre/corso1/lec6.pdf

[8] http://citeseer.comp.nus.edu.sg/cs

[9] http://faculty.smu.edu/yzhou/publications/direct.pdf

[10] http://www.stanford.edu/class/ee363/review4.pdf

[11] http://www.ece.rutgers.edu/~gajic/psfiles/LinearLyapunov.pdf

[12] http://www.ee.ucla.edu/ee236b/lectures/problems.pdf

[13] http://ocw.mit.edu/NR/rdonlyres/Sloan-School-of-Management/15-[]084JSpring2004/7240EF84-B20D-419F-B1C0-2DAF3277F5C4/0/lec6_constr_opt.pdf

[14] http://paper.ijcsns.org/07_book/200706/20070627.pdf

[15] http://www.aueb.gr/Users/courcou/courses/ecobiz/Notes/Lagrange.pdf

[16] http://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1326&context=ecetr

[17] http://www.cs.ubc.ca/~ascher/542/chap10.pdf

[18] http://etd.caltech.edu/etd/available/etd-06262006-171822/unrestricted/thesis_master.pdf

[19] http://www.se.cuhk.edu.hk/~zhang/Courses/Seg5520/seg5520_lecture5.pdf

[20] http://etd.lib.fsu.edu/theses/available/etd-04092004-143712/unrestricted/Ch_6lms.pdf

[21] http://icapeople.epfl.ch/baltchev/papers/cdc.pdf

[22] http://www.cds.caltech.edu/~murray/courses/cds110/fa02/cds101/caltech/mls93-lyap.pdf

[23] A Linear Systems Primer, Panos J Antsaklis, Antony N Michel.

Appendix

Hessian matrix:

Let $x \in \mathbb{R}^n$ and let $f : \mathbb{R}^n \to \mathbb{R}$ be a real valued function having 2nd-order partial derivatives in an open set U containing x. The Hessian matrix of f is the matrix of second partial derivatives evaluated at x:

$$\mathbf{H}(x) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

If f is in $C^2(U)$, $H(x)$ is symmetric because of the equality of mixed partials. Note that $H(x)$, the jacobian of the gradient of f .Given a vector $\boldsymbol{v} \in \mathbb{R}^n$, the Hessian of f at $\boldsymbol{v}$ is:

$$\mathbf{H}(x)(\boldsymbol{v}) := \frac{1}{2}\boldsymbol{v}\mathbf{H}(x)\boldsymbol{v}^{\mathrm{T}}.$$

Here we view $\boldsymbol{v}$ as a 1 by n matrix so that $\boldsymbol{v}^{\mathrm{T}}$ is the transpose of $\boldsymbol{v}$.

The Hessian of f at $\boldsymbol{v}$ is a quadratic form, since

$$\mathbf{H}(x)(r\boldsymbol{v}) = r^2\mathbf{H}(x)(\boldsymbol{v}) \text{ for any } r \in \mathbb{R}.$$

If f is further assumed to be in $C^2(U)$, and x is a critical point of f such that H(x) is positive definite, then x is a strict local minimum of f.

This is not difficult to show. Since $H(x)$ is positive definite, the Rayleigh Ritz [1)] theorem shows that there is a c > 0 such that for all $h \in \mathbb{R}^n$,

$$h^T \mathbf{H}(x) h \geq 2c||h||^2$$

Thus by Taylor's theorem[2)]

$$f(x+h) = f(x) + \frac{1}{2} h^T \mathbf{H}(x) h + o(||h||^2) \geq c||h||^2 + o(||h||^2).$$

For small ||h|| the first term on the right dominates the second, so that both sides are positive for small ||h||.

1) Theorem : Rayleigh-Ritz

Let $A \in \mathbf{C}^{n \times n}$ *be a Hermittian (complex square matrix) matrix. Then its eigenvectors are the critical points (vectors) of the 'Rayleigh quotient' which is the real function* $R : \mathbb{C}^n \backslash \{\mathbf{0}\} \to \mathbb{R}$

$$R(\mathbf{x}) = \frac{\mathbf{x}^H A \mathbf{x}}{\mathbf{x}^H \mathbf{x}}, ||\mathbf{x}|| \neq 0$$

and its eigenvalues are its values at such critical points.

As a consequence we have :

$$\lambda_{max} = \max_{||\mathbf{x}|| \neq 0} \frac{\mathbf{x}^H A \mathbf{x}}{\mathbf{x}^H \mathbf{x}}$$

$$\lambda_{min} = \min_{||\mathbf{x}|| \neq 0} \frac{\mathbf{x}^H A \mathbf{x}}{\mathbf{x}^H \mathbf{x}}$$

2) *Theorem : Taylor's theorem*

Let U *be an open subset of a real banach space (X,||*|| is a normed vector space).*

If $f: U \to \mathbb{R}$ *is differiantiable n +1 times on* U*, it may be expanded by Taylor's formula:*

$$f(x) = f(a) + \mathrm{D}f(a) \cdot h + \frac{1}{2!}\mathrm{D}^2 f(a) \cdot h^2 + \cdots + \frac{1}{n!}\mathrm{D}^n f(a) \cdot h^n + R_n(x),$$

Taylor series:

The resulting infinite series is called the Taylor Series of the function $f(x)$ expanded about the point a. If it converges to the value of the function we get

$$\boxed{f(a+h) = f(a) + hf'(a) + \frac{h^2}{2!}f''(a) + \cdots + \frac{h^n}{n!}f^{(n)}(a) + \cdots}$$

If $a = 0$ then the series is known as the Maclaurin Series of the function f, which we might write as:

$$\boxed{f(x) = f(0) + xf'(0) + \frac{x^2}{2!}f''(0) + \cdots + \frac{x^n}{n!}f^{(n)}(0) + \cdots}$$

Example

What is the Maclaurin series of $f(x) = e^x$?

To write down the Maclaurin series we need to know the value at $x = 0$ of every derivative of the function. This is usually the practical problem that we face in

working out Taylor series. In this case it is easy since every derivative of e^x is e^x and this has value 1 at $x = 0$. So the Maclaurin series becomes

$$1 + x.1 + \frac{x^2}{2!}.1 + \cdots + \frac{x^n}{n!}.1 + \cdots$$

It turns out that this is actually equal to the value of e^x for any value of x (I cannot prove that here). So we have the famous result that

$$\boxed{e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots + \frac{x^n}{n!} + \cdots}$$

Lipschitz condition and Lipschitz continuity:

Definition (Lipschitz-continuous function).

Lipschitz-continuity comes in three different flavours.
Let $\boldsymbol{f} : \mathbb{R}^m \to \mathbb{R}^m$.

Given an open set $B \subseteq \mathbb{R}^m$ we say that f is Lipschitz-continuous on the open subset B if there exists a constant $\Lambda \in \mathbb{R}_0^+$ (called the Lipschitz constant of f on B) such that

$$\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})\| \leq \Lambda \|\boldsymbol{x} - \boldsymbol{y}\|, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in B. \tag{1}$$

The function f is called locally Lipschitz-continuous, if for each $z \in \mathbb{R}^n$ there exists an $L > 0$ such that f is Lipschitz-continuous on the open ball of center z and radius L

$$\mathrm{B}_L(\boldsymbol{z}) := \{\boldsymbol{y} \in \mathbb{R}^m : \ \|\boldsymbol{y} - \boldsymbol{z}\| < L\}\,. \tag{2}$$

If f is Lipschitz continuous on all of the space $\mathbb{R}^m$ (i.e. $B = \mathbb{R}^m$ in (1)), then f is called globally Lipschitz-continuous.

Remark (local vs. global)

Notice the fundamental difference between the local and global versions of the Lipschitz-continuity. Whereas in the local version the Lipschitz constant Λ and the open set B depend on each point $z \in \mathbb{R}^m$, in the global version the constant Λ is fixed and $B = \mathbb{R}^m$. In particular, a globally Lipschitz-continuous function is locally Lipschitz-continuous, but the vice versa is not true.

Remark (norms and Lipschitz constants)

In (1) the norm $\|\cdot\|$ can be any norm. However, once a norm has been chosen one should stick to that single norm, as the Lipschitz constant Λ depends on the particular choice of this norm. Unless otherwise stated, we use the Euclidean norm in all our analysis.

Interpretation

To see what these definitions mean, let us consider the situation in one dimension. Suppose f is a Lipschitz function on a neighborhood B of $x \in \mathbb{R}$. This implies that, $\forall y \in B$,

$$|f(x) - f(y)| \leq \Lambda |x - y| \ \Rightarrow \ \left| \frac{f(x) - f(y)}{x - y} \right| \leq \Lambda$$

$$\Rightarrow \ \left| \frac{f(x + h) - f(x)}{h} \right| \leq \Lambda$$

by putting y = x + h

If we were to let $h \longrightarrow 0$ and if the function f were differentiable, then the result would mean

$|f'(x)| \leq \Lambda$, (the derivative is bounded by the Lipschitz constant).

However, there is nothing in the definition of Lipschitz-continuity that implies that f is differentiable. So in general we can't proceed to this limit, since we don't know if f is differentiable at x. But this tells us all we need to know: being Lipschitz just means f can't be too steep, the bound on the difference quotient being Λ.

The Lyapunov Matrix Equation:

$$\dot{x} = Ax. \tag{3}$$

The derivative can be determined *without explicitly solving for the solutions of (3) by noting that*

$$\begin{aligned}\dot{v}(x) &= \dot{x}^T x + x^T \dot{x} = (Ax)^T x + x^T (Ax) \\ &= x^T (A^T + A)x.\end{aligned}$$

If the matrix A is such that *u'(x)* is negative for all $x \neq 0$ then it is reasonable to expect that the distance of the state of (3) from *x=0* will decrease with increasing time and that the state will therefore tend to the equilibrium *x=0* of (3) with increasing time *t*.

In the following discussion we will employ as a '' *generalized distance function* '' the quadratic form given by:

$$v(x) = x^T P x, \quad P = P^T, \tag{4}$$

Where P is a real *n x n* matrix .The time derivative of *u(x)* along the solutions of (3) is determined as:

$$\begin{aligned}\dot{v}(x) &= \dot{x}^T P x + x^T P \dot{x} = x^T A^T P x + x^T P A x \\ &= x^T (A^T P + P A)x;\end{aligned}$$

$$\dot{v} = x^T C x, \tag{5}$$

where

$$C = A^T P + P A. \tag{6}$$

Note that C is real and $C^T = C$. The system of equations given in (6) is called the *Lyapunov Matrix Equation*.

We recall that since P is real and symmetric, all its eigenvalues are real.

Theorem 1: The equilibrium x=0 of (3) is stable if there exists a real, symmetric, and positive definite n x n matrix P such that the matrix C given in (6) is negative semidefinite.

Proof. Along any solution

$$\phi(t, x_0) \triangleq \phi(t)$$

of (3) with

$$\phi(0, x_0) = \phi(0) = x_0,$$

we have:

$$\phi(t)^T P\phi(t) = x_0^T Px_0 + \int_0^t \frac{d}{d\eta}\phi(\eta)^T P\phi(\eta)d\eta = x_0^T Px_0 + \int_0^t \phi(\eta)^T C\phi(\eta)d\eta$$

for all $t \geq 0$.Since *P* is positive definite and *C* is negative definite we have:

$$\phi(t)^T P\phi(t) - x_0^T Px_0 \leq 0$$

for all $t \geq 0$. And there exist:

$$c_2 \geq c_1 > 0$$

such that

$$c_1 \parallel \phi(t) \parallel^2 \leq \phi(t)^T P\phi(t) \leq x_0^T Px_0 \leq c_2 \parallel x_0 \parallel^2$$

for all $t \geq 0$. It follows that

$$\parallel \phi(t) \parallel \leq (c_2/c_1)^{1/2} \parallel x_0 \parallel$$

for all $t \geq 0$ and for any $x_0 \varepsilon R^n$. Therefore the equilibrium *x=0* of (3) is stable.

Theorem 2: The equilibrium x=0 of (3) is exponentially stable in the large if there exists a real, symmetric, and positive definite n x n matrix P such that the matrix C given in (6) is negative definite.

Proof. We let:

$$\phi(t, x_0) \triangleq \phi(t)$$

denote an arbitrary solution of (3) with

$$\phi(0) = x_0$$

In view of the hypotheses of the theorem, there exist constants

$$c_2 \geq c_1 > 0$$

and

$$c_3 \geq c_4 > 0$$

such that:

$$c_1 \parallel \phi(t) \parallel^2 \leq v(\phi(t)) = \phi(t)^T P \phi(t) \leq c_2 \parallel \phi(t) \parallel^2$$

and

$$-c_3 \parallel \phi(t) \parallel^2 \leq \dot{v}(\phi(t)) = \phi(t)^T C \phi(t) \leq -c_4 \parallel \phi(t) \parallel^2$$

for all $t \geq 0$ and for any $x_0 \in R^n$.

Then

$$\begin{aligned} \dot{v}(\phi(t)) &= \frac{d}{dt}[\phi(t)^T P \phi(t)] \leq (-c_4/c_2)\phi(t)^T P \phi(t) \\ &= (-c_4/c_2) v(\phi(t)) \end{aligned}$$

for all $t \geq t_0$

This implies, after multiplication by the appropriate integrating factor, and integrating from 0 to *t*, that

$$v(\phi(t)) = \phi(t)^T P \phi(t) \leq x_0^T P x_0 e^{-(c_4/c_2)t}$$

or

$$c_1 \parallel \phi(t) \parallel^2 \leq \phi(t)^T P \phi(t) \leq c_2 \parallel x_0 \parallel^2 e^{-(c_4/c_2)t}$$

or

$$\parallel \phi(t) \parallel \leq (c_2/c_1)^{1/2} \parallel x_0 \parallel e^{-\frac{1}{2}(c_4/c_2)t}, \quad t \geq 0.$$

This inequality holds for all $x_0 \ \varepsilon \ R^n$. Therefore, the equilibrium $x=0$ of (3) is exponentially stable in the large.

Theorem 3: The equilibrium x=0 of (3) is unstable if there exists a real, symmetric n x n matrix P that is either negative definite or indefinite such that the matrix C given in (6) is negative definite.

Proof: We first assume that P is indefinite. Then P possesses eigenvalues of either sign, and every neighborhood of the origin contains points where the function:

$$v(x) = x^T P x$$

is positive and negative. Consider the neighborhood

$$B(\epsilon) = \{x \in R^n : \| x \| < \epsilon\},$$

where $\| \cdot \|$ denotes the Euclidean norm and let

$$G = \{x \in B(\epsilon) : v(x) < 0\}.$$

On the boundary of G we have either $\| x \| = \varepsilon$ or $u(x) = 0$. In particular, note that the origin $x=0$ is on the boundary of G. Now, since the matrix C is negative definite, there exist constants c3 > c4 > 0 such that

$$-c_3 \| x \|^2 \leq x^T C x = \dot{v}(x) \leq -c_4 \| x \|^2$$

for all $x \in R^n$. Let

$$\phi(t, x_0) \triangleq \phi(t)$$

and

$$x_0 = \phi(0) \in G$$

Then $u(x_0) = -\alpha < 0$. The solution $\phi(t)$ starting at $x=0$ must leave the set G. To see this, note that as long as

$$\phi(t) \in G, v(\phi(t)) \leq -a$$

since

$$\dot{v}(x) < 0 \qquad \text{in } G \ .$$

Let $-c = sup \{u'(x): x \in G \text{ and } u(x) < -\alpha\}$.

Then $c>0$ and

$$v(\phi(t)) = v(x_0) + \int_0^t \dot{v}(\phi(s))ds \leq -a - \int_0^t cds$$
$$= -a - tc, t \geq t_0.$$

The inequality shows that $\Phi(t)$ must escape the set G in finite time because $u(x)$ is bounded from below on G. But $\Phi(t)$ cannot leave G through the surface determined by $u(x) = 0$ since

$$v(\phi(t)) \leq -a.$$

Hence, it must leave G through the sphere determined by $\| x \| = \varepsilon$. Since the above argument holds arbitrarily small $\varepsilon > 0$, it follows that the origin $x=0$ of (3) is unstable.

Next we assume that P is negative definite. Then G as defined is all of $B(\varepsilon)$. The proof proceeds as above.

The proof of the above theorem shows that for $\varepsilon > 0$ sufficiently small when P is negative definite, all solutions $\Phi(t)$ of (3) with initial conditions $x_0 \in B(\varepsilon)$ will tend away from the origin. This constitutes a severe case of instability called complete instability.

Theorem 4: Assume that the matrix A [for the system (3)] has no eigenvalues with real part equal to zero. If all the eigenvalues of A have negative real parts or if at

least one of the eigenvalues of A has a positive real part then there exists a quadratic Lyapunov function

$$v(x) = x^T P x, \quad P = P^T,$$

whose derivative along the solutions of (3) is definite (it is either negative or positive definite).

This result shows that when A is a stable matrix, then from (3) the conditions of *Theorem 1* are also necessary conditions for stability. Moreover, in the case when the matrix *A* has at least one eigenvalue with positive real part and no eigenvalues on the imaginary axis, then the conditions of *Theorem 3* are also necessary conditions for instability.

Controllability:

A system with internal state vector *x* is called controllable if and only if the system states can be changed by changing the system input.

$x' = Ax + Bu$

x is the state vector,

u is the input or control vector,

A is the state matrix,

B is the input matrix.

The controllability matrix is given by:

$R = [\, B \;\; AB \;\; A^2B \;\; \ldots \;\; A^{n-1}B]$.

The system is controllable if the controllability matrix R has a full row rank of *p* where *p* is the dimension of the matrix A, and *p x q* is the dimension of matrix B.