Stability Analysis of Dynamic Fuzzy System

A Dissertation

Submitted in partial fulfillment of the requirements for the award of

Degree of

MASTERS OF ENGINEERING IN

ELECTRICAL ENGINEERING (*Control & Instrumentation*)

Under the Guidance of

Dr. Narendra Kumar
Assistant Professor

Submitted By

Siba Brata Panda

Department of Electrical Engineering,
Delhi College of Engineering, University of Delhi

CERTIFICATE

This is to certify that the dissertation entitled " **STABILITY ANALYSIS OF DYNAMIC FUZZY SYSTEM** " , which is being submitted by **Siba Brata Panda** in partial fulfillment of the award of ME Degree in Electrical Engineering with specialization in Control and instrumentation of Delhi College of Engineering, Delhi, is a record of students own work carried out by him under my guidance and supervision. The matter embodied in this dissertation has not been submitted for the award of any degree to the best of my knowledge and belief.

Dr. Narender Kumar
Assistant Professor
Electrical Engineering Department
Delhi College of Engineering, Delhi

ACKNOWLEDGEMENT

I express my foremost and deepest gratitude to Dr. Narendra kumar,Professor, Department of Electrical Engineering, Delhi College of Engineering, Delhi-42,University of Delhi for his guidance, support & encouragement throughout my dissertation . I consider myself fortunate for having the opportunity to learn and work under his valuable guidance over the entire period of association with him. I have deep sense of admiration for his innate goodness. I can offer my profound indebtedness to him for his deep concern for my academics. My sincere thanks to Dr. Pramod Kumar for his support and encouragement.

Lastly my deepest gratitude is due to Almightily God whose divine light provided me the perseverance, guidance, inspiration and strength to complete this work.

Siba Brata Panda

ABSTRACT

Fuzzy control system can be described as a real-time expert system, implementing a part of a human operator's or process engineer’s expertise. A representation theorem mainly due to Kosko, states that any continuous nonlinear function can be approximated as exactly as needed with a finite set of fuzzy variables, values, and rules. In general the Lyapunov approach reveals only the existence of stable points, not their number or nature [Kosko'92].There are a few stability test methods in the linguistic fuzzy dynamic models represented by ‘if-then’ rules that can be used for modeling real plants. Tanaka and Sugeno [Tanaka'92] suggested an effective method for determining stability. Another method, which has been suggested by Kim et al. [Kim'95] treats the fuzzy model as a linear system having modeling uncertainties. It suggests a new Lyapunov method for determining stability that is less restrictive than the previous one..

The second chapter is followed by theorems giving a sufficient condition of stability for fuzzy systems in the sense of Lyapunov. An example follows which shows that the overall system may be unstable even if all the subsystems are stable. Following this is a MATLAB program which plots the behaviour of the subsystems and the overall system as given in the example. The chapter ends with giving a necessary condition for the existence of a common positive definite matrix satisfying Lyapunov equations for all the subsystem matrices.The third chapter starts with presenting a fuzzy state space model, which can approximate non-linear systems. A theorem follows concerning the stability of this model, which again uses Lyapunov's direct method. This way of testing the stability has been termed negative bounds approach as suggested by the procedure involved. This is followed by an example, which shows that the overall system may be stable even if some of them are unstable. Following this is a MATLAB program, which simplifies some related calculations. A MATLAB program is given at the end of the chapter whose sample run shows how a stable system is identified to be stable after the sixth iteration.The fourth chapter,

in the beginning, discusses how linear feedback control affects the equations given in chapter 3. This is followed by a MATLAB program giving values of the control gain, K, which can be used for stabilizing a fuzzy system having nine rules.

The following important conclusions can be drawn from the discussions: If all the subsystem matrices are stable, the overall fuzzy system may or may not be stable. Even if some of the subsystem matrices are unstable, the overall fuzzy system may be stable. The existence of a common positive definite matrix satisfying Lyapunov equation for all the subsystem matrices shows that the fuzzy system is stable. If all the subsystem matrices are stable, but the product of any two subsystem matrices is unstable, there can't be any common positive definite matrix satisfying Lyapunov equation for all the satisfying Lyapunov equation for all the subsystem matrices. Negative bounds approach used to test the stability of fuzzy systems is better than common positive definite matrix approach in the sense that it does not fail even if some of the subsystem matrices are unstable. Though the negative bounds approach gives only a sufficient condition of stability like common positive definite matrix approach, an algorithm based on this approach can be used to identify some stable systems as stable.In case of stabilization using linear feedback control, negative bounds approach can be used for the determination of control gains, which can stabilize the system.

Literature Review

Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore 639798.Existing Takagi-Sugeno-Kang (TSK) fuzzy models proposed in the literature attempt to optimize the global learning accuracy as well as to maintain the interpretability of the local models. Most of the proposed methods suffer from the use of offline learning algorithms to globally optimize this multi-criteria problem. Despite the ability to reach an optimal solution in terms of accuracy and interpretability, these offline methods are not suitably applicable to learning in adaptive or incremental systems. Furthermore, most of the learning methods in TSK-model are susceptible to the limitation of the curse-of-dimensionality. This work attempts to study the criteria in the design of TSK-models. They are: 1) the interpretability of the local model; 2) the global accuracy; and 3) the system dimensionality issues. A generic framework is proposed to handle the different scenarios in this design problem. The framework is termed the generic fuzzy input Takagi-Sugeno-Kang fuzzy framework (FITSK) their performances are encouraging when benchmarked against other popular fuzzy systems

Electrical Engineering Department, University of Nevada, Reno, NV 89512, USA propose a new approach for the stability analysis of continuous Sugeno Types II and III dynamic fuzzy systems. They introduce the concept of fuzzy positive definite and fuzzy negative definite systems and use them in arguments similar to those of traditional Lyapunov stability theory to derive new conditions for stability and asymptotic stability for continuous Type II/III dynamic fuzzy systems. To demonstrate the new approach, they apply it to numerical examples.

The stability analysis of a generalized class of continuous fuzzy systems in terms of Lyapunov stability theory is presented. Firstly, the stability problem of fuzzy systems described by Takagi-Sugeno's continuous model is stated. Secondly, new stability conditions which guarantee the stability of the fuzzy system are derived. The new stability conditions can be regarded as a general solution for Takagi-Sugeno fuzzy system, in which the offset term is not equal to zero. Finally, the suggested stability theorems are verified by some illustrative examples. Fuzzy Sets and Systems Volume 129 , Issue 3 (August 2002).

This paper presents a stability analysis method for discrete-time Takagi-Sugeno fuzzy dynamic systems based on a piecewise smooth Lyapunov function. It is shown that the stability of the fuzzy dynamic system can be established if a piecewise Lyapunov function can be constructed, and moreover, the function can be obtained by solving a set of linear matrix inequalities that is numerically feasible with commercially available software. It is also demonstrated via numerical examples that the stability result based on the piecewise quadratic Lyapunov functions is less conservative than that based on the common quadratic Lyapunov functions. This paper appears in: **Fuzzy Systems,IEEE,Transactions,on**.PublicationDate:Feb.2004Volume:12,Issue:1On page(s):22-28ISSN:1063-6706

CONTENTS

List of Figures

List of Programs

Chapter 1

Introduction

1.1 Fuzzy Control Systems

A fuzzy control system can be described as a real-time expert system, implementing a part of a human operator's or process engineer’s expertise, which does not lend itself to being easily expressed in PID-parameters or differential equations but rather in situation or action rules [Driankov'93].

Fuzzy control differs from mainstream expert system technology in several aspects. Fuzzy control systems exist at two distinct levels: there are symbolic if-then rules and, qualitative fuzzy variables and values such as:

if temperature is high *and* slightly increasing *then* energy supply is medium negative.

The above rule is nothing but an informal ‘nonlinear PD-element’. A collection of such rules can be used and, in fact, results in the definition of a nonlinear transition function, without the need for defining each entry of the table individually, and without necessarily knowing the closed form representation of that function. One way to combine fuzzy and PID-control then is to use a linear PID-system around the set point, where it does its job, and to ‘delinearize’ the system in other areas by describing the desired behaviour or control strategy with fuzzy rules.

A representation theorem mainly due to Kosko, states that any continuous nonlinear function can be approximated as exactly as needed with a finite set of fuzzy variables, values, and rules. This theorem describes the representational power of fuzzy control in principle, but it does not answer the questions, how many rules are needed and how they can be found, which. are of course essential to the real world problems and solutions. In many cases, relatively small and simple systems will do, and that is why already several hundreds of real, industrial applications of fuzzy control exist.

The fuzzy values such as ‘slightly increasing’ and fuzzy operators such as ‘and’ are compiled into very elementary numerical objects and algorithms: function tables, interpolations, comparators, etc. The existence of this compiled level is the basis for fast real-time implementations, as well as for embedding fuzzy control into the essentially numerical environment of conventional control.

Fuzzy control has right from the beginning been considered as an extension to existing technology, seeking hybrid solutions by enhancing control engineering where it is needed and where it makes sense. In fact, most of the inventors of fuzzy control have a strong control engineering or systems theory background. From their perspective, fuzzy control can be seen as a heuristic and modular way for defining nonlinear, table-based control systems.

The industrial interest in fuzzy control, which hitherto has not been recognized as a serious discipline, has been dramatically increasing since 1990. There are still, however, two predominant, extreme positions as to the benefits of fuzzy control. On one hand, many proponents of this technology claim that fuzzy control will revolutionize control engineering, promises major breakthroughs, and will be able to solve complex engineering problems with very little effort. On the other, many representatives of the control engineering community still proclaim the philosophy that "everything that can be done in fuzzy control can be done conventionally as well," and announce a breakdown of the ‘fuzzy hype’ in the near future.

The insight that neither of the two positions accounts for the real

potential of fuzzy control is only gradually increasing. In many cases, fuzzy control leads to a higher degree of automation for complex, ill structured processes, but only if there is relevant knowledge about the process and its

control available that can be well expressed in terms of fuzzy logic. There are processes for which that kind of knowledge simply is not at all or not to the necessary extent available. Secondly, fuzzy controllers are more robust than conventional controllers in many applications. But, there are other cases, too. We know of two attempts to control air conditioning systems with fuzzy logic, with only minor differences in structure and knowledge base: one turned out to be highly robust even in the presence of major disturbances, the other one was unstable. It is not yet fully understood for which kinds of control engineering problems fuzzy control really leads to improved robustness and stability, and which are the relevant design choices that affect these properties.

1.2 Stability Issues

How do we prove stability for a system defined with arbitrarily many interlocked differential or difference equations? The first, or direct, approach ‘simply’ solves the equations and then studies how the system evolves with time. This is seldom feasible in the high-dimensional nonlinear case.

The second approach finds a Lyapunov function. The Lyapunov approach offers a shortcut to proving a global stability of a dynamical system. If we cannot find a Lyapunov function, nothing follows. The dynamical system mayor may not be stable.

But if we can find a Lyapunov function, stability holds. Often, though, we cannot establish anything else. In general the Lyapunov approach reveals only the existence of stable points, not their number or nature [Kosko'92].

There are a few stability test methods in the linguistic fuzzy dynamic models represented by ‘if-then’ rules that can be used for modeling real plants. Tanaka and Sugeno [Tanaka'92] suggested an effective method for determining stability. They dealt with a model that can be well identified with input-output data. It is suggested that if there exist a common solution matrix of the Lyapunov equations for all rules, then the model is stable. This method, however, does not provide a systematic way to find a Lyapunov function, and it requires heavy computational load as a result. Furthermore, some of the stable models may not be identified as stable. Another method, which has been suggested by Kim et al. [Kim'95] treats the fuzzy model as a linear system having modeling uncertainties. It suggests a new Lyapunov method for determining stability that is less restrictive than the previous one. It has been shown with the help of an example that the stability of a model, which is not determined by the former method, can be determined by the latter one. We shall investigate both the methods in the forthcoming chapters.

It is not true as opponents of fuzzy control often argue that there are no stability criteria available for fuzzy control systems.

We have to realize that in this respect, fuzzy control competes with nonlinear conventional control, where stability issues are not as easy to handle as for simple linear systems.

1.3 Organization and Outline of the Chapters

The next chapter begins with discussing a special case concerning continuous systems, which shows that Hermitian matrices as subsystem matrices are easier to deal with when stability of the overall system is to be determined using eigenvalue conditions. This is followed by theorems giving a sufficient condition of stability for fuzzy systems in the sense of Lyapunov. An example follows which shows that the overall system may be unstable even if all the subsystems are stable. Following this is a MATLAB program which plots the behaviour of the subsystems and the overall system as given in the example. The chapter ends with giving a necessary condition for the existence of a common positive definite matrix satisfying Lyapunov equations for all the subsystem matrices.

The third chapter starts with presenting a fuzzy state space model, which can approximate non-linear systems. A theorem follows concerning the stability of this model, which again uses Lyapunov's direct method. This way of testing the stability has been termed negative bounds approach as suggested by the procedure involved. This is followed by an example, which shows that the overall system may be stable even if some of them are unstable. Following this is a MATLAB program, which simplifies some related calculations. An iterative procedure follows to identify a stable system as stable, as the stability condition given in this chapter is again not a necessary condition. A MATLAB program is given at the end of the chapter whose sample run shows how a stable system is identified to be stable after the sixth iteration.

The fourth chapter, in the beginning, discusses how linear feedback control affects the equations given in chapter 3. This is followed by a MATLAB program giving values of the control gain, K, which can be used for stabilizing a fuzzy system having nine rules. Another MATLAB program follows which plots the behaviour of the same systems for any given value of K.

Chapter 5 presents concluding remarks. Limitations of the proposed methods and suggestions for future research are also given in this chapter.

Chapter 2

Stability Test Method-1

(Common Positive Definite Matrix Approach)

2.1 Introduction

For a given control system, stability is usually the most important attribute to be determined. The Lyapunov's method of stability analysis is, in principle, the most general method for the determination of the stability of non-linear and/or time-varying systems. Fuzzy systems are basically non-linear in nature. Theorems are derived in this chapter for the stability of a fuzzy system in accordance with the definition of stability in the sense of Lyapunov. A sufficient condition which guarantees the stability of a fuzzy system is obtained in terms of Lyapunov's direct method.

2.2 Eigenvalue Conditions for the Stability of Dynamical Systems

How a system evolves with time is very closely attached to the nature of the eigenvalues of the system matrix. For continuous case, all the eigenvalues having negative real parts show a stable system; and for discrete case, all the eigenvalues must lie within the unit circle in the z-plane for the system to be stable. But prediction about the behaviour of the overall system is very difficult by just looking at the eigenvalues of the subsystem matrices. As a simple example,

Let

$$A_1 = \begin{bmatrix} -1 & -1 \\ 2 & -4 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -5 & 0 \\ -102 & -4 \end{bmatrix}$$

The eigenvalues of A_1 are -2, -3 and those of A_2 are –4, -5. But the eigenvalues of A_1+A_2 are 3.0499, -17.0499. We see that one eigenvalue of A_1+A_2 has a positive real part. It is also interesting to note that the eigenvalues of A_1+A_2 are 122.0165, 0.9835.

Now we shall discuss a special case concerning continuous systems. From matrix algebra [Horn and Johnson] it comes out that if A_1 and A_2 are Hermitian matrices of order n (Hermitian matrices are conjugates of their own transposes.), and if the eigenvalues of A_1+A_2 and A_1+A_2 are arranged in algebraically decreasing order, then

$$\sum_{i=1}^{k} \lambda_i(A_1 + A_2) \le \sum_{i=1}^{k} [\lambda_i(A_1) + \lambda_i(A_2)] \qquad (2.1)$$

for k = 1, 2, ..., n. Where λ_i denotes its eigenvalue of the given matrix.

From the above inequality we get the following inequalities:

$$\lambda_1(A_1 + A_2) \le \lambda_1(A_1) + \lambda_1(A_2) \qquad (2.1.1)$$

$$\lambda_1(A_1 + A_2) + \lambda_2(A_1 + A_2) \le \lambda_1(A_1) + \lambda_1(A_2) + \lambda_2(A_1) + \lambda_2(A_2) \qquad (2.1.2)$$

From inequality 2.1.1, if $\lambda_1(A_1)$ and $\lambda_1(A_2)$ are both negative, $\lambda_1(A_1+A_2)$ will obviously be more negative. From inequalities 2.1.1 and 2.1.2, it follows that $\lambda_1(A_1+A_2)$ will also be negative if $\lambda_1(A_1)$ and $\lambda_1(A_2)$ are both negative too. Thus we see that Hermitian matrices as subsystem matrices are easier to deal with than others.

2.3 Lyapunov Stability Criteria

Following is the well-known Lyapunov theorem:

Theorem 2.1:

Consider a discrete system described by $x(k+1) = f(x(k))$,

Where $x(k) \in R^n$, $f(x(k))$ is an $n \times 1$ function vector with the property that $f(0) = 0$ for all k. Suppose that there exists a scalar function $V(x(k))$ continuous in $x(k)$ such that

(a) $V(0) = 0$,

(b) $V(x(k)) > 0$ for $x(k) \neq 0$,

(c) $V(x(k))$ *approaches infinity as* $\|x(k)\| \to \infty$,

(d) $\Delta V(x(k)) < 0$ *for* $x(k) \neq 0$.

Then the equilibrium state $x(k) = 0$ for all k is asymptotically stable in the large and $V(x(k))$ is Lyapunov function.

The following theorem gives Lyapunov equation for discrete case [Kuo'80]:

Theorem 2.2:

For a linear system, a necessary and sufficient condition that the equilibrium state $x = 0$ be asymptotically stable in the large is that, given any positive definite Hermitian (or real symmetric) matrix Q, there exists a positive definite Hermitian (or real symmetric) matrix P, such that

$A^T PA - P = -Q$

Proof:

If P is a positive definite matrix,

$V(x(k)) = x^T(k)\, P\, x(k)$ is also positive definite. (Sylvester's theorem)

$$\Delta V(x(k)) = V(x(k+1)) - V(x(k))$$
$$= x^T(k+1)\, P\, x(k+1) - x^T(k)\, P\, x(k)$$

But, $x(k+1) = A\, x(k)$,

Therefore,

$$\Delta V(x(k)) = x^T(k) A^T\, P\, A\, x(k) - x^T(k)\, P\, x(k)$$
$$= x^T(k)\, [A^T\, P\, A - P]\, x(k)$$
$$= -\, x^T(k)\, Q\, x(k),$$

which shows that,

$A^T\, P\, A - P = -Q.$

Asymptotic stability often corresponds to an eigenvalue condition in engineering settings, a practice we shall follow. In particular, a dynamical system (continuous case) is asymptotically stable if and only if the Jacobian matrix of the dynamical system has eigenvalues with negative real parts. A general theorem in dynamical systems theory relates convergence rate to eigenvalues sign.

A nonlinear dynamical system converges exponentially quickly if its system Jacobian has eigenvalues with negative real parts. Locally such nonlinear systems behave linearly.

For discrete case, this corresponds to the eigenvalues lying inside the unit circle in the z-plane.

2.4 Takagi and Sugeno's Fuzzy Model

The fuzzy model suggested by Takagi & Sugeno is of the following form:

L^i: IF $x(k)$ is A_1^i and … and $x(k–n+1)$ is A_n^i and
$u(k)$ is B_1^i and … and $u(k–m+1)$ is B_m^i

THEN $x^i(k+1) = a_o^i + a_1^i x(k) + \ldots + a_n^i x(k–n+1) + b_1^i u(k) + \ldots\ldots$
$+b_m^i u(k–m+1)$ (2.2)

where L^i ($i = 1, 2, \ldots , l$) denotes the i-th implication; l is the number of i-th implications; $x^i(k+1)$ is the output from the i-th implication, a_p^i ($p = 0,1, \ldots\ldots,n$) and a_q^i ($q = 0,1,\ldots\ldots, m$) are consequent parameters; $x(k), \ldots , x(k- n+1)$ are state variables; $u(k), \ldots\ldots\ldots,u(k–m+1)$ are input variables and A_p^i and B_q^i are fuzzy sets whose membership functions denoted by the same symbol are continuous piecewise-polynomial functions.

Given an input $(x(k), x(k-1), \ldots ,x(k-n+1), u(k), u(k–1), \ldots,u(k-m+1))$ the final output of a fuzzy model is inferred by taking the weighted average of the $x^i(k+1)$'s:

$$x(k+1) = \sum_{i=1}^{l} w^i x^i(k+1) / \sum_{i=1}^{l} w^i \qquad (2.2)$$

where $\sum_{i=1}^{l} w^i > 0$, and $x^i(k+1)$ is calculated for the input by the consequence equation of the i-th implication, and the weight w^i implies the over all truth value of the premise of the I-th implication for the input, calculated as

$$w^i = \prod_{p=1}^{n} A_p^i(x(k-p+1)) x \prod_{q=1}^{m} B_q^i(u(k-q+1)), \qquad (2.4)$$

A set of fuzzy implications shown in eq. (2.2) can express a highly nonlinear functional relation in spite of a small number of fuzzy implications.

2.5 Stability of Fuzzy Systems in the Sense of Lyapunov: Common Positive Definite Matrix Approach

We derive theorems for the stability of a fuzzy system in accordance with the definition of stability in the sense of Lyapunov. A sufficient condition, which guarantees the stability of fuzzy system, is obtained in terms of Lyapunov's direct method.

Let us consider the following fuzzy free system:

L^i: IF x(k) is A_1^i and … and x(k–n+1) is A_n^i

THEN $x^i(k+1) = a_1^i\,(k) + \ldots a_n^i\,x(k-n+1)$,

Where i = 1,2 …, *l*. The linear subsystems in the consequent part of the i-th implication can be written in the matrix form $A_i x(k)$,

Where $x(k) \in R^n \times R^n$,

$x(k) = [x(k), x(k-1), \ldots..x(k-n+1)]^T$, and

$$A_i = \begin{bmatrix} a_1^i & a_2^i & \cdots & a_{n-1}^i & a_n^i \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ . & . & & . & . \\ . & . & & . & . \\ 0 & 0 & & 0 & 0 \\ 0 & 0 & & 1 & 0 \end{bmatrix}$$

The output of the fuzzy system is inferred as follows:

$$x(k+1) = \sum_{i=1}^{l} w^i A_i x(k) / \sum_{i=1}^{l} w^i ,$$

where *l* is the number of fuzzy implications.

Using previous theorems and lemma 2.3.1, we derive theorem 2.3, which is an important theorem concerning the stability of a fuzzy system.

Theorem 2.3:

The equilibrium of a fuzzy system is globally asymptotically stable if there exists a common positive definite matrix P for all the subsystems such that

$A_i^T P A_i - P < 0 \; for \; I \in \{1,2,\ldots, l\}$

Proof:

Let $V(x(k)) = x^T(k)Px(k)$ be a scalar function where P is a positive definite matrix such that

(a) $V(0) = 0,$

(b) $V(x(k)) > 0$ for $x(k) \neq 0,$

(c) $V(x(k)) \to \infty$ as $\| x(k) \| \to \infty.$

Now $\Delta V(x(k)) = V(x(k+1)) - V(x(k))$

$$= x^T(k+1)\, Px(k+1) - x^T(k)Px(k)$$

$$= \sum_{i=1}^{l} w^i A_i\, x(k) / \sum_{i=1}^{l} w^{i\,T} P(\sum_{i=1}^{l} w^i A_i\, x(k)) / \sum_{i=1}^{l} w^i) - x^T(k)Px(k)$$

$$= x^T(k)\{\sum_{i=1}^{l} w^i A_i^T / \sum_{i=1}^{l} w^i)P\left(\sum_{i=1}^{l} w^i A / \sum_{i=1}^{l} w^i \right) - P\}x(k)$$

$$= = \sum_{i=1}^{l} w^i w^j x^T(k)\{A_i{}^T P A_j - P\}x(k) / \sum_{i=1}^{l} w^i w^j$$

$$= \sum_{i=1}^{l} (w^i)^2 x^T(k)\{A_i{}^T P A_j - P\}xx(k) + \sum_{i=1}^{l} w^i w^j x^T(k)\{A_i{}^T P A_j$$

$$+ A_j{}^T P A_i - 2P\}x(k)] / \sum_{i=1}^{l} w^i w^j$$

From lemma 2.3.1, the statement of the theorem, and the conditions that

$w^i \geq 0$ and $\sum_{i=1}^{l} w^i > 0,$

it follows that

$\Delta V(x(k)) < 0$.

Lemma 2.3.1:

If P is a positive definite matrix such that

$A^T PA - P < 0$

$B^T PB - P < 0,$

Where A, B, P $\in R^{nxn}$, *then*

$A^T PB + B^T PA - 2P < 0.$

Proof:

$$-(A-B)^T P(A-B) - 2P = -(A^T - B^T)P(A-B) - 2P$$
$$= -A^T PA + A^T PB + B^T PA - B^T PB - 2P$$

or

$$A^T PB + B^T PA - 2P = (A-B)^T P(A-B) + A^T PA + B^T PB - 2P$$
$$= -(A-B)^T P(A-B) + A^T PA - P + B^T PB - P$$

As P is positive definite,

$-(A-B^T P(A-B) \leq 0.$

Thus, the conclusion of the lemma follows.

V(x(k)) is a Lyapunov function and the fuzzy system is globally asymptotically stable. This theorem is reduced to the Lyapunov stability theorem for linear discrete systems when $l = 1$.

This theorem can be applied to the stability analysis of a nonlinear system which is approximated by a piecewise linear function if the given conditions are satisfied.

We can point out that a piecewise linear function can be described as a special case of eq.(2.2) if we use crisp sets instead of fuzzy sets in the premise parts of a fuzzy system. It is easy to divide a nonlinear system into some linearized subsystems on an input state space. This means that the system is approximated by a piecewise linear system. Since many nonlinear systems can be approximated by piecewise linear functions, this theorem can be widely applied not only to a fuzzy system, but also to nonlinear systems.

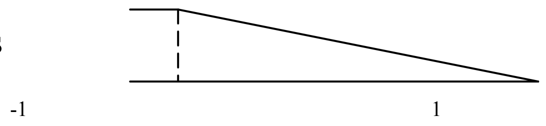
Theorem 2.3 is, of course, a sufficient condition for ensuring the stability of system (2.2). We may intuitively guess that an approximated nonlinear system is stable if all locally approximating nonlinear systems are stable. However, it is not the case in general. Here we notice the following fact.

All the Ai's are stable matrices if there exists a common positive definite matrix P. There does not always exist a common positive definite matrix P even if all the Ai's are stable matrices. Of course, a fuzzy system may be globally asymptotically stable even if there does not exist a common positive definite matrix P. However, we must notice that a fuzzy system is not always globally asymptotically stable even if all the Ai's are stable matrices as shown in the forthcoming example.

Example: 2.1:

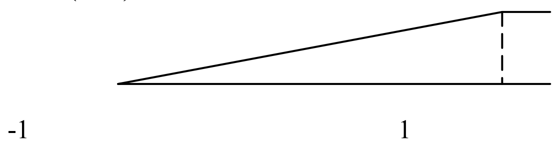
Let us consider the following fuzzy system

L^1: IF x(x-1) is



THEN x (k+1) = x(k) – 0.5 x(k-1)

L^2:IF x(x-1) is



THEN x (k+1) = x(k) – 0.5 x(k-1)

Initial conditions are given by

X(0) = - 0.7

X(1) = 0.9

Membership values are calculated as follows:

$W^1 = (1-x(k-1))/2$ for $-1 \leq x(k-1) \leq 1$

$= 1$ for $x(k-1) \leq 1$

$= 0$ otherwise

$W^2 = (1+x(k-1))/2$ for $-1 \le x(k-1) \le 1$

$= 1$ for $x(k-1) > 1$

$= 0$ otherwise

For the linear subsystems, we obtain

$$A_1 = \begin{bmatrix} 1 & -0.5 \\ 1 & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -1 & 0.5 \\ 1 & 0 \end{bmatrix}$$

With the help of the problem specific program the near system obtain plots 2.1, 2.2 and 2.3. It can be seen that both the stable but the overall fuzzy system is not stable.

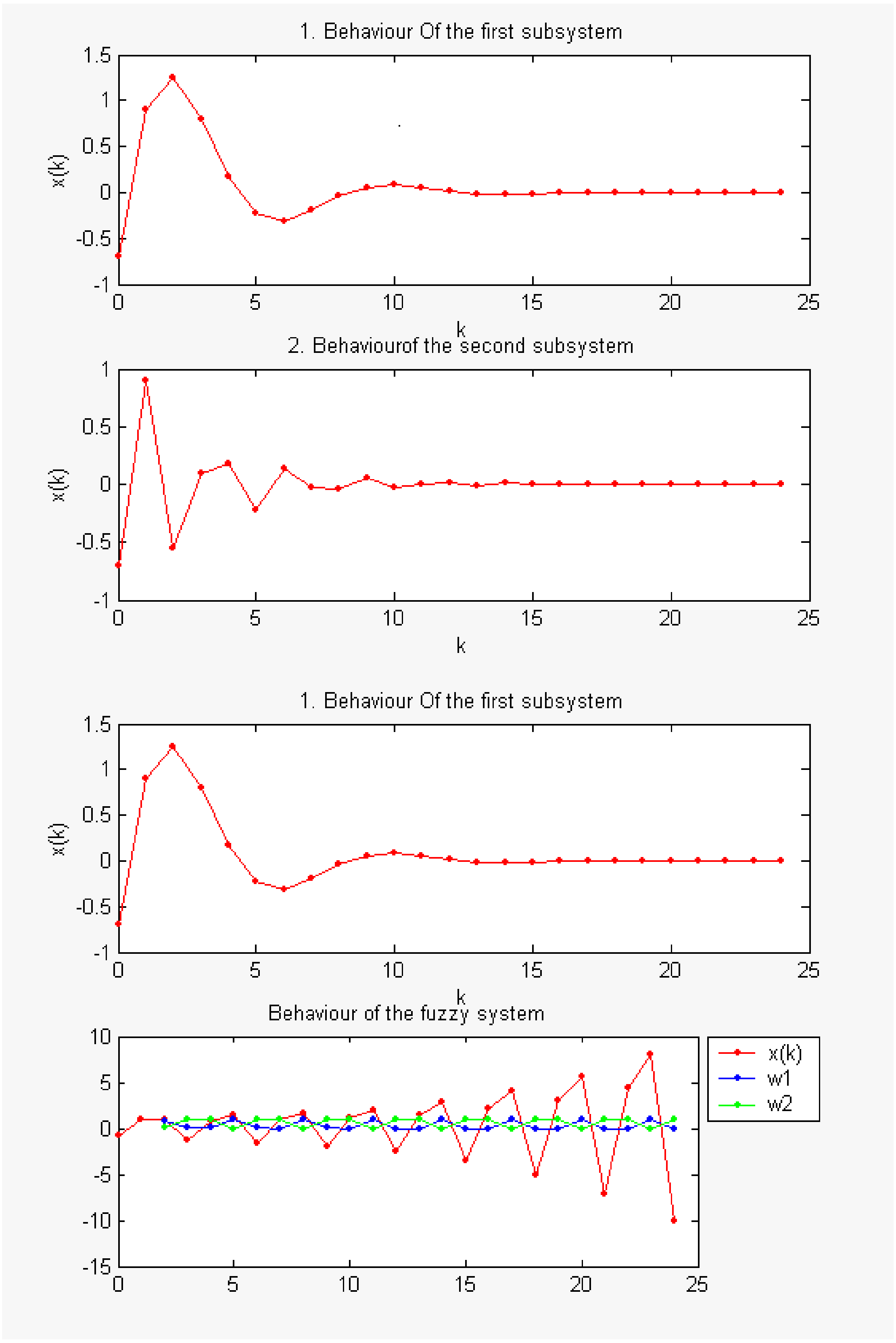
Program 2.1 : To plot the behavior of the Subsystems

And the overall fuzzy system

```
% A mat lab program written by Siba Brata Panda
 x1(1)=-0.7;
x1(2)=0.9;
x2(1)=-0.7;
x2(2)=0.9;
x(1)=-0.7;
x(2)=0.9;
i=1;
k1=2;
while k1~=25
x1(k1+1)=x1(k1)-0.5*x1(k1-1);
x2(k1+1)=-x2(k1)-0.5*x2(k1-1);
if     x(k1-1)>1
   wl(i)=0;
elseif x(k1-1)<-1
wl(i)=1;
else
wl(i)=(1-x(k1-1))/2;
wl(i)=1;
else
   wl(i)=(1-x(k1-1))/2;
end
if   x(k1-1)<-1
```

```
    w2(i)=0;
  elseif   x(k1-1)>1
    w2(i)=1;
  else
    w2(i)=(1+x(k1-1))/2;
  end
  x(k1+1)=(wl(i)*(x(k1)-0.5*x(k1-1))+w2(i)*(-x(k1)-0.5*x(k1-1)))/(wl(i)+w2(i));
  k1=k1+1;
  i=i+1;
  end
  k=0:24;
  subplot(2,1,1)
  plot(k,x1(k+1),'-r.')
  xlabel('k')
  ylabel('x(k)')
  title('1. Behaviour Of the first subsystem')
  subplot(2,1,2)
  plot(k,x2(k+1),'-r.')
  xlabel('k')
  ylabel('x(k)')
title('2. Behaviourof the second subsystem')
  ans=input('Enter 1 for the next plot.' )
  if   ans==1
    plot(k,x(k+1),'-r.')
    title ('Behaviour of the fuzzy system')
```

```
           hold on
          k=2:24;
          plot(k,(1-w2(k-1)),'-b.')
           plot(k,w2(k-1),'-g.')
          legend('x(k)','w1','w2',-1)
        hold off
else
          exit
      end
      end
```



Figures 2.1, 2.2 and 2.3

The linear systems are stable since A^1 and A^2 are stable matrices. However, the fuzzy system, which consists of the linear systems is unstable as shown, where $w^1$and w^2 denote the weights of L^1 and L^2, respectively.

Obviously, in this example, there does not exist a common P since the fuzzy ststem is unstable. Next, a necessary condition for ensuring the existence of a common P is given.

2.6 Necessary Condition for the Existence of a Common Positive Definite Matrix

Theorem 2.4:

Assume that A_i is a stable and nonsingular matrix for $I = 1,2...,l$. A_iA_j is a stable matrix for $I, j = 1,2 ..., l$ if there exists a common positive definite matrix P such that.

$A_i^TPA_i - P < 0$

(The above expression means that $A_i^TPA_i–P$ is negative definite.)

Proof:

$A_i^TPA_i - P < 0$

or $\quad A_i^TPA_i < P$

Also $\quad A_j^TPA_j - P < 0 \qquad$ (a)

or $\quad (A_j^{-1})^T(A_j^T PA_j -P) A_j^{-1}< 0$

or $\quad (A_j^T)^{-1}(A_j^T PA_j^{-1} - (A_j^{-1})^TPA_j^{-1}< 0$

or $\quad P\ 1\ (A_j^{\ -1})^TPA_j^{-1} < 0$

or $\quad P < (A_j^{-1})TPA_j^{-1} \qquad$ (b)

From (a) & (b)

$A_i^TPA_j < (A_j^{-l})^TPA_j^{-1}$

$A_i^TPA_j - (A_j)^TPA_j^{-1} < 0$

or $\quad A_j(A_i^TPA_i - (A_j^{-l})^TPA_j^{-1})Aj <0$

or $\quad A_j^T\ A_i\ PA_iAj -P < 0$

or $\quad (A_iAj)^TP(A_iA_j) -P < 0$

In the example given, eigenvalues of A_l are {0.5::t 0.5i}, eigenvalues of A_2 {-0.5 ± 0.5i}, and eigenvalues of A_1A_2 are {-0.134, -1.866}.Obviously, A_l and A_2 are stable as their eigenvalues lie within the unit circle; but A_1A_2 is unstable, as one of its eigenvalues is outside the unit circle. From theorem 2.4, it can be inferred that there does not exist a common P in this example.

2.7 Conclusion

In general, existence of a common positive definite matrix satisfying Lyapunov equation for all the. subsystem matrices shows that the overall system is stable. Even if all the subsystem matrices are stable, the overall system may be unstable. If the product of any two subsystem matrices is unstable, no common positive definite matrix exists. However, such a condition does not prove the system to be unstable.

Chapter 3

Stability Test Method-2

(Negative Bounds Approach)

3.1 Introduction

Stability test method given in chapter 2 fails to recognize a stable system in situations. Even if one or more subsystem matrices are unstable , the overall system may stable. A common positive definite matrix can not exist in such case and there fore the method discussed in this chapter 2 fails completely . A new stability test method is discussed in this chapter, which again gives a sufficient condition of stability, but which works even if some of subsystem matrices are un stable.

3.2 Stability of Fuzzy Systems in the Sense of Lyapunov: Negative Bounds Approach

Let P be the solution of the following Lyapunov equation for a positive definite and symmetric matrix Q:

$$A_0{}^T P A_0 - P = -Q \tag{3.9}$$

and let

$$D_i = (A_0 + \delta A_i)^T P (A_0 + \delta A_i) - P \tag{3.10}$$

from eq.(3.9), D_i becomes

$$D_i = \delta A_i{}^T P \delta A_i + \delta A_i{}^T P \delta A_0 + \delta A_0{}^T P \delta A_i - Q \tag{3.11}$$

We define $x \in \mathrm{Supp}(L_i)$ to mean that the state x is in the supports of the premise part fuzzy sets of the *i*th rule. That is

$$x_j \in \mathrm{Supp}(L_{ij}),\ j=1,2,\ldots,n$$

Theorem 3.1:

The model in eq.(3.2) with u = 0 is stable if for some stable A0, there exists a positive definite matrix Q such that for any rule of i, $x^T D_i x \le 0$, for all $x \in \{x | x \in \mathrm{Supp}(Li)\}$, where the equality holds only when x=0.

Proof:

For a positive definite Q, the solution P of the Lyapunov equation is also positive definite when A_0 is stable. Thus, $V(k) = x(k)^T P x(k)$ is always positive, and

$$\Delta V(k) = V(k+1) - V(k)$$

$$= x(k+1)^T P x(k+1) - x(k)^T P x(k)$$

$$= x(k)^T (A_0 + \Sigma\alpha_i \delta A_i)^T P (A_0 + \Sigma\alpha_i \delta A_i) x(k) - x(k)^T P x(k)$$

$$= x(k)^T [(A_0 + \Sigma\alpha_i \delta A_i)^T P (A_0 + \Sigma\alpha_i \delta A_i) - P] x(k)$$

$$= x(k)[(A_0{}^T + \Sigma\alpha_i \delta A_i{}^T) P (A_0 + \Sigma\alpha_i \delta A_i) - P] x(k)$$

$$= x(k)^T [A_0{}^T P A_0 + \Sigma\alpha_i \delta A_i{}^T P A_0 + A_0{}^T P \Sigma\alpha_i \delta A_i$$

$$+ \Sigma\alpha_i \delta A_i{}^T P \Sigma\alpha_i \delta A_i - P] x(k) \qquad (3.12)$$

Using the property $2|ab| \le a^2 + b^2$

$$\sum_{i=1}^{m} x(k)^T \alpha_i \delta A_i{}^T P \sum_{i=1}^{m} \alpha_i \delta A_i\, x(k) = \sum_{i=1}^{m}\sum_{j=1}^{m} x(k)^T \alpha_i \delta A_i{}^T P \alpha_j \delta A_j\, x(k)$$

$$= \sum_{i=1}^{m} x(k)^T \alpha_i \alpha_i \delta A_i{}^T P \delta A_i\, x(k) + \sum_{i=1}^{m}\sum_{j\neq 1}^{m} x(k)^T \alpha_i \alpha_j \delta A_i{}^T P \delta A_j\, x(k)$$

$$\le \sum_{i=1}^{m} x(k)^T \alpha_i \alpha_i \delta A_i{}^T P \delta A_i\, x(k) + (1/2)\sum_{i=1}^{m}\sum_{j\neq 1}^{m} x(k)^T \alpha_i \alpha_j (\delta A_i{}^T P \delta A_i$$

$$+ \delta A_j{}^T P \delta A_j)\, x(k)$$

$$= (1/2)\sum_{i=1}^{m} x(k)^T \sum_{j=1}^{m} \alpha_i \alpha_j (\delta A_i{}^T P \delta A_i + \delta A_j{}^T P \delta A_j) x(k)$$

$$= \sum_{i=1}^{m} \alpha_i x(k)^T \delta A_i{}^T P \delta A_i x(k)$$

(Using $\sum_{i=1}^{m}\alpha_i = 1$)

By applying this inequality to eq.(3.12)

$$\Delta V(k) \le \sum_{i=1}^{m}\alpha_i x(k)^T(\delta A_i{}^T P\,\delta A_0 + A_0{}^T P\delta A_i + \delta A_i{}^T PA_i - Q)x(k)$$

$$= \sum_{i=1}^{m}\alpha_i x(k)^T D_i x(k)$$

The α_i's are zero for all $i \in \{j \mid x \notin \mathrm{Supp}(L_j)\}$.thus for each rule, if $x^T D_i x \le 0$ for all $x \in \{x \mid x \in \mathrm{Supp}(L_i)\}$, then $\Delta V(k) \le 0$. this means that V(k) is a Lyapunov function. Therefore the system is stable at the origin.

This theorem gives only a sufficient condition of stability. Even if the system is stable, sometimes there does not exist such a Q. it is due to three reasons:

(1) $x^T Px$ is used as a Lyapunov function.

(2) The inequality $2|ab| \le a^2 + b^2$ is used during the proof, and

(3) α_i's are treated as uncertain for $x \in \mathrm{Supp}(L_i)$.

First, $x^T Px$ is a necessary and sufficient condition only for linear systems. In case of a linear system, it is only sufficient. Thus, if a nonlinear system is proved as stable using $x^T Px$, then the system is stable. Second, we use the inequality $2|ab| \le a^2 + b^2$ during the proofwhose equality holds only when $|a| = |b|$. Third, in this proof we assume that α_i's are uncertain parameters. Strictly speaking, the third reason does not describe the characteristics of this theorem well. Here, we assume that they are uncertain only when the corresponding rule affects the results. In other words, they are uncertain only when $x \in \mathrm{Supp}(L_i)$. in case of the model having uncertainty problems, α_i's are entirely uncertain parameters regardless $x \in \mathrm{Supp}(L_i)$ or not. In that case the condition $x^T D_i x \le 0$ is equivalent to the negative definiteness of D_i. if all D_i's are negative definite, then the system is stable.

The system is found to be stable if $x^T D_i x \leq 0$ for $x \in$ Supp(L_i)., even when D_i is not negative definite. Fuzzy model stability can be determined more adequately by theorem 3.1 than by the negative definiteness od D_i's. as mentioned, the method suggested by Tanaka and Sugeno [Tanaka'92] assumes that the parameters are entirely uncertain. Thus, their stability test sometimes does not work correctly. The example illustrates one such case.

The use of $\sum \alpha_i(x) A_i$ when $x \equiv 0$ is recommended for an A_0, since we are primarily interested in stability at the origin. For the system to be stable at the origin, A_0 should be a stable matrix at the origin.

In order to apply the theorem we need to know the maximum bound of $x^T D_i x$ for all $x \in$ Supp(L_i). it is not difficult to find the bounds since all D_i's are symmetric. The lower and upper bounds of $x^T D_i x$ can readily be calculated after diagonalizing D_i with its eigen vectors, though they are not tight bounds.

Example 3.1:

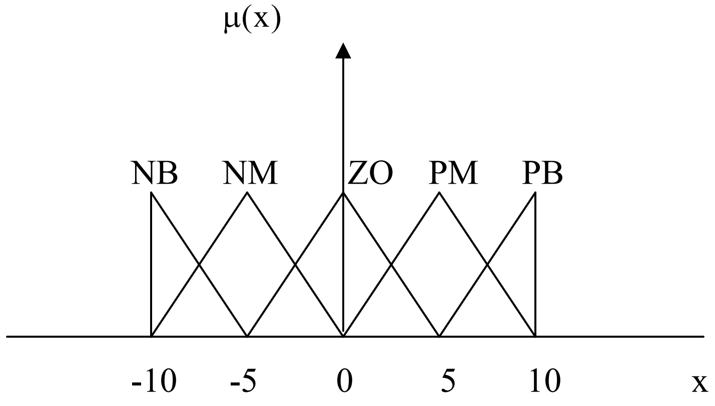


Figure 3.1: Membership Functions

Consider a system modeled by the following rules:

Rule 1: IF x_1 is ZO and x_2 is ZO, THEN $x(k+1) = A_1 x(k)$

Rule 2: IF x_1 is NM, THEN $x(k+1) = A_1 x(k)$

Rule 3: IF x_1 is PM, THEN $x(k+1) = A_1 x(k)$

Rule 4: IF x_1 is ZO and x_2 is NB, THEN $x(k+1) = A_2x(k)$

Rule 5: IF x_1 is ZO and x_2 is PB, THEN $x(k+1) = A_2x(k)$

Rule 6: IF x_1 is NB and x_2 is ZO, THEN $x(k+1) = A_3x(k)$

Rule 7: IF x_1 is PB and x_2 is ZO, THEN $x(k+1) = A_3x(k)$

Let the subsystem matrices be

$$A_1 = \begin{bmatrix} 0.95 & -0.1 \\ 0.13 & -0.9 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} 1.1 & -0.1 \\ 0.11 & -0.6 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0.6 & -0.12 \\ 0.15 & -1.2 \end{bmatrix}$$

The eigenvalues of A_1 are {0.9429, -0.8929},

The eigenvalues of A_2 are {1.0935, -0.5935},

The eigenvalues of A_3 are {0.5899, -1.1899}.

The subsystem matrix at the origin, A_1 is stable, while A_2 and A_3 are unstable matrices. So, there is no positive definite matrix P that makes $A_i^TPA_i$-P; i = 2,3 negative definite. Thus this system seems not to be stable. We use theorem 3.1 for this model.

As A_1 is stable, let $A_0 = A_1$.

Let us also assume that Q = I, then the solution of the equation

$A_0^TPA_0 – P = -Q$ gives

$$P = \begin{bmatrix} 9.1509 & -0.7790 \\ -0.7790 & 5.0068 \end{bmatrix}$$

Now $D_i = A_i^T P A_i - P$ then

$$D_1 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 1.7937 & -0.0354 \\ -0.0354 & -3.2063 \end{bmatrix}$$

$$D_3 = \begin{bmatrix} -5.8841 & -0.2062 \\ -0.2062 & 2.1104 \end{bmatrix}$$

After diagonalizing with the corresaponding eigenvectors, bounds of $\Delta V(k)$ comes out to be

[-200, 0] for rules 1, 2, 3;

[-322.9, -32.9] for rules 4, 5; and

[-603.8, -81.2] for rules 6, 7.

The procedure adopted for calculating the bounds is given as follows:

Let us consider rules 4 & 5. the eigenvectors of D_2 give its diagonalizing matrix, say, M. we get the following values:

$$M = \begin{bmatrix} 0.0071 & -1.0000 \\ -1.0000 & 0.0071 \end{bmatrix}$$

$$M^{-1} = \begin{bmatrix} 0.0071 & -1.0000 \\ -1.0000 & 0.0071 \end{bmatrix}$$

$$M^T D_2 M = \begin{bmatrix} -3.2066 & 0 \\ 0 & 1.7940 \end{bmatrix}$$

Let $y = M^{-1}x$

$$= M^{-1}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Therefore,

$$x^TD_2x = (My)^TD_2(My)$$

$$= y^T(M^TD_2M)y$$

$$= -3.2066(0.0071x_1+x_2)^2+1.7940(-x_1+0.0071x_2)^2$$

For rules 4 & 5,

x_1 ranges from -5 to 5,

x_2 ranges from -10 to -5 and from 5 to 10.

For finding the lower bound we put $x_1 = 5$, and $x_2 = 10$, in the first term, and $x_1 = 0.0355$, $x_2 = 5$, in the second term 9as these values make the second term zero) of the expression for x^TD_2x. Whereby, the value of the lower bound is found to be -322.9.

For finding the upper bound: we put $x_1 = -5$, $x_2 = 5$, in the first term, and $x_1 = -5$,

$x_2 = 10$, in the second term of the expression for x^TD_2x. Which gives the value of the upper bound as -32.9.

obviously, the above procedure can only give loose bounds.

**Program 3.1: To find and diagonalize the D matrices
For any number of subsystem matrices**

```
% A matlab program written by Siba Brata Panda
disp('the order of all the subsystem matrices should be 2.')
n=input('Enterthe number of subsystem matrices.');
if n<=0
   error('Improper input.')
end
A(:,:,1) =input('Enter the first subsystem matrix.');
for i=2:n
A(:,:,i)=input('Enter the next subsystem matrix. ');
end
disp('The respective eigenvalues are:')
fori=1:n
disp(eig(A(:,:,i)))

A0=input('Enter a stable matrix,A0. ');
Q=[1 0;0 1];
P=[0 0;0 0];
for i=0:1000
  P=P+(A0')^i*Q*(A0^i);
end
disp('P equals')
disp(P)
disp('forQ=')
disp(Q)
fori=1:n
 disp('The D matrices corresponding to the subsystem matrices are respectively:')
for i=1:n
  end
 for i=1:n
   disp('**************************************************')
   no=sprintf('For D matrix no. %g,',i);
   disp(no)
   disp('the diagonalizing matrix, M is')

   disp('the inverse of the diagonalizing matrix is')

   disp('the matrix after diagonalization')

end
   disp('************************************************')
```

Sample Run of Program 3.1

```
The order of all the subsystem matrices should be 2.
Enterthe number of subsystem matrices.3
Enter the first subsystem matrix.[0.95 -0.1;013 -0.9]
Enter the next subsystem matrix. [1.1 -0.1;0.11 -0.6]
Enter the next subsystem matrix. [0.6 -0.12;0.15 -1.2]
The respective eigenvalues are:

fori =
    1    2    3
   0.5899
  -1.1899

Enter a stable matrix,A0. [0.95 -0.1;013 -0.9]
P equals
  213.3198  -14.9730
  -14.9730    2.3055

forQ=
    1    0
    0    1

fori =

    1    2    3

The D matrices corresponding to the subsystem matrices are respectively:
***************************************************
For D matrix no. 1,
the diagonalizing matrix, M is
the inverse of the diagonalizing matrix is
the matrix after diagonalization
***************************************************
For D matrix no. 2,
the diagonalizing matrix, M is
the inverse of the diagonalizing matrix is
the matrix after diagonalization
***************************************************
For D matrix no. 3,
the diagonalizing matrix, M is
the inverse of the diagonalizing matrix is
the matrix after diagonalization
*************************************************
```

3.3 Identification of Stable System as Stable:

Some Q's give all values of $x^T D_i x$ less than zero; and for other Q's all values are not less than zero. Theorem 3.1 states that if there exists atleast one Q which satisfies the stability condition, the system is stable. Thus we need an algorithm with which we can identify such a Q. unfortunately, it is difficult to form a generalized algorithm. However if we can make the eigenvalues of D_i more negative (i.e. if we can make the maximum eigenvalue of D_i smaller) the probability that $x^T D_i x \leq 0$ is increased. Thus if we can adjust Q such that the maximum eigenvalue of the matrix D_i is smaller, the probability that a stable system is identified as stable is increased. Using a gradient based algorithm, we can systematically decrease the maximum eigenvalue of D_i. let function J_i be given by

$$J_i = \lambda_M(D_i) \tag{14}$$

This makes J_i a function of Q, where $\lambda_M(.)$ represents the maximum eigenvalue. Q being a positive definite matrix, it can be decomposed into $Q = L^T L$ where L is a full rank matrix. J_i can be minimized by an iterative adjustment of L using the gradient based algorithm. The gradient can be described by the following theorem. Note that subscript i has been omitted for notational convenience.

Theorem 3.2

Let J be defined as in eq.(14). Then

$$\partial J/\partial L = 2L(W - g_M g_M^T), \tag{3.15}$$

where W satisfies

$$A_0 W A_0^T - W = -(A_0+\delta A)\, g_M g_M^T (A_0+\delta A)^T + A_0\, g_M g_M^T A_0^T \tag{3.16}$$

And g_M is an eigenvector corresponding to $\lambda_M(D)$.

Proof:

D may be diagonalized using eigenvectors as it is a symmetric matrix. In other words, $G^T DG$ is a diagonal matrix, where G is an orthogonal matrix whose columns are eigenvectors. That is, $G = [g_1 \dots g_2]$ where g_i is the *i*th eigenvector.

Then ΔJ

resulting from ΔQ is

$\Delta J = g_M{}^T \Delta g_M$

$= Tr\{g_M{}^T(\delta A^T \Delta P A_0 + A_0{}^T \Delta P \delta A + \delta A^T \Delta P \delta A - \Delta Q)\, g_M\}$ (3.17)

where g_M is an eigenvector corresponding to $\lambda_M(D)$, and ΔD and ΔP are the increment resulting from ΔQ. Tr{.} means matrix trace. We know that

$Tr(AB) = Tr(BA) = Tr(B^T A^T)$ (3.18)

and

$Tr(A+B) = Tr(A) + Tr(B).$ (3.19)

Using eqs.(3.18) and (3.19),

$Tr(g_M{}^T \Delta Q g_M) = Tr(g_M g_M{}^T \Delta Q)$

$= Tr(g_M g_M{}^T(\Delta L^T L + L^T \Delta L))$

$= 2Tr(g_M g_M{}^T\, L^T \Delta L)$ (3.20)

The solution of the lyapunov equation (3.9) is given by

$$P = \sum_{i=0}^{\infty} (A_0^T)^i Q A_0^i\ , \tag{3.21}$$

and ΔP is given by,

$$\Delta P = \sum_{i-0}^{\infty} (A_0^T)^i (\Delta L^T L + L^T \Delta L)(A_0)^i\ . \tag{3.22}$$

Using eqs.(3.20) and (3.22), eq.(3.17) becomes

$\Delta J = 2Tr\{\big(\sum A_0{}^i \big(\delta A\ \ g_M\, g_M{}^T\, A_0{}^T + A_0\, g_M\, g_M{}^T\, \delta A^T$

$+\delta A\, g_M\, g_M{}^T\, \delta A^T\big)\big(\, A_0{}^T)^i - g_M\, g_M{}^T\,\big) L^T \Delta L\}$

Program 3.2: To decrease the maximum eigenvalues of Di’s by updating Q

```
% A matlab program written by Siba Brata Panda
disp('The order of all the subsystem matrices should be 2.')
n=input('Enter the number of subsystem matrices. ');
if n<=0
   error('Improper input.')
end
A(:,:,1)=input('Enter the first subsystem matrix.');
for i=2:n
    A(:,:,i)=input('Enter the next subsystem matrix.');
end
disp('The respective eigen values are:')
for i=1:n
   disp(eig(A(:,:,i)))
end
A0=input('Enter a stable m,atrix,A0.');
L=[1 0;0 1];
answer=1;
iteration=0;
while answer==1
Q=L'*L
P=[0 0;0 0]
 fori=0:100
   P=(P+(A0')^i)*Q*(A0)^i;
end
disp('P equals')
disp(p)
for i=1:n
   D(:,:,i)=A(:,:,i)'*P*A(:,:,i)-P;
end
disp('The D matrices corresponding to the subsystem matrices are respectively:')
for i=1:n
   disp(D(:,:,i))
end
disp('The respective eigenvalues are:')
for i=1:n
   disp(eig(D(:,:,i)))
end
for i=1:n
   [M(:,:,i),eig_D(:,:,i)]=eig(D(:,:,i));
end
answer=input('Does any eigenvalues need to be update? Enter 1 for yes. ');
```

```
if answer==1
n1=input('The eigen value of which D matrix is to be updated? Enter no. ');
delA=A(:,:,n1)-A0;
M1=M(:,:,n1);
n2=input('Which eigenvalue is to be updated? Enter no. ');
gm=M1(:,n2)
W=[0 0;0 0];
for i=0:1000

W=W+(A0^i)*(delA*gm*gm'*A0'+A0*gm*gm'*delA'+delA*gm*gm'*delA')*(A0'
)^i;
end
disp('W equals')
disp(W)
L=L-2*L*(W-gm*gm')
end
iteration=iteration+1;
iteration1=sprintf('This was iteration no. %g',iteration);
disp(iteration1)
```

Sample Run of Program 3.2

```
The order of all the subsystem matrices should be 2.

Enter the number of subsystem matrices. 2

Enter the first subsystem matrix.[0.9997 0.009899;-0.05939 0.9799]

Enter the next subsystem matrix.[0.9998 0.009949;-0.02985 0.9899]

The respective eigen values are:

 0.9898 + 0.0221i

 0.9898 - 0.0221i


 0.9949 + 0.0165i

 0.9949 - 0.0165i


Enter a stable m,atrix,A0.[0.9998 0.0099;-0.0446 0.9849]
```

3.4 Conclusion

It is found tahat negative definiteness of $X^T Dix$ for all the rules is a sufficient condition of stability for the fuzzy systems discussed in this chapter. Maximum bound of $X^T Dix$ can be found easily as Di's are symmetric . Some times the method fails to identify a stable system as stable. Using a gradient based algorithm , the maximum eigenvalue of Di is systematically decreased, which increases the probability of X^T Dix being less than zero and thereby increases the probability that a stable system is identified as stable.

Chapter 4

Stabilizing Control

Stability is the ultimate goal.

4.1 Introduction

It is shown in the first sample run of program 4.2 that the fuzzy free system taken there is highly unstable. Linear feedback control can be used to stabilize the system. A particular value of K, which is not necessarily unique , can make the system stable as shown in second sample run of program 4.2. An algorithm is given to determine this particular value of K.

4.2 Stabilizing using Linear Feedback Control

Since Ao in the theorem 3.1 has to be substituted with Ao+boK as shown in equation 4.1 , we first find a range of K for which Ao+boK is stable; then we check the eigen values of Dc matrices given by equation 4.2 . Let us consider the linear feedback control u=Kx. Then the subsystem is described by

$$X(K+1) = (A0 + boK + \sum \acute{\alpha} (\delta Ai + \delta biK))X(K) .$$

After substituting Ao and δAi in the theorem 3.1 with Ao+boK and δAo+δbiK respectively , we can determine the stability of the feed back system. The model in eq.(3.2) with control

u = Kx is stable if for given rule i, $X^T\ Di^c\ x \leq 0$ for all

$$Di^c = (\delta Ai + \delta biK)^T P (\delta Ai + \delta biK) + (\delta Ai + \delta biK)^T P(A_0 + boK)$$
$$+ (Ao+boK)^T P (\delta Ai + \delta biK) - Q$$

and P is the solution of Lyapunov equation :

$$(Ao+boK)^T P (Ao + boK) - P = -Q$$

To find the stabilizing control, we have to find a control gain K that satisfies represents $X^T\ Di^c\ x \leq 0$. Since Di^c represents Di in eq.(3.10), we can increase the probability that $X^T\ Di^c\ x \leq 0$ if we can decrease the maximum eigenvalue of Di^c

Program 4.1 : To find and plot the eigenvalues of Di^c Matrices for a range of K, taking aFuzzy system having 9 rules

```
n=9;
disp('Enter the subsystemmatrix for rule 1,i.e.,')
A(:,:,1)=input('when x1 is low and x2is low.(2x2)');
eigenvalues=eig(A(:,:,1))
b(:,1)=input('Enter the cooresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 2, i.e.,')
A(:,:,2)=input('when x1 is LOW and x2 is MED.(2x2)');
eigenvalues=eig(A(:,:,2))
b(:,2)=input('Enter the cooresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 3, i.e.,')
A(:,:,3)=input('when x1 is LOW and x2 is HIGH.(2x2)');
eigenvalues=eig(A(:,:,3))
b(:,3)=input('Enter the cooresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 4, i.e.,')
A(:,:,4)=input('when x1 is MED and x2 is LOW.(2x2)');
eigenvalues=eig(A(:,:,4))
b(:,4)=input('Enter the cooresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 5, i.e.,')
A(:,:,5)=input('when x1 is MED and x2 is MED.(2x2)');
eigenvalues=eig(A(:,:,5))
b(:,5)=input('Enter the cooresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 6, i.e.,')
A(:,:,6)=input('when x1 is MED and x2 is HIGH.(2x2)');
eigenvalues=eig(A(:,:,6))
b(:,6)=input('Enter the cooresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 7, i.e.,')
A(:,:,7)=input('when x1 is HIGH and x2 is LOW.(2x2)');
eigenvalues=eig(A(:,:,7))
b(:,7)=input('Enter the cooresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 8, i.e.,')
A(:,:,8)=input('when x1 is HIGH and x2 is MED.(2x2)');
eigenvalues=eig(A(:,:,8))
b(:,8)=input('Enter the cooresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 9, i.e.,')
A(:,:,9)=input ('when x1 is HIGH and x2 is HIGH. (2x2) ' );
eigenvalues=eig(A(:,:,9))
b(:,9)=input('Enter the corresponding input matrix.(2x1)
');
disp('A0+b0*k should be stable.')
A0=input('Enter A0.(2x2 )');
b0=input('Enter b0. (2x1)');
ans=1;
while ans==1
disp('Suggest a range for k.')
high1=input('Enter the higher limit for k(1).');
low1=input('Enter the lower limit for k(1).');
high2=input('Enter the higher limit for k(2).');
```

```
low2=input('Enter the lower limit for k(2).');
incr=input ('Enter the increment.');
disp('The eigenvalues of A0+b0*k and the corresponding k''s
are listed below.')
for p=low2:incr:high2
    forq=low1:incr:high1
    k(1)=q;
    k(2)=p;
    res=A0+b0*k;
    disp(eig(res))
    disp('For k=')
    disp(k)
    disp('-----------')
end
end
ans=input('Do you want to change the range of k? Enter 1
for yes .');
end
ans=input('Should the program be terminated? Enter 1 for
yes.');
if ans1==1
    break
end
disp('the eigenvalues of the Dc matrices ofand the
cooresponding k''s are listed below.')
ii=0;
for p=low2:incr:high2
    for q=low1:incr:high1
        k(1)=q;
        k(2)=p;
        ii=ii+1;
        k1(ii)=k(1);
        k2(ii)=k(2);
        for i=1:n
            delA(:,:,i)-A0;
            delb(:,i)-b0;
        end
        Q=[1 0;0 1];
        p=[0 0;0 0];
        for i=0:1000
            p=p+((A0+b0*k)')^i*Q*(A0+bo*k)^i;
        end
  for i=1:n

     Dc(:,:,i)=(delA(:,:,i)+delb(:,i)*k)'*p*(delA(:,:,i)+de
     lb(:,I)*k)+...
                (deiA(:,:,i)+delb(:,i)*k)'*p*(A0+b0*k)+...
                (A0+b0*k)'*p*(delA(:,:,i)+delb(:,i)*k)-q;
        end
        for i=1:n
            disp(eig(dc(:,:,i)))
        end
        disp('For k=')
        disp(k)
        disp('-----')
        for i=1:n
            e(:,i)=eig(Dc(:,:,i));
            eig1(ii,i)=e(1,i);
            eig2(ii,i)=e(2,i);
        end
    end
end
total=((high1-low1)/incr+1)*((high2-low2)/incr+1);
ans2=1;
while ans2==1
    ans2=input('Entering 1will plot the eigenvalues ofa Dc
matrix against k.');
    if ans2~=1
        break
    end
    n1=input ('which Dc matrix do you select.Enter no.');
    if low<low2
        lowlim=low1;
    else
        lowlim=low2;
    end
    if high1>high2
        highlim=high1;
    else
        highlim=high2;
    end
    ti = lowlim:incr*0.1:highlim;
    [XI,YI] = meshgrid(ti,ti);
    ii=i:total==0.5
    ZI1 = griddata (k1(ii),k2(ii),eig1(ii,n1),XI,YI);
    ZI2 = griddata (k1(ii),k2(ii),eig2(ii,n1),XI,YI);
    subplot(2,1,1);
    mesh(XI,YI,ZT1),hold

     plot3(k1(ii),k2(ii),eig1(ii,n1),':wo','MarkerFaceColor
     ','k','MarkerSize',8)
    axis([low1 high1 low2 high2])
    xlabel('k(1)','fontsize',8)
    ylabel('k(2)','fontsize',8)
    zlabel('First Eigenvalue','fontsize',8)
    title('Eigenvalues of the Dc matrix against k')
    colorbar
    hold off
    subplot(2,1,2);
    mesh(XI,YI,ZT2),hold
```

```
 plot3(k1(ii),k2(ii),eig2(ii,n1),'two','MarkFaceColor',
 'k','markersize',8)
axis([low1 high1 low2 high2])
xlabel('k(1)','Fontsize',8)
ylabel('k(2)','Fontsize',8)
zlabel('Second Eigenvalue','fontsize',8)
colorbar
hold off
end
```

Sample Run of Program 4.1

```
>> Enter the subsystemmatrix for rule 1,i.e.,
when x1 is low and x2is low.(2x2) [1.2 0.75;-4.8 -1.3]

eigenvalues =

  -0.0500 + 1.4274i
  -0.0500 - 1.4274i

Enter the cooresponding input matrix.(2x1) [-1;3]
Enter the subsystem matrix for rule 2, i.e.,
when x1 is LOW and x2 is MED.(2x2)[-1.8 -1.125;5.6 2]


eigenvalues =

   0.1000 + 1.6401i
   0.1000 - 1.6401i

Enter the cooresponding input matrix.(2x1)[1.5;-3.5]
Enter the subsystem matrix for rule 3, i.e.,
when x1 is LOW and x2 is HIGH.(2x2) [-4 -2.25;-6.4 -2.4]

eigenvalues =

   -7.0781
    0.6781

Enter the cooresponding input matrix.(2x1) [3;4]
Enter the subsystem matrix for rule 4, i.e.,
when x1 is MED and x2 is LOW.(2x2) [-0.8 -0.675;-2.88 -0.7]

eigenvalues =

   -2.1452
    0.6452

Enter the cooresponding input matrix.(2x1) [0.9;1.8]
Enter the subsystem matrix for rule 5, i.e.,
when x1 is MED and x2 is MED.(2x2)[7.4 3.75;-9.6 -3.8]
```

```
eigenvalues =

   1.8000 + 2.1541i
   1.8000 - 2.1541i

Enter the cooresponding input matrix.(2x1) [-5;6]
Enter the subsystem matrix for rule 6, i.e.,
when x1 is MED and x2 is HIGH.(2x2) [-2.5 -1.5;6.4 2.3]

eigenvalues =

  -0.1000 + 1.9596i
  -0.1000 - 1.9596i

Enter the cooresponding input matrix.(2x1) [2;-4]
Enter the subsystem matrix for rule 7, i.e.,
when x1 is HIGH and x2 is LOW.(2x2) [0.1 -0.375;2.4 0.5]

eigenvalues =


  0.3000 + 0.9274i
   0.3000 - 0.9274i

Enter the cooresponding input matrix.(2x1) [0.5;-1.5]
Enter the subsystem matrix for rule 8, i.e.,
when x1 is HIGH and x2 is MED.(2x2)[-2.5 -1.5;9.6 5]

eigenvalues =

   1.2500 + 0.5809i
   1.2500 - 0.5809i

Enter the cooresponding input matrix.(2x1) [2;-6]
Enter the subsystem matrix for rule 9, i.e.,
when x1 is HIGH and x2 is HIGH. (2x2) [7 3;-2 0.01]

eigenvalues =

    5.9980
    1.0120
```

```
Enter the corresponding input matrix.(2x1)  [-4;1.25]
A0+b0*k should be stable.
Enter A0.(2x2 )[0.1 -0.375;2.4 0.5]
Enter b0. (2x1)[0.5;-1.5]
Suggest a range for k.
Enter the higher limit for k(1).2
Enter the lower limit for k(1).1
Enter the higher limit for k(2).1
Enter the lower limit for k(2).0
Enter the increment.0.2
The eigenvalues of A0+b0*k and the corresponding k's are listed below.
   0.5500 + 0.5788i
   0.5500 - 0.5788i

For k=
     1     0

-----------
   0.6000 + 0.4637i
   0.6000 - 0.4637i

For k=
    1.2000         0

-----------
   0.6500 + 0.3000i
   0.6500 - 0.3000i

For k=
    1.4000         0

-----------
    0.9000
    0.5000




For k=
    1.6000         0

-----------

   1.1683
   0.3317



For k=
   1.8000         0

-----------
   1.3612
   0.2388

For k=
     2     0

-----------
  0.4000 + 0.4555i
  0.4000 - 0.4555i

For k=
   1.0000    0.2000

-----------
  0.4500 + 0.3202i
  0.4500 - 0.3202i

For k=
   1.2000    0.2000

-----------
   0.5866
   0.4134

For k=
   1.4000    0.2000

-----------
   0.9000
   0.2000

For k=
   1.6000    0.2000

-----------
   1.0924
   0.1076

For k=
   1.8000    0.2000

-----------
   1.2562
   0.0438


For k=
   2.0000    0.2000

-----------
  0.2500 + 0.1871i
  0.2500 - 0.1871i

   1.0000    0.4000

-----------
   0.5345
   0.0655

For k=
   1.2000    0.4000

-----------
   0.7373
  -0.0373

For k=
   1.4000    0.4000

-----------
   0.9000
  -0.1000

For k=
   1.6000    0.4000

-----------
   1.0458
  -0.1458

For k=
   1.8000    0.4000

-----------
   1.1819
  -0.1819
```

```
For k=
   2.0000    0.4000

-----------
   0.5272
  -0.3272

For k=
   1.0000    0.6000

-----------
   0.6574
  -0.3574


For k=
   1.2000    0.6000

-----------
   0.7809
  -0.3809

For k=
   1.4000    0.6000

-----------
   0.9000
  -0.4000

For k=
   1.6000    0.6000

-----------
   1.0159
  -0.4159

For k=
   1.8000    0.6000

-----------
   1.1294
  -0.4294

For k=
   2.0000    0.6000
```

```
-----------
    0.6171
   -0.7171

For k=
    1.0000    0.8000

-----------
    0.7106
   -0.7106

For k=
    1.2000    0.8000

-----------
    0.8050
   -0.7050

For k=
    1.4000    0.8000

-----------
    0.9000
   -0.7000

For k=
    1.6000    0.8000

-----------
    0.9956
   -0.6956

For k=
    1.8000    0.8000

-----------
    1.0916
   -0.6916

For k=
    2.0000    0.8000

-----------
    0.6675
   -1.0675

For k=
     1     1

-----------
    0.7430
   -1.0430

For k=
    1.2000    1.0000

-----------
    0.8206
   -1.0206

For k=
    1.4000    1.0000

-----------
    0.9000
   -1.0000

For k=
    1.6000    1.0000

-----------
    0.9811
   -0.9811

For k=
    1.8000    1.0000

-----------
    1.0637
   -0.9637

For k=
     2     1

-----------

Do you want to change the range of k? Enter 1 for yes .1
Suggest a range for k.
Enter the higher limit for k(1).1.6
Enter the lower limit for k(1).1.5
Enter the higher limit for k(2)..7
Enter the lower limit for k(2)..6
Enter the increment..05
```

```
The eigenvalues of A0+b0*k and the corresponding k's are listed below.
    0.8409
   -0.3909

For k=
    1.5000    0.6000

-----------
    0.8706
   -0.3956




For k=
    1.5500    0.6000

-----------
    0.9000
   -0.4000

For k=
    1.6000    0.6000

-----------
    0.8443
   -0.4693

For k=
    1.5000    0.6500

-----------
    0.8722
   -0.4722

For k=
    1.5500    0.6500

-----------
    0.9000
   -0.4750
```

```
For k=
   1.6000    0.6500

-----------
   0.8473
  -0.5473

For k=
   1.5000    0.7000

-----------
   0.8737
  -0.5487

For k=

  1.5500    0.7000




-----------
   0.9000
  -0.5500

For k=
   1.6000    0.7000

-----------
Do you want to change the range of k? Enter 1 for yes .0
Should the program be terminated? Enter 1 for yes.0
the eigenvalues of the Dc matrices and the cooresponding k's are listed below.
   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1
```

```
    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

For k=
    1.5000    0.7000

-----
    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1


  -1
    -1

    -1
    -1

    -1
    -1

For k=
    1.5500    0.7000
```

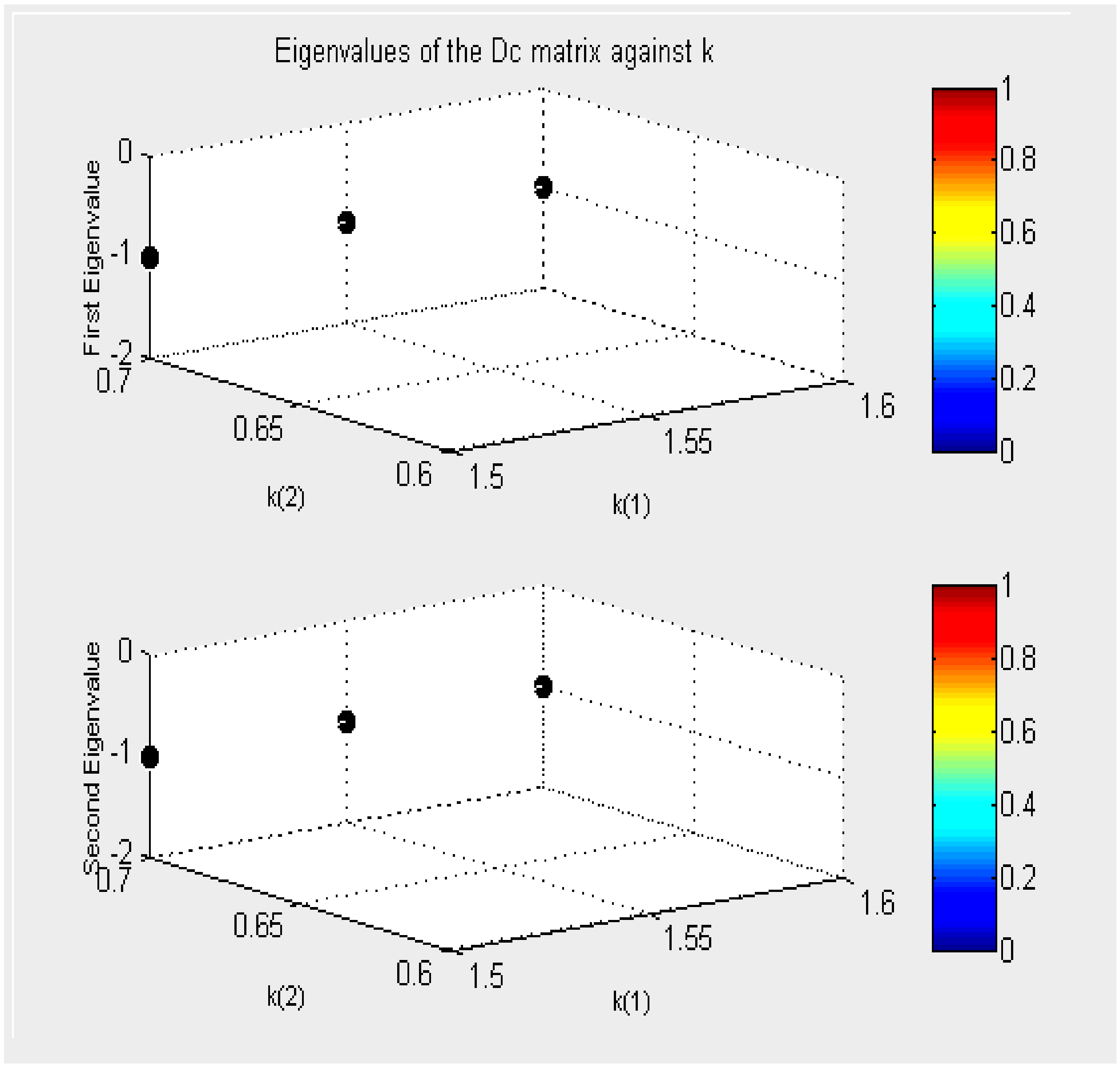
```
-----
   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

For k=
   1.6000    0.7000

-----
   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1
```

```
    -1


   -1

    -1
    -1

    -1
    -1

    -1
    -1

For k=
    1.5000    0.7000

-----
    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

For k=
    1.5500    0.7000

-----
    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1





   -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

For k=
    1.6000    0.7000

-----
    -1
    -1

    -1
    -1

    -1
    -1

    -1
```

```
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

For k=
    1.5000    0.7000

-----
    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1

    -1
    -1
```

```
For k=
   1.5500    0.7000

-----
   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

   -1
   -1

For k=
   1.6000    0.7000

-----
Entering 1 will plot the eigenvalues of a Dc matrix against k. 1
which Dc matrix Do you select. Enter no. 3

ii =

     1     2     3     4     5     6     7     8     9

Current plot held
Entering 1 will plot the eigenvalues of a Dc matrix against k. 0
>> 1
```

```
ans =

     1
```



Figures 4.1 and 4.2

Program 4.2 : To inspect and plot the behavior of a Fuzzy system having 9 rules, for a Given value K

```
x=input("Enter the initial state. (2 1)' );
low=input('Enter parameters for triangular LOW .(1 2)');
med=input('Enter parameters for triangular MED .(13)');
high=input('Enter parameters for triangular HIGH .(1 2)');
k=input("Enter the gain k.. (2 1)' );
iteration=input('Enter the no of iterations.');
disp('Enter the subsystem matrix for rule 1, i.e.,')
A(:,:,1)=input('when x1 is  LOWand x2 is LOW. (2x2)');
b(:,1)=input('Enter the corresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 2, i.e.,')
A(:,:,2)=input('when x1 is  LOW and x2 is MED. (2x2)');
b(:,2)=input('Enter the corresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 3, i.e.,')
A(:,:,3)=input('when x1 is  LOW and x2 is HIGH. (2x2)');
b(:,3)=input('Enter the corresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 4, i.e.,')
A(:,:,4)=input('when x1 is  MED and x2 isLOW. (2x2)');
b(:,4)=input('Enter the corresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 5, i.e.,')
A(:,:,5)=input('when x1 is  MED and x2 is MED. (2x2)');
b(:,5)=input('Enter the corresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 6, i.e.,')
A(:,:,6)=input('when x1 is  MED and x2 is HIGH. (2x2)');
b(:,6)=input('Enter the corresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 7, i.e.,')
A(:,:,7)=input('when x1 is  HIGH and x2 is LOW. (2x2)');
b(:,7)=input('Enter the corresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 8, i.e.,')
A(:,:,8)=input('when x1 is  HIGH and x2 is MED. (2x2)');
b(:,8)=input('Enter the corresponding input matrix.(2x1)');
disp('Enter the subsystem matrix for rule 9, i.e.,')
A(:,:,9)=input('when x1 is  HIGH and x2 is HIGH. (2x2)');
b(:,9)=input('Enter the corresponding input matrix.(2x1)');
ii=1;
x1(ii)=x(1);
x2(ii)=x(2);
```

```
k=0;
response=input('Entering 1 will display the state vector after each iteration. ');
while k<iteration
for j=1:2

if x(j)<=low(1)
  LOW(j)=1;
elseif x(j)>low(1)&x(j)<low(2)
  LOW(j)=(x(j)-low(2))/(low(1)-low(2));
else
  HIGH(j)=0
elseif x(j)>high(1)&x(j)<high(2)
  HIGH(j)=(x(j)-high(1))/(high(2)-high(1));
else
  HIGH(j)=1;
end
end
fs(1)=min(LOW(1),LOW(2));
fs(2)=min(LOW(1),MED(2));
fs(3)=min(LOW(1),HIGH(2));
fs(4)=min(MED(1),LOW(2));
fs(5)=min(MED(1),MED(2));
fs(6)=min(MED(1),HIGH(2));
fs(7)=min(HIGH(1),LOW(2));
fs(8)=min(HIGH(1),MED(2));
fs(9)=min(HIGH(1),HIGH(2));
  sum=[0;0];
for i=1:9
   fire(ii,i)=fs(i);
end
for i=1:9
  sum=sum+fs(i);
end
sumfs=0;
for i=1:9
  sumfs=sumfs+fs(i);
end
x=sum/sumfs;
k=k+1;
next_state=mat2str(x);
ii=ii+1;
```

```
x1(ii)=x(1);
x2(ii)=x(2);
state= sprintf(‘State vector after iteration % 3g is %50s’, ii-1,next_state);
disp(state)
end
end
disp(The final state is’)
disp(x)
disp(‘Entering 1 will display the firing strength bof various rule for’)
for ii=1:iterations
fi=sprint f(‘itration %3g:%9.2g%9.2g9.2g9.2g9.2g9.2g9.2g9.2g9.2g9.2g9.2g9.2g9.2g
9.2g’,ii,fire(ii,1)…..
 fire(ii,2) fire(ii,3) fire(ii,4) fire(ii,5) fire(ii,6) fire(ii,7) fire(ii,8) fire(ii,9));
disp(fi)
end
end
answer=input('Entering 1 will plot the trajectory for the second state variable. ');
if answer==1
   ii=1:1:iteration
   plot(ii,x2(ii),'r','linewidth',2)
   xlabel('Iterations','color','b')
   ylabel('Second state variable','color','m')
   title(['Plot for k=',mat2str(k)])
   grid on
end
answer=input('Entering 1 will plot the trajectory for the second state variable. ');
if answer==1
   ii=1:1:iteration
   plot(ii,x2(ii),'r','linewidth',2)
   xlabel('Iterations','color','b')
   ylabel('Second state variable','color','m')
   title(['Plot for k=',mat2str(k)])
   grid on
end
answer=input('Enter 1 for logarithmic plots. ');
if answer==1
answer=input('Entering 1 will plot the trajectory for the first state variable. ');
if answer==1
   ii=1:1:iteration
   semilogy(ii,abs(x1(ii)),'r','linewidth',2)
   xlabel('Iterations','color','b')
   ylabel('Absolute value of the second state variable on logarithmic scale.',...
      'color','m','fontsize',8)
```

```
title(['Plot for k=',mat2str(k)])
   grid on
end
answer=input('Entering 1 will plot the trajectory for the second state variable. ');
if answer==1
   ii=1:1:iteration
   semilogy(ii,abs(x2(ii)),'r','linewidth',2)
   xlabel('Iterations','color','b')
   ylabel('Absolute value of the second state variable on logarithmic scale.',...
      'color','m','fontsize',8)
title(['Plot for k=',mat2str(k)])
grid on
end
end
disp('The program ends here.')
```

First Sample Run of Program 4.2

```
Enter the initial state.(2x1)[-10;15]
Enter parameters for triangular LOW .(1x2) [-1 0]
Enter parameters for triangular MED .(1x3) [-1 0 1]
Enter parameters for triangular HIGH .(1x2) [0 1]
Enter the gain k.(1x2)[0 0]
Enter the no of iterations. 50
Enter the sub system matrix for rule 1, i.e.,
when x1 is  LOW and x2 is LOW. (2x2) [1.2 0.75;-4.8 -1.3]
Enter the corresponding input matrix.(2x1)[-1;3]
Enter the subsystem matrix for rule 2, i.e.,
when x1 is  LOW and x2 is MED. (2x2) [--1.8 -1.125;5.6 2]
Enter the corresponding input matrix.(2x1) [1.5;-3.5]
Enter the subsystem matrix for rule3, i.e.,
when x1 is  LOW and x2 is HIGH. (2x2)[-4 -2.25;-6.4 -2.4]
Enter the corresponding input matrix.(2x1)[3;4]
Enter the subsystem matrix for rule 4, i.e.,
when x1 is  MED and x2 is LOW (2x2)[-0.8 -0.675;-2.88 -0.7]
Enter the corresponding input matrix.(2x1)[0.9;1.8]
Enter the subsystem matrix for rule 5, i.e.,
when x1 is  MED and x2 is MED. (2x2)[7.4 3.75;-9.6 -3.8]
Enter the corresponding input matrix.(2x1)[-5;6]
Enter the subsystem matrix for rule 6, i.e.,
when x1 is  MED and x2 is HIGH. (2x2)[-2.5 -1.5;6.4 2.3]
Enter the corresponding input matrix.(2x1)[2;-4]
Enter the subsystem matrix for rule 7, i.e.,
when x1 is  HIGH and x2 is LOW. (2x2) [0.1 -0.375;2.4 0.5]
Enter the corresponding input matrix.(2x1)[0.5;-1.5]
Enter the subsystem matrix for rule 8, i.e.,
when x1 is  HIGH and x2 is MED. (2x2) [-2.5 -1.5;9.6 5]
Enter the corresponding input matrix.(2x1) [2;-6]
Enter the subsystem matrix for rule 9, i.e.,
when x1 is  HIGH and x2 is HIGH. (2x2)[7 3;-2 0.01]
Enter the corresponding input matrix.(2x1) [-4;1.25]
Entering 1 will display the state vector after each iteration. 1

MED =
    1

HIGH =

    0    1
```

```
MED =

     1     1


State vector after iteration   1 is                                   [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration   2 is                                   [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration   3 is                                   [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration   4 is                                   [1;1]

MED =

     1     1


MED =
    1     1

State vector after iteration   5 is                              [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration   6 is                              [1;1]

MED =

    1     1


MED =

    1     1



State vector after iteration   7 is                              [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration   8 is                              [1;1]

MED =

    1     1


MED =
    1     1



State vector after iteration   9 is                                        [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration  10 is                                        [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration  11 is                                        [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration  12 is                                        [1;1]

MED =

    1     1


MED =
    1     1

State vector after iteration  13 is                                   [1;1]

MED =

    1     1

MED =

    1     1

State vector after iteration  14 is                                   [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration  15 is                                   [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration  16 is                                   [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration  17 is                                   [1;1]
```

```
MED =

     1     1



MED =

     1     1

State vector after iteration  18 is                    [1;1]

MED =

     1     1

MED =

     1     1

State vector after iteration  19 is                    [1;1]

MED =

     1     1

MED =

     1     1

State vector after iteration  20 is                    [1;1]

MED =

     1     1

MED =

     1     1

State vector after iteration  21 is                    [1;1]
```

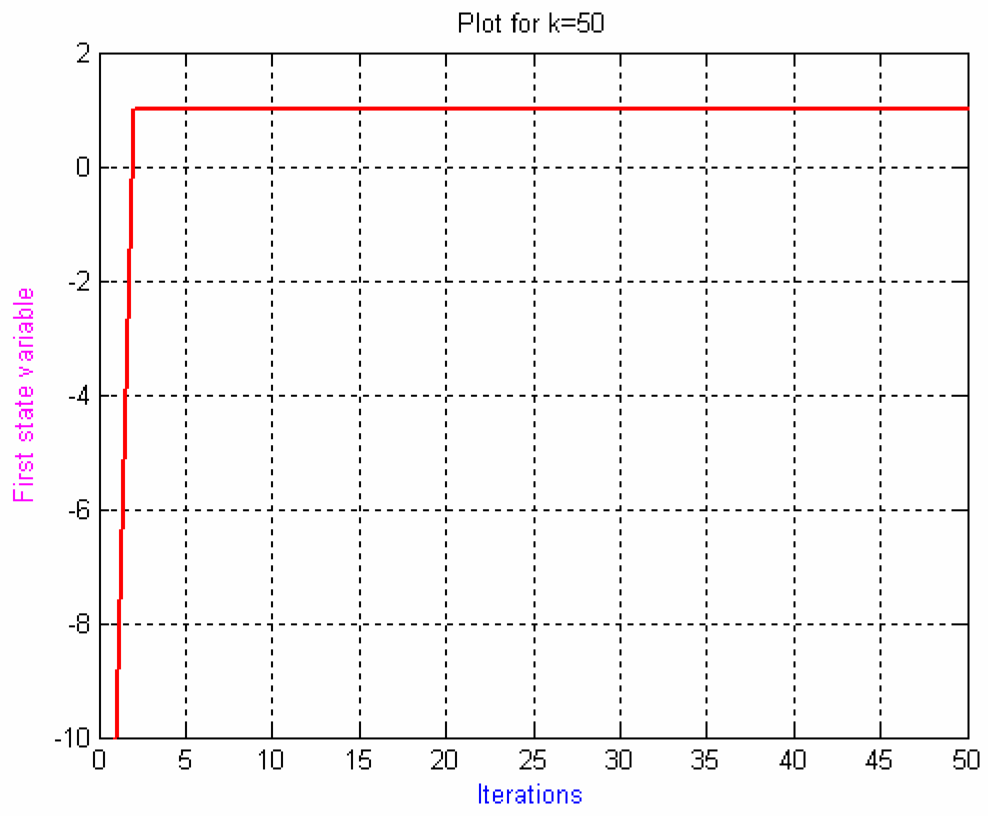
```
MED =

     1     1


MED =

     1     1

State vector after iteration  22 is                                    [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  23 is                                    [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  24 is                                    [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  25 is                                    [1;1]
```

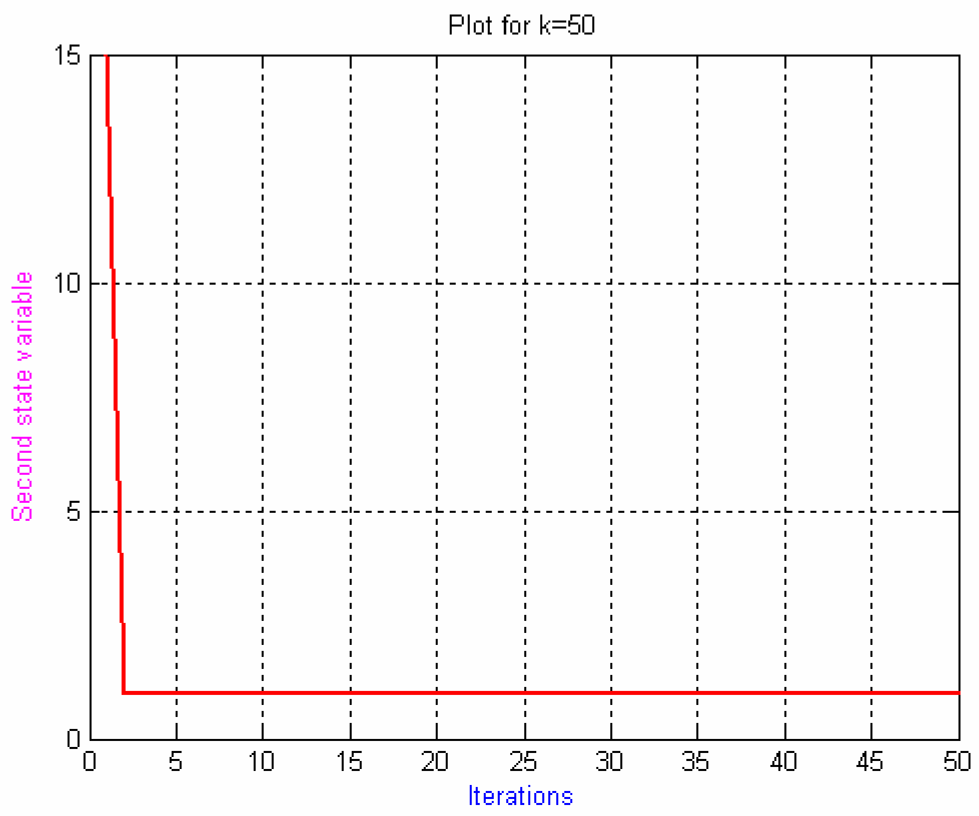
```
MED =

     1     1


MED =

     1     1

State vector after iteration  26 is                              [1;1]


MED =

     1     1


MED =

     1     1

State vector after iteration  27 is                              [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  28 is                              [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  29 is                              [1;1]

MED =
    1     1


MED =

    1     1

State vector after iteration  30 is                              [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration  31 is                              [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration  32 is                              [1;1]

MED =

    1     1


MED =

    1     1

State vector after iteration  33 is                              [1;1]

MED =

    1     1
```

```
MED =

     1     1

State vector after iteration  34 is                              [1;1]

MED =

     1     1


MED =

     1     1



State vector after iteration  35 is                              [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  36 is                              [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  37 is                              [1;1]

MED =

     1     1


MED =
     1     1

State vector after iteration  38 is                              [1;1]

MED =

     1     1

MED =

     1     1
State vector after iteration  39 is                              [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  40 is                              [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  41 is                              [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  42 is                              [1;1]
```

```
MED =

     1     1


MED =

     1     1

State vector after iteration  43 is                              [1;1]

MED =

     1     1


MED =

     1     1



State vector after iteration  44 is                              [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  45 is                              [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  46 is                              [1;1]

MED =
     1     1


MED =

     1     1

State vector after iteration  47 is                          [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  48 is                          [1;1]



MED =

     1     1


MED =

     1     1

State vector after iteration  49 is                          [1;1]

MED =

     1     1


MED =

     1     1

State vector after iteration  50 is                          [1;1]
The final state is
     1
     1
```

```
Entering 1 will display the firing strengths of various rules for
all the iterations.1
Iteration  1:    0    1    1    0    1    1    0    0    0
Iteration  2:    0    0    0    0    1    1    0    1    1
Iteration  3:    0    0    0    0    1    1    0    1    1
Iteration  4:    0    0    0    0    1    1    0    1    1
Iteration  5:    0    0    0    0    1    1    0    1    1
Iteration  6:    0    0    0    0    1    1    0    1    1
Iteration  7:    0    0    0    0    1    1    0    1    1
Iteration  8:    0    0    0    0    1    1    0    1    1
Iteration  9:    0    0    0    0    1    1    0    1    1
Iteration 10:     0    0    0    0    1    1    0    1    1
Iteration 11:     0    0    0    0    1    1    0    1    1
Iteration 12:     0    0    0    0    1    1    0    1    1
Iteration 13:     0    0    0    0    1    1    0    1    1
Iteration 14:     0    0    0    0    1    1    0    1    1
Iteration 15:     0    0    0    0    1    1    0    1    1
Iteration 16:     0    0    0    0    1    1    0    1    1
Iteration 17:     0    0    0    0    1    1    0    1    1
Iteration 18:     0    0    0    0    1    1    0    1    1
Iteration 19:     0    0    0    0    1    1    0    1    1
Iteration 20:     0    0    0    0    1    1    0    1    1
Iteration 21:     0    0    0    0    1    1    0    1    1
Iteration 22:     0    0    0    0    1    1    0    1    1
Iteration 23:     0    0    0    0    1    1    0    1    1
Iteration 24:     0    0    0    0    1    1    0    1    1
Iteration 25:     0    0    0    0    1    1    0    1    1
Iteration 26:     0    0    0    0    1    1    0    1    1
Iteration 27:     0    0    0    0    1    1    0    1    1
Iteration 28:     0    0    0    0    1    1    0    1    1
Iteration 29:     0    0    0    0    1    1    0    1    1
Iteration 30:     0    0    0    0    1    1    0    1    1
Iteration 31:     0    0    0    0    1    1    0    1    1
Iteration 32:     0    0    0    0    1    1    0    1    1
Iteration 33:     0    0    0    0    1    1    0    1    1
Iteration 34:     0    0    0    0    1    1    0    1    1
Iteration 35:     0    0    0    0    1    1    0    1    1
Iteration 36:     0    0    0    0    1    1    0    1    1
Iteration 37:     0    0    0    0    1    1    0    1    1
Iteration 38:     0    0    0    0    1    1    0    1    1
Iteration 39:     0    0    0    0    1    1    0    1    1
Iteration 40:     0    0    0    0    1    1    0    1    1
Iteration 41:     0    0    0    0    1    1    0    1    1
Iteration 42:     0    0    0    0    1    1    0    1    1
Iteration 43:     0    0    0    0    1    1    0    1    1
Iteration 44:     0    0    0    0    1    1    0    1    1
Iteration  45:      0      0      0      0      1      1      0      1      1
Iteration  46:      0      0      0      0      1      1      0      1      1
Iteration  47:      0      0      0      0      1      1      0      1      1
Iteration  48:      0      0      0      0      1      1      0      1      1
Iteration  49:      0      0      0      0      1      1      0      1      1
Iteration  50:      0      0      0      0      1      1      0      1      1
Entering 1 will plot the  trajectory for the first state variable. 1

ii =

  Columns 1 through 10

     1     2     3     4     5     6     7     8     9    10

  Columns 11 through 20

    11    12    13    14    15    16    17    18    19    20

  Columns 21 through 30

    21    22    23    24    25    26    27    28    29    30



Columns 31 through 40

    31    32    33    34    35    36    37    38    39    40

  Columns 41 through 50

    41    42    43    44    45    46    47    48    49    50

Entering 1 will plot the trajectory for the second state variable.
```





Figures 4.5 and 4.6

4.3 Conclusion

A new matrix Dic, which involves the control gain, K represents Dic Discussed in chapter 3. Thus, for stabilizing control, a control gain, K needsTo be found such that XtDicX is negative definite. This condition is achivedIf all the eigenvalues of Dic are negative. Solution of the new LyapunovEquation is needed in order to find Dic. The new Lyapunov equation can be solved only if Ao+boK is stable. First a discrete range of K satisfying thisCondition is found, and then eigen values of Dic are calculated for differentK’s. A value of K is choosen for which either all the eigenvalues of Dic areNegative or the negative eigenvalues are much larger in magnitude as compared to the positive ones.

Chapter 5
Conclusions, Limitations, and Suggestions For Further Work

From causes which appear similar, we expect similar effects. This is the sum total of all our experimental conclusions.

5.1 Conclusions

Lyapunov's approach offers a shortcut to proving the global stability of a dynamical system. A Lyapunov function summarizes total system behaviour. At each moment a single real number represents the entire system. This is somewhat similar to the case in statistical mechanics, where a single numerical temperature summarizes the interactions of arbitrarily many molecules.

A dynamical system is stable if some Lyapunov function L decreases along trajectories: $L \leq 0$. A dynamical system is asymptotically stable if it strictly decreases along trajectories: $L < 0$. Monotonicity of a Lyapunov function provides a *sufficient* not necessary condition for stability and asymptotic stability. Inability to produce a Lyapunov function proves nothing. The system mayor may not be stable. Demonstration of any Lyapunov function proves stability.

The two stability test methods discussed in the previous chapters give a sufficient condition, which guarantees the stability of fuzzy systems in terms of Lyapunov's direct method. The following important conclusions can be drawn from the discussions:

(1) If all the subsystem matrices are stable, the overall fuzzy system may or may not be stable.

(2) Even if some of the subsystem matrices are unstable, the overall fuzzy system may be stable.

(3) The existence of a common positive definite matrix satisfying Lyapunov equation for all the subsystem matrices shows that the fuzzy system is stable.

(4) If all the subsystem matrices are stable, but the product of any two subsystem matrices is unstable, there can't be any common positive definite matrix satisfying Lyapunov equation for all the satisfying Lyapunov equation for all the subsystem matrices.

(5) Negative bounds approach used to test the stability of fuzzy systems is better than common positive definite matrix approach in the sense that it does not

fail even if some of the subsystem matrices are unstable.

(6) Though the negative bounds approach gives only a sufficient condition of stability like common positive definite matrix approach, an algorithm based on this approach can be used to identify some stable systems as stable.

(7) In case of stabilization using linear feedback control, negative bounds approach can be used for the determination of control gains, which can stabilize the system.

5.2 Limitations of the Given Methods

Both the stability test methods give sufficient but not necessary conditions of stability. This means that a system proved to be stable using any of these methods is actually stable, but those systems, which cannot be proved to be stable, may also be stable.

In order to check the stability of a fuzzy system using the method given in chapter 2, we must find a common positive definite matrix. No procedure has been suggested for finding this matrix.

In negative bounds approach, the procedure given for the determination of bounds cannot always be adopted. This procedure does not give tight bounds and therefore desirable results may not be obtained in some cases.

5.3 Suggestions for Further Work

1. An effective algorithm for finding common positive definite matrix needs to be developed. This will considerably improve the stability test method given in chapter 2.
2. Attempts should be made to discover sufficient as well as necessary conditions of stability for fuzzy systems.

References

[Junhong's98]	Junhong Nie and Derek Linkens., " Fuzzu and Neural Control, PHI, 1998
[Driankov'93]	Driankov, D.,Hellendoom, H., and Rinfank, M., "An introduction to Fuzzy Control," Narosa,1993
[Kim'95]	Kim, W. C.,Ahn,S.C., and Kwon, W.H., " Stability Analysis and Stabilization of Fuzzy State Space Models," Elsvier, Fuzzy Sets and Systms, 71(1995) 131-142
[Kosko'92]	Kosko, B., "Neural Networks and Fuzzy Systems: A Dynamical Sustems Approach to Machine Intelligence," Englewood Cliffs, NJ: Prentice – Hall,1992
[Kuo'80]	Kuo, B. C., " Digital Control Systems," Holt-Saunders,1980
[Takagi'85]	Takagi, T., and Sugeno, M., " Fuzzy Identification of Systems and its Applications to Modelling and control," IEEE Trans. On Systems, Man, and Cybernetics,15(1985)116-132.
[Tanaka'92]	Tanaka, K., and Sugeno, M., "Stability Analysis and Design of Fuzzy Control Systems," Elsevier, Fuzzy Sets and systems , 45(1992) 135 -156.