ROBUST ANALYSIS OF FEEDBACK SYSTEMS WITH PARAMETRIC AND DYNAMIC STRUCTURED UNCERTAINTY

Thesis by

Ricardo S. Sánchez Peña

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1989

(Submitted June 3, 1988)

to Mónica, Pablo and Lucila

Acknowledgments

I wish to thank the Organization of American States and the Comisión Nacional de Investigaciones Espaciales who supported me during my first two years at Caltech. Also the Consejo Nacional de Investigaciones Científicas y Técnicas of Argentina for granting me a fellowship for these last two years, as well as the California Institute of Technology for its financial aid in the form of teaching and research assistanships.

I would like gratefully to acknowledge my advisor Prof. Athanasios Sideris for introducing me in the area of parametric uncertainty and for the guidance, help and friendship he offered me throughout this work. Prof. John Doyle also played an important role in my education and I owe him many thanks for being always responsive and for the new world of ideas he lead me into. I especially would like to mention my gratitude to Prof. Manfred Morari for encouraging me to continue studying for my Ph.D. in difficult moments of my career. Finally I owe a great deal of thanks to Prof. Pedro Zadunaisky, with whom I had the pleasure to work in Argentina and who inspired me to pursue a higher academic level.

I also would like to thank other control students I met at Caltech with whom I had enlightning discussions and who along these years offered me their friendship: Lane Dailey, Roy Smith, Matt Newlin, Davut Kravanoğlu, Gary Balas, Bobby Bodenheimer, Marty Brenner and special thanks to Andy Packard who provided useful comments in the writing of parts of this thesis.

I would like to thank my parents who continuously helped and encouraged me in the pursuit of this objective and finally I owe a great deal of gratitude to my wife Mónica for her love and understanding and to my children Pablo and Lucila just because they are lovely.

Abstract

This thesis presents the first general program implementation of the algorithm by deGaston and its generalization by Sideris and deGaston to compute the Multivariable stability margin or Structured singular value of a feedback system under real (independent or related) parametric uncertainty. An improved implementation of the algorithm mentioned above is also considered, which simplifies significantly the code and increases the computational speed. The latter also allows a simple and fast analysis by just checking the extreme values of the set of parameters, with a high probability of achieving the actual stability margin, this being supported by an intense statistical analysis performed at the end of this thesis.

A great deal of work has recently been done related to this class of uncertain systems initiated by the well known theorem of Kharitonov. A connection is made in Chapter 4 between these procedures and the above ones in terms of generality of the class of uncertain polynomials considered. A theorem characterizing the set of polynomials whose robust stability can be determined by a finite number of tests is addressed. Sufficient conditions to determine when the latter conditions apply are also given, which in some cases can considerably simplify the analysis. In particular cases, polynomials with related uncertain parameters can be treated in the same way as independent parameters as shown in two examples.

The main part of this thesis is concerned with the analysis of more general type of uncertainties. In particular, the analysis of robust stability for the case when unstructured dynamic uncertainty is combined with real parametric uncertainty is treated in Chapter 5. This can also be applied in the analysis of robust performance for plants with parametric uncertainty. Chapter 6 generalizes the latter to the most general case in which **structured** dynamic and real parametric uncertainty appear simultaneously in the plant. A computational scheme is given in both cases which uses the algorithm mentioned in the first part and is applied to several examples.

At the end, an example of the robust analysis of an experimental aircraft demonstrates how a practical situation can be handled by this procedure.

Table of Contents

List of Tables and illustrations

Chapter 1

Introduction

1.1 CONTROL PHILOSOPHY

In the analyis and design of control systems we need to clearly distinguish two stages. The first is concerned with the problem of selecting a reasonable mathematical model for the physical system that it will describe, given the set of inputs and outputs we will be dealing with. A very rich family of these models that will adequately "fit" many physical systems of interest is the set of finite dimensional, linear, time-invariant ordinary differential equations (FDLTI). Most of the research in control theory has dealt with this type of system description and all the material in this thesis will also deal with these systems.

In the second step we directly work on the mathematical model selected and try, by using feedback, to make the closed loop system stable and obtain an adequate performance defined in some sense. Two conceptually different kinds of unknown perturbations will make this task nontrivial. First, the unavoidable presence of external noise and disturbances introduced additively into the system, due to sensor noise and in general by the interaction with the external environment. Usually performance is defined in terms of minimizing the effect of these type of perturbations at the output of the feedback loop.

The second kind of perturbation gives us the connection between the physical system and the mathematical model and is described as an uncertainty in the latter. In this way, we are dealing not with a single model, but with a family of models that will hopefully include the actual physical plant. Model uncertainty will arise due to two main reasons. The first one has to do with the limitations on the **structure** of the model and appears because of linearization errors and neglected higher order dynamics. The second type of model uncertainty is due to errors in the **parameters** of the model which naturally arise in the parameter identification process. Both kinds of uncertainty can appear independently and in general will also appear simultaneously.

There is an important conceptual difference in FDLTI models between the two types of perturbations described before. The first type (noise, disturbances) can deteriorate significantly the performance, but the second kind (model uncertainty) can make the closed loop system unstable which invalidates even the definition of performance. This is the main reason why an increased interest in this last kind of perturbation has arisen in the last years.

Before we continue with the analysis of model uncertainty, a brief historical overview will introduce the subject of robustness - the guarantee of stability and performance in the presence of model uncertainty.

1.2 BACKGROUND

Classical control methods have been very effective in dealing with the design and analysis problems stated above. Even today, many control systems are designed and built following its simple procedures. The main characteristics of this approach are the following. It is restricted to Single Input Single Output (SISO) systems and reduces the whole system to be analyzed to a dominant second order differential equation, the performance objectives being easily translated from time domain specifications to frequency domain specifications. The latter also allows the designer to have a better insight into the conflicting requirements of the system.

Also the simplicity of working in the frequency domain (i.e. with the eigenvalues of the FDLTI operator) has been one of the fundamental legacies of classical control theory to the methods used nowadays. In terms of robustness, the concepts of phase and gain margin were useful to deal with many of the problems encountered. The names of Bode, Nyquist, Evans and Nichols are associated with these initial steps in control theory.

Although this worked in many situations, the trial and error approach and the oversimplification of the physical system makes this design methodology ineffective for complex system descriptions and for more stringent requirements. In the late 50's the state space description of systems entered the picture which basically allowed designers to treat SISO and Multiple Input Multiple Output (MIMO) systems in the same way. The fast development of computers directs the approach to an algorithmic straightforward design methodology. The theory of systems becomes more mathematically elegant although less insightful. The concept of optimality alters the designer's question from - "Can the system meet the specifications ?" to - "What is the best we can achieve with this system ?". Finally, the dual theory of optimal regulators and optimal observers leads to an algorithmic design procedure known as LQG (Linear Quadratic Gaussian). From this period, the state space model remained as the most efficient way to deal computationally with the analysis and design of control systems. The names of Kalman, Bucy, Pontryagin and Luenberger among others defined this stage.

Although this was an important step towards a theory that could deal with more "realistic" situations, robustness against plant model uncertainties was still an unsolved issue[1]. Only by assuming perfect measurements could we arrive at a good robust design[2]. The concept of plant model uncertainty and robustness motivated most of the research efforts from that point on.

A way to recover the robustness characteristics of the optimal regulator (LQR) in the LQG design was developed by Doyle and Stein[3], called the LQG Loop Transfer Recovery (LQG/LTR) procedure. A generalization of the concept of loop shaping for MIMO systems by using the singular values of the closed loop matrix introduced the robustness issue into the design procedure in a way similar to the classical control methods[3,4].

Finally, the H_∞ design methodology appeared as an algorithmic procedure that would shape the singular values of the feedback system matrix to solve the problem of robustness for the case of unstructured dynamic uncertainty in the plant model. It was introduced by Zames[5] and developed by many others [6,7,8] during these last years. Very recently Doyle et al. developed a computationally more efficient[9] way to solve this problem, as well as a duality theorem similar to the one in the LQG procedure.

Still the limitation is that the unstructured dynamic model uncertainty considered above can be very conservative in describing many practical examples. One of these situations appears in plants with uncertainty described by unknown real parameters in the differential equations of the model. Another example is the case were we have dynamic uncertainty in different components of the system. Both these situations can be described mathematically as a block diagonal uncertainty matrix Δ, its elements being either the real parameter errors or the dynamic block uncertainty of each of the different components of the model, respectively (see figure 1.1). This type of uncertainty is defined as **structured** uncertainty and depending on the nature of this structure we will use different approaches to analyze these kinds of plants.

1.3 STRUCTURED UNCERTAINTY

In the analysis of feedback control systems it is important not only to determine the stability and performance properties of the nominal closed loop system, but also to guarantee that such properties are *robust* with respect to certain plant model uncertainty. Thus the issues of robust stability and performance have been given considerable attention in the control literature and depending on how the set of plant perturbations is defined, various results have been proposed to check these feedback properties.

Most notable measures of robustness which have been proposed in the literature are the Multivariable Stability Margin[10] and the Structured Singular Value[16]. These measures are defined next for easy reference and some related results are briefly summarized.

1.3.1 Multivariable Stability Margin (MSM)

In [10] Safonov considered a canonical block diagonal perturbation system obtained by rearranging uncertainty blocks and parameters from various plant locations into a block diagonal form (see figure 1.1). The condition for the robust stability of the feedback system is given in terms of the multiloop stability margin k_m, which is defined as

$$k_m \overset{\text{def}}{=} \inf\{k \in [0,\infty) \mid \det(I + k\Delta M) = 0 \quad \text{for} \quad \text{some} \quad \Delta \in \mathbf{\Delta}\} \tag{1.1}$$

where

$$\mathbf{\Delta} \overset{\text{def}}{=} \{diag[\Delta_1, \Delta_2, ..., \Delta_n] \mid \Delta_i \in \mathcal{C}^{m_i x m_i}\} \tag{1.2}$$

and the Δ_i's are norm bounded, i.e. by using stable and minimum phase weighting factors incorporated in the nominal block $M(s)$, we can assume with no loss of generality that $\bar{\sigma}(\Delta_i) \leq 1$ for all ω, where $\bar{\sigma}(\cdot)$ denotes the maximum singular value, i.e. $\rho(\Delta_i^* \Delta_i)$ being ρ the spectral radius and Δ_i^* the conjugate transpose of Δ_i.

Then robust stability is assured if and only if $k_m \geq 1$ for all ω. Clearly k_m is a function of frequency, of the nominal plant and also of the block uncertainty structure. In [11,12] an algorithm is derived that exactly calculates k_m in the case of unrelated real parametric uncertainty, i.e. $\Delta_i = \delta_i \in \mathcal{R}$.

This corresponds to having a closed-loop characteristic polynomial with coefficients being multilinear functions of the parameters δ_i.

In [13] this algorithm is extended to the case of polynomially related real parameters, that is the case of closed-loop characteristic polynomials with coefficients being general multivariate polynomials in the δ_i's. For a computer implementation of the algorithm in [13] see [14]. Another algorithm in the same spirit is reported in [15].

1.3.2 Structured Singular Value (SSV)

On the other hand, Doyle defines in [16] the structured singular value $\mu(\cdot)$ as:

$$\mu_{\mathbf{\Delta}}(M) = \begin{cases} 0 \quad \text{if} \quad \det(I + \Delta M) \neq 0 \quad \forall \Delta \in \mathbf{\Delta} \\ \qquad\qquad \text{or} \\ \left[\inf_{\Delta \in \mathbf{\Delta}} \{\max_i \bar{\sigma}(\Delta_i) \mid \det(I + \Delta M) = 0\}\right]^{-1} \end{cases} \tag{1.3}$$

where $\mathbf{\Delta}$ is defined as in (1.2). From (1.1) and (1.3) we observe that

$$\mu_{\mathbf{\Delta}}(M) = {k_m}^{-1}(M) \tag{1.4}$$

In the case of unrepeated complex blocks Δ_i, the calculation of μ is achieved by the use of the following upper and lower bounds:

$$\sup_{U \in \mathcal{U}} \rho(MU) \leq \mu_{\mathbf{\Delta}}(M) \leq \inf_{D \in \mathcal{D}} \bar{\sigma}\left(DMD^{-1}\right) \tag{1.5}$$

where

$$\boldsymbol{\Delta} \overset{\text{def}}{=} \{diag\,(\Delta_1,\ldots,\Delta_i,\ldots,\Delta_n) \mid \Delta_i \in \mathcal{C}^{m_i x m_i}\} \tag{1.6}$$

$$\mathcal{U} \overset{\text{def}}{=} \{diag\,(U_1,\ldots,U_i,\ldots,U_n) \mid U_i^* U_i = I\} \tag{1.7}$$

$$\mathcal{D} \overset{\text{def}}{=} \{diag\,(d_1 I,\ldots,d_i I,\ldots,d_n I) \mid d_i \in \mathcal{R}_+\} \tag{1.8}$$

Robust performance is also naturally addressed in this framework[16] as an extension of the concept of robust stability, by considering an extra *performance* block in the uncertainty structure $\boldsymbol{\Delta}$.

The main theorem in [16] establishes in (1.5) equality for the lower bound for any n, and also for the upper bound for $n \leq 3$. However the optimization problem defining the upper bound for μ is convex while the one defining the lower bound is not. Therefore the SSV is usually estimated by using its upper bound. Efficient algorithmic procedures can be found in [17,18,19].

Although the upper bound has been found to be in general tight for complex uncertainty blocks for any $n > 3$, it can be arbitrarily conservative in the case of real parametric uncertainty.

Thus the approach of replacing real parameters with complex ones leads to conservative results in many cases of practical interest. Attempts to reduce such conservatism by exploiting the degrees of freedom in covering the variation of a real parameter by that of a complex one, while in special cases are successful, in general demonstrate that the real parameter case is fundamentally different from the complex one.

1.3.3 Coefficient uncertain polynomials

The problem of robust stability in systems with real parametric uncertainty is also the subject of intense research from the viewpoint of looking at the coefficients of the closed loop characteristic polynomial as functions of the uncertain parameters. The problem reduces to check if the roots of the closed loop characteristic polynomial (CLCP)

$$f(s,p) = s^n + c_{n-1}(p) \cdot s^{n-1} + \ldots + c_1(p) \cdot s + c_0(p) \qquad (1.9)$$

are inside a particular region in the complex plane, where $p \in \mathcal{R}^r$ is the set of parameters with $p_i \in [a_i, b_i]$. This region could be the open left-half plane to check for robust stability, a smaller region for robust performance defined in the sense of closed-loop pole locations or the unit disc if instead of (1.9) we have a similar equation in the z-transform to analyze a digital system.

The approach is to determine if the roots are inside or outside the given region by testing the stability of a subset of polynomials in the family described by (1.9). It has been initiated by Kharitonov[20] and has led to many interesting works in the area[21–27]. Some of them make a *qualitative* analysis of robustness[20,23,24,25,26], while others give a *quantitative* answer of robustness in terms of a single scalar value[21,22]. The main restriction is given by the fact that the coefficients $c_i(\cdot)$ need to be either all independent[20–23], even and odd sets independent[24] or linear in the parameters[25,26]. This completely restricts the applicability of this analysis. As we can see, even in the simple case of two uncertain pole locations of the form:

$$f(s,p) = (s+p_1)(s+p_2) = s^2 + (p_1+p_2)s + p_1p_2 \quad ; \quad p_1, p_2 \in [0,1] \qquad (1.10)$$

we cannot handle it in a nonconservative way by using [20-24] because it has dependent coefficients or by the "edge check" of [25,26], not being a polytopic coefficient region.

1.3.4 More general uncertainty structures

There have been some recent results dealing with the problem of computing a robustness measure for systems combining both real parametric and complex dynamic uncertainty. In [28], the case of real parameters and one unstructured complex block is discussed for SISO and minimum phase plants. In [29], the analysis is applied to SISO plants with the same uncertainty structure but with the parameters varying inside a polytope. In [30], the case of non-repeated real parameters and simultaneous unstructured dynamic uncertainty is treated in the case of MIMO systems.

The main objective of this thesis is to obtain a procedure to compute in a non-conservative way the stability margin defined in 1.3.1 and 1.3.2 for more general uncertainty structures - that is the combination of real (possibly related) uncertain parameters and structured dynamic uncertainty. Furthermore, an efficient implementation of this procedure for the real parametric uncertainty case that could be extended to the more general situation is also addressed. The contents are as follows. Chapter 2 will describe the theory to compute exactly k_m in the case of real related parameters as well as details on the implementation of this procedure in a computer code. Chapter 3 will describe another implementation of this program that will increase significantly the computational speed. Chapter 4 will relate this approach with the ones described in section 1.3.4 and will give some shortcuts that can simplify the statement of complicated problems in many cases. Chapter 5 addresses the exact computation of these stability margins for the important case of real related uncertain parameters combined with unstructured dynamic uncertainty. The next chapter generalizes this last approach to the more general case of structured dynamic uncertainty. All the chapters just mentioned will also show with examples the application of their contents. Chapter 7 describes a statistical analysis on the performance of the algorithm implementations described in Chapters 2 and 3 and Chapter 8 gives a detail of a practical application of this analysis to an experimental aircraft. Finally conclusions and directions for future research are drawn in Chapter 9. Parts of this thesis have been already published or submitted for publication[14,49,50,51].

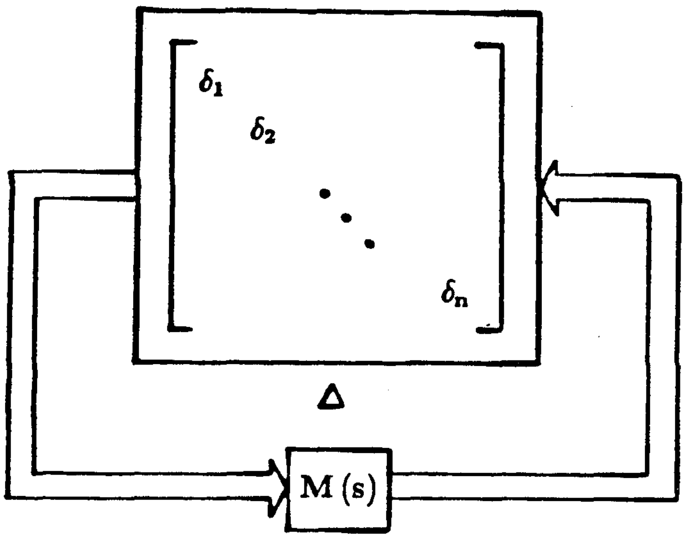


Figure 1.1: General uncertainty description and nominal system.

Chapter 2

Program to compute k_m: Theory and implementation

2.1 INTRODUCTION

A computer program implementing the algorithm in [11-13] for computing the Multivariable Stability Margin to check the robust stability of feedback systems with real parametric uncertainty is proposed. An example demonstrating the performance of the program is discussed. This chapter reports the first general implementation of this algorithm for the case of independent parameters. More specifically, all possible cases that can arise in applications of the theory have been given consideration and programmed accordingly. Furthermore, this algorithm is perhaps the first to treat the related parameter case as reported in [13].

The contents of this chapter are as follows. In section 2.2, a compact description of the results in [11-13] is presented. In section 2.3, we describe in some detail important aspects of the program and apply it to an example in section 2.4. Finally an Appendix details one of the procedures in the program.

2.2 SUMMARY OF PREVIOUS RESULTS

In the algorithm of [11-13] for the exact computation of k_m, we can identify two mechanisms. The first is used to produce upper and lower bounds on k_m. The second is used to refine these bounds, so that an increasing sequence of lower bounds and a decreasing sequence of upper bounds that converge to k_m are produced. In order to simplify the exposition we first explain the algorithm for $\delta_i's$ independent of each other. The more general case is treated next.

2.2.1 k_m for independent real parameters

The derivation of upper and lower bounds on k_m is based on the observation that the function $f: \ \delta = (\delta_1, \ldots, \delta_n) \in \mathcal{R}^n \longrightarrow \mathcal{C}$, defined by:

$$f(\delta) = det[I + k\Delta M] \tag{2.1}$$

is multilinear, i.e. linear (to be more precise affine) in each of the $\delta_i's$. This property of $f(\cdot)$ is responsible for the following lemmas[11–12]. We first introduce some notation. Let

$$\mathcal{D} \overset{\text{def}}{=} \{\delta \in \mathcal{R}^n / |\delta_i| \leq 1, \ i = 1, \ldots, n\} \tag{2.2}$$

denote the nth dimensional hypercube, and

$$V \overset{\text{def}}{=} \{V_i = \delta \in \mathcal{D} | \delta_j = 1 \ or \ -1, \ j = 1, \ldots, n\} \tag{2.3}$$

be the vertices of $\mathcal{D}$. Let also $f(\mathcal{D})$ denote the image of $\mathcal{D}$ and $M_i \equiv f(V_i)$ the images of the vertices of $\mathcal{D}$ by f. Finally let $co\{M_i\}$ denote the convex hull of the complex points M_i, $i = 1, \ldots, 2^n$ (i.e. the smallest convex set that contains the $M_i's$).

Lemma 2.2.1.1 (see figure 2.1):

$f(\mathcal{D})$ is contained in $co\{M_i\}$.

Lemma 2.2.1.2 (see figure 2.2):

Let f_1 and f_2 be defined by (2.1) for $k = k_1$ and $k = k_2$ respectively. Then for $k_1 > k_2$ it holds $f_1(\mathcal{D}) \supset f_2(\mathcal{D})$, and $co\{M_i\}_{k=k_1} \supset co\{M_i\}_{k=k_2}$.

Lemma 2.2.1.3 (see figure 2.1)

The image of an hypercube edge V_iV_j is the line segment M_iM_j.

Lemma 2.2.1.1 allows a very efficient procedure for obtaining a lower bound on k_m. More specifically we compute k_{low}, the smallest value of k in (2.1) for which the origin of the complex plane belongs to $co\{M_i\}$. Note that for $k = 0$, $f(\mathcal{D}) = \{1\}$, and by increasing k, $co\{M_i\}$ expands because of lemma 2.2.1.2 . Since $f(\mathcal{D}) \subset co\{M_i\}$, it follows that $0 \notin f(\mathcal{D})$ for $k < k_{low}$ and definition (1.1) gives $k_{low} \leq k_m$. To obtain an upper bound on k_m, we compute k_{upper}, the smallest value of k in (2.1) for which a point on some edge of $\mathcal{D}$ maps onto the origin. This can be easily done since by lemma 2.2.1.3 the images of the hypercube edges are line segments joining the images of the hypercube vertices. Since for $k < k_{upper}$ some point in $\mathcal{D}$ other than on an edge may be mapped onto the origin, we have $k_m \leq k_{upper}$.

We remark that the previous bounds on k_m are cheaply computed by mapping the vertices of $\mathcal{D}$ on the complex plane by f for various values of k. These bounds can be made to approach k_m as closely as desired. The basic idea is as follows. At the first step the hypercube is divided in two parts, and bounds are computed for each part as before (see figure 2.3). It can be shown that the bounds for each part can be combined to produce tighter bounds on k_m. This procedure is continued by subdividing at each step the subdomains of $\mathcal{D}$ derived at the previous step.

The number of subdomains at each step would increase exponentially and the algorithm would have been impractical, if there was no way to drop the majority of subdomains at each step. The mechanism to do this is again based on the lower and upper bounds obtained for each subdomain. Such a subdomain need not be considered any further, if its lower bound is greater than the upper bound of any of the other subdomains. This situation has been verified in computer simulations and as a consequence rapid convergence to k_m is observed.

A byproduct of the above algorithm is the combination of parameters that causes instability, if this is the case. This information is expected to enhance considerably the designer's intuition about the system.

2.2.2 k_m for related real uncertain parameters

A crucial assumption in the previous section was that the uncertain system parameters were unrelated, i. e., there are no functional relationships among the δ_i's. However, this is usually not the case in practice. For example suppose that the closed loop characteristic polynomial is given by

$$f(s) = s^2 + 2\zeta\omega_n s + \omega_n^2 \tag{2.4}$$

where ζ and ω_n are uncertain. To bring the closed loop system to the canonical diagonal form of figure 1.1, we have to select as our uncertain parameters $\delta_1 = \zeta\omega_n$, $\delta_2 = \omega_n^2$ or $\delta_1 = \zeta$, $\delta_2 = \omega_n$, $\delta_3 = \omega_n^2$, and in both cases the δ_i's are obviously related. In such cases where uncertain parameters are related, the algorithm of the previous section fails because $f(\delta) = det[I + k\Delta G(j\omega)]$ is not defined on all of the hypercube $\mathcal{D}$. Thus this algorithm will produce in general only a lower bound on k_m and the stability test will be conservative. In the following steps we show how to reformulate the problem in a way that this conservatism is eliminated. We assume that the relations among the parameters are polynomial. This is flexible enough to cover most cases of practical interest, but also any smooth nonlinear function of the $\delta_i's$ can be approximated on a compact set of $\mathcal{R}^n$ as closely as desired by a multivariable polynomial[31]. Details and proofs can be found in [13].

Step 1 (Modification of the mapping function)

In order to define f as given by (2.4) on an hypercube, we select as our uncertain parameters exactly those that are independent (e. g. in the previous example $\delta_1 = \zeta,\ \delta_2 = \omega_n$). However the resulting mapping function f is not multilinear and the procedure for obtaining upper and lower bounds on k_m breaks down. This situation is rectified as follows. Let $m(i)$ be the highest degree of δ_i in $f(\delta)$. Consider fictitious variables $\delta_{i1}, \ldots, \delta_{im(i)},\ i = 1, \ldots, n$ and replace in $f(\delta)$ each δ_i^p with $\delta_{i1}\delta_{i2}\ldots\delta_{ip}$. Define:

$$\overline{\delta} \overset{\text{def}}{=} (\delta_{11}, \ldots, \delta_{1m(1)}, \ldots, \delta_{n1}, \ldots, \delta_{nm(n)}) \in \mathcal{R}^{\overline{n}} \tag{2.5}$$

where $\overline{n} = \sum_{i=1}^{n} m(i)$, and

$$\overline{\mathcal{D}} \overset{\text{def}}{=} \{\overline{\delta} = (\overline{\delta_1}, \ldots, \overline{\delta_{\overline{n}}}) / |\overline{\delta_k}| \leq 1,\ k = 1, \ldots, \overline{n}\} \tag{2.6}$$

Note that $\mathcal{D}$ is embedded in $\overline{\mathcal{D}}$ in a canonical manner. In fact

$$\mathcal{D} = \{\overline{\delta} \in \overline{\mathcal{D}} / \delta_{ik} = \delta_{il},\ i = 1, \ldots, n,\ k, l = 1, \ldots, m(i)\} \tag{2.7}$$

.

The above construction results in a multilinear function $\overline{f}(\overline{\delta})$ on $\overline{\mathcal{D}}$, such that $\overline{f}(\overline{\delta}) = f(\delta)$ on $\mathcal{D}$. To illustrate the above transformation consider the previous example. Let $\delta_1 = \zeta,\ \delta_2 = \omega_n$. Then $f(\delta_1, \delta_2) = s^2 + 2\delta_1\delta_2 s + \delta_2^2$. Now define $\overline{\delta_1} = \delta_1,\ \overline{\delta_2} = \delta_2,\ \overline{\delta_3} = \delta_2$. We obtain $f(\overline{\delta_1}, \overline{\delta_2}, \overline{\delta_3}) = s^2 + 2\overline{\delta_1} \cdot \overline{\delta_2} s + \overline{\delta_2} \cdot \overline{\delta_3}$ defined on the unit cube in $\mathcal{R}^n$, which is a multilinear function in $\overline{\delta}$ and reduces to $f(\delta)$ if we restrict $\overline{f}$ to the diagonal plane $\overline{\delta_2} = \overline{\delta_3}$ (see figure 2.4).

Step 2 (Derivation of bounds on k_m)

Since $\mathcal{D} \subset \overline{\mathcal{D}}$ as explained above, the lower bound mechanism as applied to $\overline{f}$ and $\overline{\mathcal{D}}$ will still result in a lower bound on k_m. However, this is not the case for the upper bound mechanism, because not all of the edges of $\overline{\mathcal{D}}$ belong to $\mathcal{D}$. This is fixed as follows. Consider the edges of $\mathcal{D}$. If an edge of $\mathcal{D}$ is also an edge of $\overline{\mathcal{D}}$ (for example edges V_1V_2, V_5V_8 in figure 2.4), then it is mapped by $\overline{f}$ on a straight line segment because of lemma 2.2.1.3 . If an edge of $\mathcal{D}$ is not an edge of $\overline{\mathcal{D}}$ (for example edges V_1V_5, V_2V_8 in figure 2.4), we want to find an easily constructed region on the complex plane that contains the image of that edge. To this end consider the smallest face of $\overline{\mathcal{D}}$ (obtained by fixing the values of some coordinates at their extreme values 1 or -1), that contains the edge (for example in figure 2.4 for edge V_1V_5 such face is $V_1V_4V_5V_6$). The convex hull of the image of such faces of $\overline{\mathcal{D}}$ by $\overline{f}$ is easily obtained by mapping the vertices on the face and using lemma 2.2.1.1. Next let k in (2.4) increase until the images of the edges of $\mathcal{D}$, or of the faces containing these images in case these edges are only diagonals of $\overline{\mathcal{D}}$, go over the origin (see figure 2.4). In this last case, we should also check that the line connecting both constrained vertices goes through the origin an odd number of times. The value of k thus obtained is an upper bound on k_m.

Step 3 (Refinement of the bounds on k_m)

The bounds on k_m obtained in step 2 can be made to approach k_m arbitrarily close by the subdivision procedure outlined in subsection 2.2.1. However, we can take great advantage of the canonical constraints among the δ_i's to reduce the computations in this procedure. A subdivision of $\overline{\mathcal{D}}$ by a plane vertical to one of its coordinates can be clearly identified with that coordinate. Then we prescribe that if a coordinate $\overline{\delta_i}$ is subdivided in the process of subsection 2.2.1, all other coordinates related to $\overline{\delta_i}$ by equality constraints as in (2.7) are also subdivided. In this manner $\overline{\mathcal{D}}$ is split to $2^{m(i)}$ parts, where $m(i)$ is the number of variables constrained to be equal to $\overline{\delta_i}$ in (2.7). But from these subdomains of $\overline{\mathcal{D}}$ only two contain parts of $\mathcal{D}$

and the rest can be dropped from any further consideration (see figure 2.4).

We remark that the algorithm of this section retains the essential characteristics of the algorithm of subsection 2.2.1, and although it handles a more difficult problem, it requires a computational amount of the same order.

2.3 PROGRAM IMPLEMENTATION

There are three main procedures in the program which will be explained separately. Differences in each procedure in the independent or related parameter cases are pointed out. Before going into this we discuss how data is inputed to the program.

2.3.1 Data input modes

There are two basic ways to input the closed-loop system, k and the set of variations $\mathbf{\Delta}$ to define the mapping function. One is by using the structure in figure 1.1, in which case the mapping function is given by:

$$f(s, k\Delta) = det\,[I + k\Delta M(s)] \tag{2.8}$$

with Δ defined in (1.2). The other is by using the closed loop characteristic polynomial $\tilde{f}$, as a function of the parameters p,

$$\begin{gathered} \tilde{f}(s,p) = f(s, k\Delta) \\ p \overset{\text{def}}{=} \left\{ [p_1, \ldots, p_n]^t \mid p_i = p_{io} + k\delta_i \quad ; \quad i = 1, \ldots, n \right\} \end{gathered} \tag{2.9}$$

and Δ as before.

In the second case we can also perform a preliminary test, this is checking if all coefficients of the s powers are positive for $k = 1$. If this is the case we can proceed, otherwise we can easily find $k_m < 1$ for which one of these coefficients vanishes and the corresponding p for which this occurs, with no further computations to be done.

2.3.2 Lower bound search

An important difference in the two cases mentioned above is that the nominal value in (2.8) is $f(s,0) = z_o = 1$ while in (2.9) it is $\tilde{f}(s,0) = z_o \in \mathcal{C}$ in general. This fixed point z_o will be contained in all convex hulls for $k \in [0,\infty)$, according to [13]. The segment on the boundary of $co\{M_i\}$ which first meets the origin and which determines the lower bound k_{low}, is defined by two points $M_a(k) = f(V_a,k)$ and $M_b(k) = f(V_b,k)$ above and below $\overline{0z_o}$ respectively, where V_a and V_b are vertices of $\mathcal{D}$ defined as *critical vertices* in [11,12]. It would be convenient to know beforehand among the 2^n vertices in $\mathcal{D}$, which will be the critical ones.

The originality in our implementation of this procedure is based on the fact that we guess two vertices V_a and V_b to be the critical ones.

Guessing of the critical vertices is done by determining the widest angle measured among all M_i above and M_j below $\overline{0z_o}$, which is defined as the *worst segment* and is denoted by $\overline{M_aM_b}$. Once $\overline{M_aM_b}$ has intersected the origin, we verify if our choice for M_a and M_b was correct. If this is the case k_{low} is obtained, otherwise another pair of vertices is guessed as being critical and the procedure is repeated. Practice shows that only a few such iterations are necessary to obtain the critical vertices and k_{low}. Therefore computational savings result since most of the time we work with only two vertices. The whole procedure can be defined in three steps, starting from the highest possible k_{max} and decreasing until we obtain k_{low}. A searching range $[k_{min}, k_{max}] \subset [0,\infty)$ can be easily selected at the outset.

Step 1 (Selection of worst segment)

Define

$$M_a(k) \overset{\text{def}}{=} \{M_i(k) = f(V_i,k) \mid \alpha = \max_i \alpha_i \quad ; \quad \alpha_i = \widehat{M_i0z_o} \geq 0 \quad ; \quad V_i \in \mathcal{D}\}$$

and

$$M_b(k) \overset{\text{def}}{=} \{M_j(k) = f(V_j, k) \mid \beta = \min_j \beta_j \quad ; \quad \beta_j = \widehat{M_j 0 z_o} < 0 \quad ; \quad V_j \in \mathcal{D}\}$$

where the angles α_i and β_j are measured in the interval $(-180^o, 180^o]$. We now define $\overline{M_a M_b}$ as the worst segment for a particular k (see figure 2.5).

Step 2 (k-search until segment intersects origin)

A zero crossing and/or minimization routine is used to calculate the k for which the angle $\theta_l = (180^o - \beta + \alpha)$ becomes zero as in [11,12]. At this $k = k_1$ the segment $\overline{M_a M_b}$ intersects the origin (see figure 2.6).

Step 3 (Verify that $\overline{M_a M_b}$ is worst segment)

For the new $k = k_1$, we should check that $\overline{M_a M_b}$ is still the worst segment as defined in Step 1. This is done by checking that all other points M_i are in the region bounded by the line $\overline{M_a M_b}$ and which contains z_o. If this is the case then $k_{low} = k_1$. Otherwise we find a new worst segment and return to Step 2, until k_{low} is found (segment $\overline{M_1 M_4}$ in figure 2.6).

There are several comments to be made at this point.

(i) The criterion adopted at Step 1 in selecting the *worst* segment is not always the best one can do but results in efficient computation. This can be seen in figure 2.5 where $\overline{M_2 M_6}$ is actually further away from $\overline{0 z_o}$, although $\overline{M_2 M_5}$ has been selected as the *worst*. However when decreasing k to k_1, the whole configuration of points may change, so the sophistication of the criteria adopted is not important at this stage. Nevertheless at Step 3 for $k = k_1$ our criteria will evaluate correctly if there is a segment that already crossed the origin for $k < k_1$.

(ii) There are three problems that can occur at Step 2. One is when the segment "skips" the origin as k decreases and then both M_a and M_b remain above (or below) line $\overline{0z_o}$. In this case there will be no k for which θ_l goes to zero (see figure 2.7). The second one occurs when the segment "skips" the origin but latter while decreasing k crosses again $\overline{0z_o}$ as in figure 2.8. For this case the function $\theta_l(k)$ will be discontinuous at the zero crossing (angles measured in $[0^o, 360^o]$). Both problems can be solved by finding the k at which the segment goes above (or below) line $\overline{0z_o}$ and for that value start again Step 1, so that other segment will be selected. The last problem is really generic in zero crossing (minimization) routines and has to do with missing the root (or minimum) of $\theta_l(k)$ because of a large step size. The only solution is to decrease the search step δk.

(iii) In Step 3, while we search for a worst segment we also check if some of the points are close enough to the origin.This is done without additional computations and if this is so and a worst segment has not been found, we directly determine k_m, the worst perturbation being at a vertex of $\mathcal{D}$. Also, when starting Step 1 if we find for $k = k_{max}$ the worst segment to have a negative θ_l, we have that $co\{M_i\}$ does not reach the origin and $k_m > k_{low} > k_{max}$. Therefore no more calculations are necessary.

(iv) The most costly computation in the search for k_{low} is the calculation of the images of all 2^n vertices of the hypercube for a given k. This has to be repeated for many different k's as we search for the zero crossing of each segment. In the procedure given in this section, this search is made only over one single segment at Step 2 most of the time, which explains the increase in speed. However this algorithm still requires $O(2^n)$ computations for the lower bound. Possibilities for obtaining a polynomial time algorithm are currently being investigated.

2.3.3 Upper bound search

From subsection 2.3.2 and in the independent parameter case, we obtain k_{low} and either a single vertex V for which this bound is achieved, or a pair (V_a, V_b) of critical vertices such that $\overline{M_a M_b}$ intersects the origin at $k = k_{low}$. We then distinguish the following cases. The first situation is defined in [11,12] as Case 1 corresponding to the worst perturbation being a vertex V. In the second situation $\overline{V_a V_b}$ are vertices of an edge of the hypercube $\mathcal{D}$ defined as Case 2[11,12] and the worst perturbation is located at some point in the edge $\overline{V_a V_b}$, which is easy to determine. In both cases $k_m = k_{low}$ and no further computations are necessary.

Finally when both critical vertices V_a and V_b are not over the same edge of $\mathcal{D}$ we obtain Case 3[11,12]. In this case $k_{low} < k_m$ and we need to search for k_{upper}.

In the related parameter case we have the following modifications. Case 1 is obtained if vertex V also satisfies constraints (2.7). Case 2 is obtained if vertices V_a and V_b and the segment $\overline{V_a V_b}$ satisfy constraint (2.7) and $\overline{V_a V_b}$ is an edge of $\overline{\mathcal{D}}$. In these cases $k_m = k_{low}$ and no further computations are necessary. Case 3 is obtained in the remaining situation and $k_m > k_{low}$.

The algorithms in Case 3 for independent and related parameters are next described separately.

2.3.3.1 Independent parameters

As explained in section 2.2, the upper bound is found by increasing $k \in [k_{low}, k_{max}]$ until the map of an edge of the hypercube intersects the origin. Defining m as the number of coordinate changes between both critical vertices, we know $m > 1$ according to the above. All paths connecting V_a and V_b consisting of m edges are called "minimal" and will be denoted by $MP(V_a, V_b)$. As in [11,12] we consider edges contained in these paths in the search for k_{upper}.

To this end we classify (see Appendix) all pair of vertices (V_i, V_j) differing in only one coordinate (edges) and belonging to all minimal paths $MP(V_a, V_b)$, to pursue the search for k_{upper}. The main difference with searching for the lower bound is that we are not dealing with the expansion of a convex set anymore, which makes this a more difficult task.

The procedure is the following:

Step 1 (Determine set of edges)

Determine all edges $\overline{V_iV_j}$ that belong in minimal paths $MP(V_a, V_b)$.

Step 2 (Selection of worst edge)

From the above set, select the edge $\overline{V_\alpha V_\beta}$ whose image $\overline{M_\alpha M_\beta}$ at $k = k_{low}$ forms the widest angle γ with respect to the origin.

Step 3 (k-search until edge intersects origin)

By using the same routines as in subsection 2.3.2, we find k_1 for which $\theta_u = \gamma(k_1) - 180^o$ becomes zero. This is taken as $k_{upper} = k_1$.

Note that a re-check of the worst edge as in Step 3 of last section is meaningless here due to the nonconvexity of the set of edges defined in Step 1. When there is no intersection of the images of any edge in a critical path with the origin we set $k_{upper} = k_{max}$.

2.3.3.2 <u>Related parameters</u>

To determine in this case an upper bound, we need to find $k = k_{upper}$ for which the mapping of a constrained edge of $\mathcal{D}$ satisfying (2.7) intercepts the origin. To this end we first increase k starting at k_{low} until a segment $\overline{M_{ac}M_{bc}}$ joining the mapping of two vertices of a constrained edge $\overline{V_{ac}V_{bc}}$ goes through the origin at

$k = k_{int}$. This is only a necessary condition for the crossing of the image of $\overline{V_{ac}V_{bc}}$ through the origin. As in [13] we can define a "generalized constrained edge" (gce) as the smaller hyperface of $\overline{\mathcal{D}}$ that contains all $MP(V_{ac}, V_{bc})$. By finding the convex hull of this gce and increasing k from k_{int} until this hull crosses the origin, while simultaneously checking that the segment $\overline{M_{ac}M_{bc}}$ goes through the origin an odd number of times, we have a sufficient condition for the above and thus obtain k_{upper}. The idea is to establish that M_{ac} and M_{bc} are in different components of the image of the gce which are defined by the path of the origin through it.

The need for such complication can be easily seen in figures 2.9 through 2.11. We observe that if the origin goes only through the hull (figure 2.9), or only across segment $\overline{M_{ac}M_{bc}}$ (figure 2.10) or goes through the hull but an even number of times through this segment (figure 2.11), we will not have any guarantee that the line $f(V_l) = M_l$ has been crossed. The algorithm is:

Step 1 (Selection of constrained segment)

Define the set of segments joining the images of each pair of vertices which define an edge of the constrained set $\mathcal{D}$ satysfying (2.7). The worst segment $\overline{M_{ac}M_{bc}}$ over this set is determined with the same criterion as in Step 1 of 2.3.3.1, but now at $k = k_{int} \geq k_{low}$.

Step 2 (Convex hull of gce completely crosses origin)

Determine the convex hull of the image of the gce corresponding to the above constrained edge $\overline{V_{ac}V_{bc}}$. Increase k until the whole hull intercepts the origin while checking the number of crossings of $\overline{M_{ac}M_{bc}}$.

This last procedure is similar to the one that searches for k_{low}, the difference being that now we increase k until the hull completely crosses the origin.

2.3.4 Refinements of the bounds

This is acomplished by subdividing the domains and repeating the calculations over each new set of hypercubes. An upper and lower bound for this set of domains is defined as in section 2.2, and the procedure is repeated until $(k_{upper} - k_{low})$ is below a given error tolerance. The subdivision of the hypercubes is explained separately for the independent and related parameter cases.

2.3.4.1 Independent parameters

As addressed in [11,12] to obtain a strictly increasing sequence of lower bounds we should subdivide in a way so that V_a and V_b defined above, remain in different hypercubes. In general there is more than one way to achieve this. The criterion adopted in [11,12] is that the cut should be made over the coordinate that has been subdivided the least number of times. This is, by defining both vertices as $V_a = [a_1, \ldots, a_n]^t$ and $V_b = [b_1, \ldots, b_n]^t$, the set of coordinate changes between them is $\mathcal{U} \overset{\text{def}}{=} \{1 \leq i \leq n \mid (V_a - V_b)_i \neq 0\}$. The coordinate along which we will cut is then $i^* \overset{\text{def}}{=} \{i \mid (V_a - V_b)_i \leq (V_a - V_b)_j \quad ; \quad i, j \in \mathcal{U}\}$. This avoids the problem of getting into very narrow and long subdomains which can decrease the convergence speed.

2.3.4.2 Related parameters

The procedure here is the following. If both critical vertices V_a and V_b satisfy the constraints in (2.7), we partition so that both belong in different domains. If only one is constrained we subdivide so that the other vertex is eliminated from the new set of domains. If none of the critical vertices satisfy (2.7) we try to eliminate both by means of the partition.

Each partition is always made along the coordinate axis which have the greatest number of equality constraints, to reduce in this way the remaining volume. Also as in 2.3.4.1 we preserve the proportions of the hypercube avoiding narrow and long subdomains. All of the above result in increased computational speed. The following steps will explain this procedure.

Step 1

Determine if both vertices V_a and V_b meet the constraints. If this is the case, find all sets of constrained coordinates in a minimum path joining both V_a and V_b. In other words, define each set of constrained coordinates $\mathcal{A}_i \overset{\text{def}}{=} \{(i_1, \ldots, i_{m(i)}) \mid \bar{\delta}_{i_1} = \ldots = \bar{\delta}_{i_{m(i)}}\}$ for $i \in \mathcal{U}$ and $\mathcal{U}$ as in 2.3.4.1 and consider the set $\bar{\mathcal{U}} \overset{\text{def}}{=} \{\mathcal{A}_i \mid (V_a - V_b)_j \neq 0 \quad ; \quad j \in \mathcal{A}_i \quad i = 1, \ldots, n\}$. Then continue with Step 4. Else go to Step 2.

Step 2

Determine if one of the two vertices, say V_a does not meet the constraints and if so which are the sets $\mathcal{A}_i$; $i \in \mathcal{U}$ of constrained coordinates which are violated by this vertex. In other words we need to find the set $\bar{\mathcal{U}} \overset{\text{def}}{=} \{\mathcal{A}_i \mid a_{i_l} \neq a_{i_k} \quad ; \quad i_l, i_k \in \mathcal{A}_i \quad i \in \mathcal{U}\}$. Then continue with Step 4. Else go to Step 3.

Step 3

In this step we know that both V_a and V_b violate the constraints. First define the sets of coordinates which both violate, i.e.

$$\bar{\mathcal{U}}_a \overset{\text{def}}{=} \{\mathcal{A}_i \mid a_{i_l} \neq a_{i_k} \quad ; \quad i_l, i_k \in \mathcal{A}_i \quad i \in \mathcal{U}\}$$

$$\bar{\mathcal{U}}_b \overset{\text{def}}{=} \{\mathcal{A}_i \mid b_{i_l} \neq b_{i_k} \quad ; \quad i_l, i_k \in \mathcal{A}_i \quad i \in \mathcal{U}\}$$

Next define either the intersection or the union depending if both $\bar{\mathcal{U}}_a$ and $\bar{\mathcal{U}}_b$ are disjoint, i.e.

$$\bar{\mathcal{U}} \overset{\text{def}}{=} \begin{cases} \bar{\mathcal{U}}_a \cup \bar{\mathcal{U}}_b & \text{if} \quad \bar{\mathcal{U}}_a \cap \bar{\mathcal{U}}_b = \emptyset \\ \bar{\mathcal{U}}_a \cap \bar{\mathcal{U}}_b & \text{otherwise.} \end{cases}$$

and go to Step 4.

Step 4

For the set $\bar{\mathcal{U}}$ obtained above we shall determine the set of coordinates $\mathcal{A}_i$ with the highest number of parameters, i.e.

$$\mathcal{L} \overset{\text{def}}{=} \{\mathcal{A}_i \mid m(i) \geq m(j) \quad ; \quad \mathcal{A}_i, \mathcal{A}_j \in \bar{\mathcal{U}}\}$$

with $m(i), m(j)$ defined at Step 1. If $\mathcal{L}$ has only one set $\mathcal{A}_{i^*}$, cut over all its coordinates and stop. Else go to Step 5.

Step 5

Finally over the set $\mathcal{L}$ find the element $\mathcal{A}_{i^*}$ with the least number of cuts as defined in subsection 2.3.4.1 and cut over all coordinates of this set, then stop.

We conclude this section by saying a few words on convergence.

2.3.4.3 Convergence

To increase the convergence rate we need mechanisms that will reduce the amount of computation over the exponentially increasing number of domains. There are several mechanisms that contribute to this.

At partition r, at the end of the procedure over the whole set of domains, we should select among the pairs (k_{low}, k_{upper}) of each domain, the new $k_{low}(r)$ and $k_{upper}(r)$ for the whole set of domains, as described in section 2.2. At the next partition we can always discard a new domain whose $k_{low} \geq k_{upper}(r)$ and no further calculations need to be made over this domain. This is the main mechanism to reduce the number of new hypercubes[11,12].

Whenever a domain is in Case 1 or 2, we do not need to continue subdividing it. This means that at each new partition this domain will not need to be recalculated and even more important, will not generate new domains.

A third mechanism is when the upper bound of a certain domain at partition r increases above the upper bound of the last partition $k_{upper}(r-1)$. We do not need to continue the calculations, being that this will not influence the choice of the new upper bound as explained in section 2.2. This reduces the calculations at each partition by effectively reducing the ranges of k's in which to conduct the search.

2.4 EXAMPLE

The lateral directional control system of a particular aircraft is analyzed. The linear model has been obtained for a Mach number of 0.6 and an altitude of 15,000 ft. as the nominal flight conditions. The plant has 4 states and 2 inputs as defined below:

$$\mathbf{x} = [\beta \quad p \quad r \quad \phi]^t \quad ; \quad \mathbf{u} = [\delta_a \quad \delta_r]^t$$

with

β :	sideslip angle
p :	roll rate
r :	yaw rate
ϕ :	roll angle
δ_a :	aileron deflection
δ_r :	rudder deflection

The rolling and yawing moment parameters will have constant but uncertain values around the nominal and appear in the system matrices as:

$$A = \begin{bmatrix} -0.186 & 0.068 & -0.998 & 0.05 \\ L_\beta & -2.782 & 1.099 & 0 \\ N_\beta & -0.110 & -0.078 & 0 \\ 0 & 1 & 0.069 & 0 \end{bmatrix} \quad ; \quad B = \begin{bmatrix} -0.7e{-3} & 0.76e{-3} \\ L_{\delta a} & L_{\delta r} \\ N_{\delta a} & N_{\delta r} \\ 0 & 0 \end{bmatrix}$$

with

$$L_\beta = -11.02 \pm 33\%$$

$$N_\beta = 5.64 \pm 11\%$$

$$L_{\delta a} = 1.48 \pm 8\%$$

$$N_{\delta a} = 0.16 \pm 43\%$$

$$L_{\delta r} = 0.19 \pm 13\%$$

$$N_{\delta r} = -0.05 \pm 14\%$$

The controller has been designed for particular values of these parameters which are supposed to be the worst ones in terms of stability according to *physical intuition.*

To analyze robust stability we open the loop between plant and controller so that the mapping function used by the program is $\det\left[I + K(j\omega)\,G(j\omega,\Delta)\right]$. This gives us a total of 8 parameters, 4 independent and 4 related by the constraints described in (2.7). The value of the Multivariable Stability margin was $k_m = 3.7$ and the worst combination of parameters:

$$L_\beta = -14.69 \quad ; \quad N_\beta = 5.02$$

$$L_{\delta a} = 1.36 \quad ; \quad N_{\delta a} = 0.094$$

$$L_{\delta r} = 0.21 \quad ; \quad N_{\delta r} = -0.048$$

The nominal system has a closed loop pole at 0.011 rad/s and for the set of parameters above we can verify that it moves 30% towards the instability bound (1).

It is important to note that the parameters N_β and $N_{\delta r}$ used to design the controller do not coincide with the worst ones mentioned above, in fact they are both in opposite sides of the uncertainty interval. Although this is not important in this case being $k_m > 1$ it shows how engineering *common sense* can be misleading.

(1) In particular this pole is unimportant even if it is unstable, being slow enough to be controlled by the pilot

2.5 APPENDIX

Assume two vertices of the hypercube defined as in 2.3.4.1 $V_x = [x_1, \ldots, x_n]^t$ and $V_y = [y_1, \ldots, y_n]^t$ and also $(x_i - y_i) \neq 0$ for $i \in \{l_1, \ldots, l_m\}$, i.e. there are $m!$ minimum paths of length m between both vertices. In this context the general formulation of the problem is:

"Given (V_x, V_y) as above, find all pairs of neighbor vertices (V_i, V_j) in all minimum paths going from V_x to V_y."

To solve this we can go through the following procedure.

Step 1

Assign a number to each vertex starting from $V_x \to 0$ to $V_y \to (2^m - 1)$. The most appropiate way to do this is by assigning a binary 0 to the upper (or lower) bounds of each parameter we find at each coordinate of V_x and proceed counting in binary until we reach V_y. The binary 1's will be the corresponding lower (upper) bounds taking V_x as the reference. Finally transform from binary to decimal. We will have a total of $2^m - 2$ vertices and $\sum_{i=0}^{m-1} \binom{m}{i}(m-i)$ edges. This is smaller than the total number of paths times the number of edges at each path $m.m!$, because there are repeated edges at each path.

Step 2

Generate Table II.1 where each entry corresponds to a vertex numbered as described above.

Step 3

The neighbor edges (V_i, V_j) to be generated are the ones going from each element in the left column to each element in the corresponding row. In this way we generate all $\sum_{i=0}^{m-1} \binom{m}{i}(m-i) = m.2^{m-1}$ edges in all minimum paths without any repetition.

Note: If we just generated all vertices in between V_x and V_y and calculated the neighbor vertices for each one of them, we would actually calculate $m.2^m$ pairs (V_i, V_j) which is exactly twice as much as we have done above.

V_x	$\longrightarrow$	2^0	2^1	$\dots$	2^{m-1}
2^0	$\longrightarrow$	(2^0+2^1)	(2^0+2^2)	$\dots$	(2^0+2^{m-1})
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
2^{m-1}	$\longrightarrow$	$(2^{m-1}+2^0)$	$(2^{m-1}+2^1)$	$\dots$	$(\sum_{m-2}^{m-1})$
(2^0+2^1)	$\longrightarrow$	$(2^0+2^1+2^2)$	$\dots$	$(2^0+2^1+2^{m-1})$	$-$
(2^0+2^2)	$\longrightarrow$	$(2^0+2^2+2^1)$	$\dots$	$(2^0+2^2+2^{m-1})$	$-$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$-$
$\sum_{i=m-2}^{m-1} 2^i$	$\longrightarrow$	$(\sum_{m-2}^{m-1} 2^i + 2^0)$	$\dots$	$(\sum_{m-2}^{m-1} 2^i + 2^{m-3})$	$-$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$-$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$-$
$\sum_{i\neq 0}^{m-1} 2^i$	$\longrightarrow$	$\sum_{i=1}^{m-1} 2^i$	$-$	$-$	$-$
$\sum_{i\neq 1}^{m-1} 2^i$	$\longrightarrow$	$\sum_{i=1}^{m-1} 2^i$	$-$	$-$	$-$
$\vdots$	$\vdots$	$\vdots$	$-$	$-$	$-$
$\sum_{i\neq m-1}^{m-1} 2^i$	$\longrightarrow$	$\sum_{i=1}^{m-1} 2^i$	$-$	$-$	$-$

Table II.1

Example:

For the three parameter case of figure 2.12 the table will take the form:

$0(V_1)$	$\longrightarrow$	$1(V_2)$	$2(V_3)$	$4(V_5)$
$1(V_2)$	$\longrightarrow$	$3(V_4)$	$5(V_6)$	–
$2(V_3)$	$\longrightarrow$	$3(V_4)$	$6(V_7)$	–
$4(V_5)$	$\longrightarrow$	$5(V_6)$	$6(V_7)$	–
$3(V_4)$	$\longrightarrow$	$7(V_8)$	–	–
$5(V_6)$	$\longrightarrow$	$7(V_8)$	–	–
$6(V_7)$	$\longrightarrow$	$7(V_8)$	–	–

Table II.2

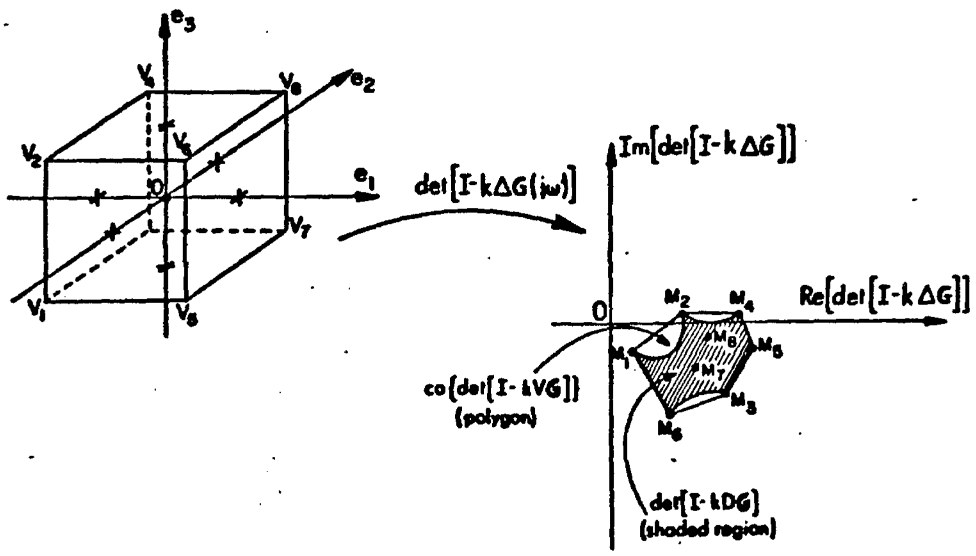


Figure 2.1: Three dimensional parameter space and complex image.

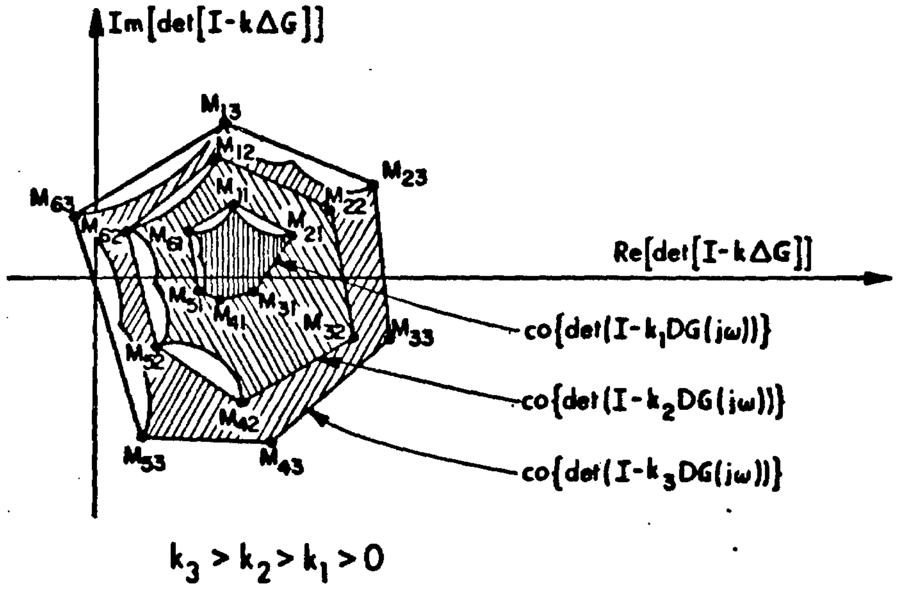


Figure 2.2: Hypercube's image for different values of k.

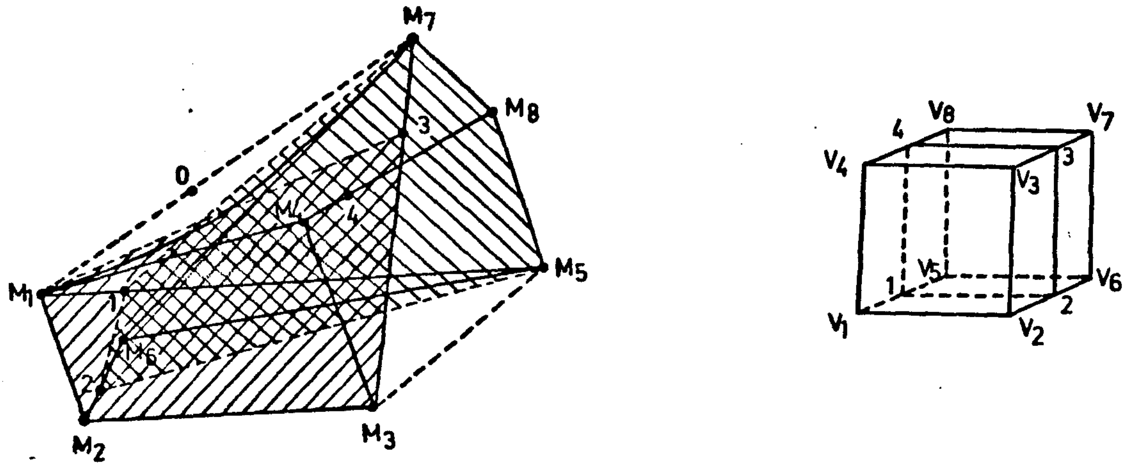


Figure 2.3: Hypercube's partition.

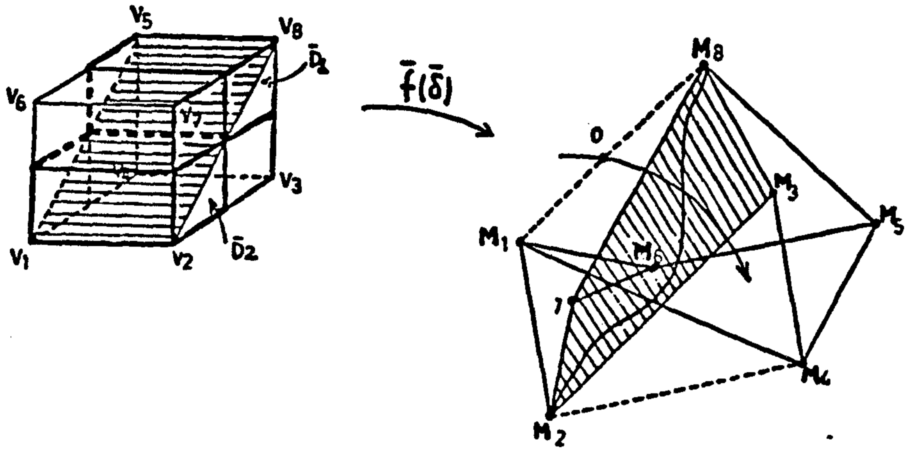


Figure 2.4: Constrained (diagonal plane) and generalized (cube) sets of parameters.

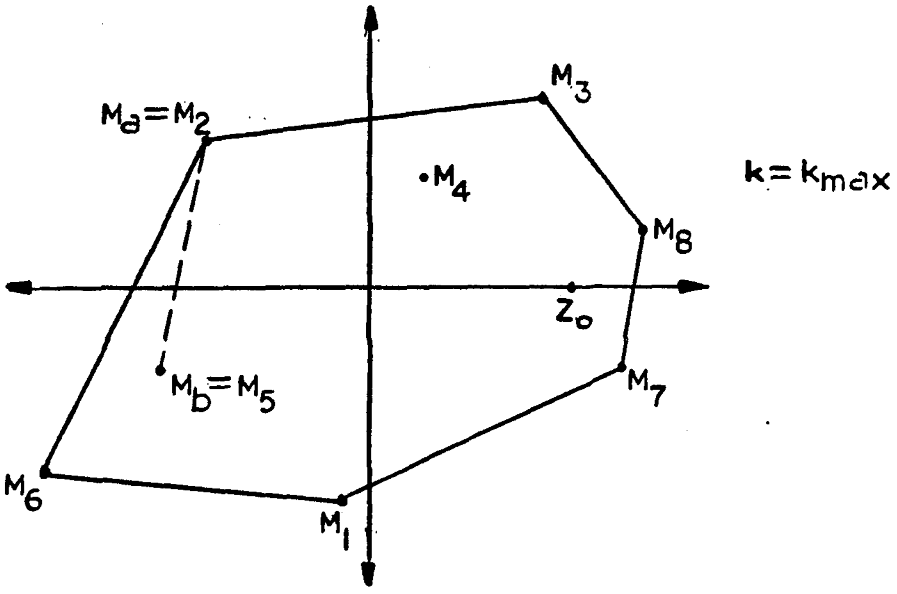


Figure 2.5: Convex hull of a general hypercube and choice of critical vertices (*worst* segment $\overline{M_aM_b}$).

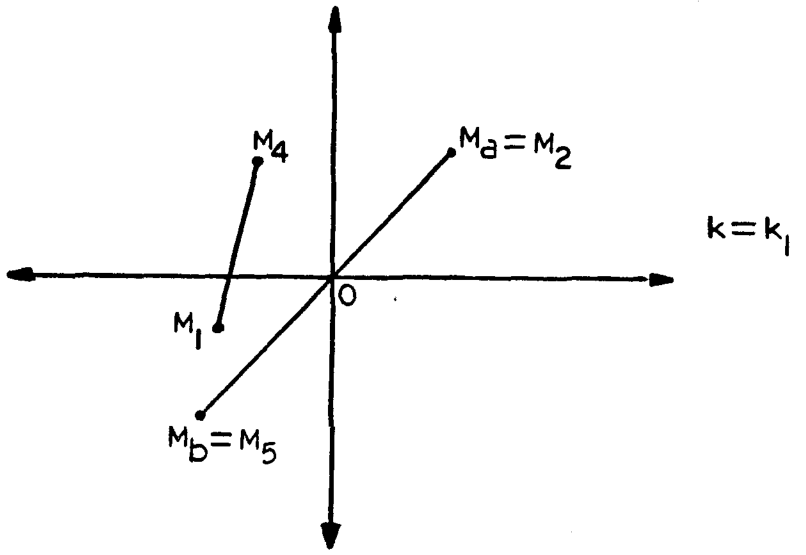


Figure 2.6: Last choice of critical vertices ($\overline{M_aM_b}$) and new one ($\overline{M_1M_4}$).

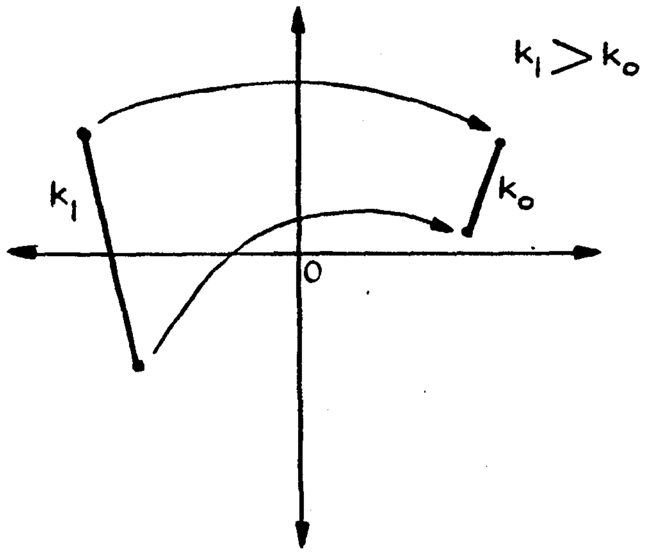


Figure 2.7: "Skipping" of the origin by the image of an hypercube's edge.

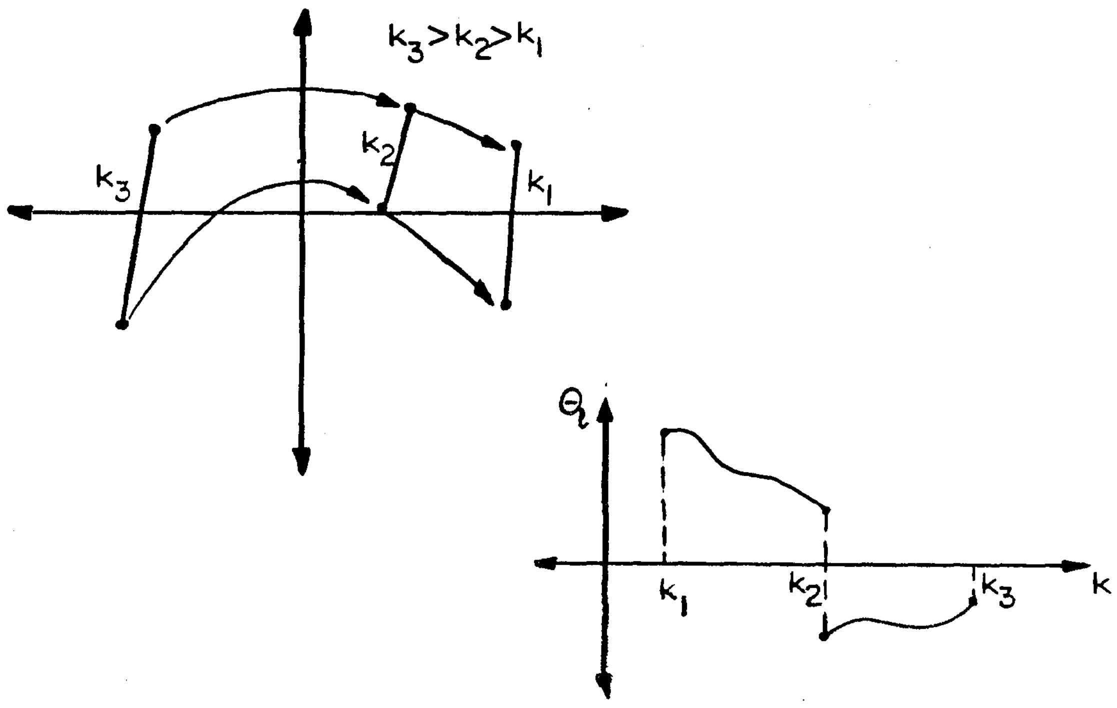


Figure 2.8: Discontinuity of the angle $\theta_l(k)$.

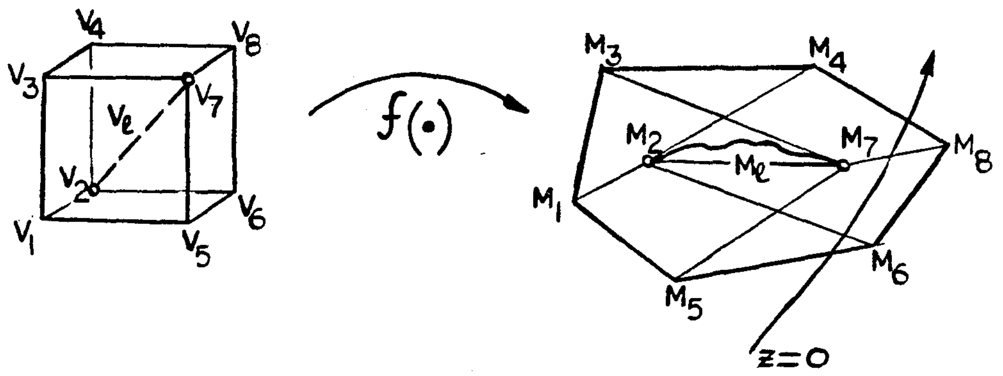


Figure 2.9: Origin crosses convex hull of generalized constrained edge but avoids segment $\overline{M_2M_7}$.

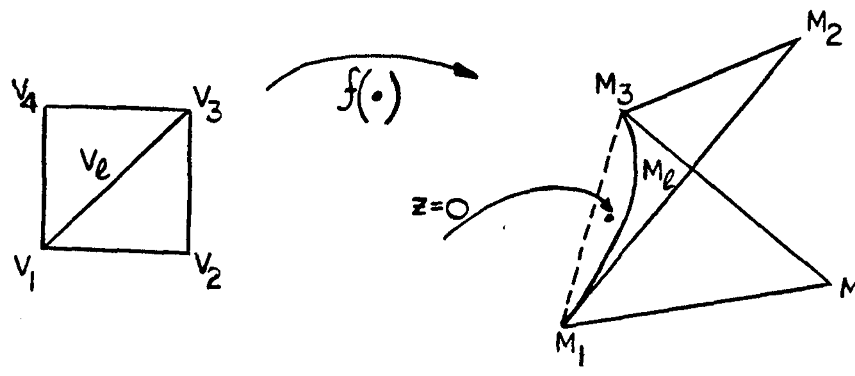


Figure 2.10: Origin crosses $\overline{M_1M_3}$ but not completely the convex hull of generalized constrained edge.

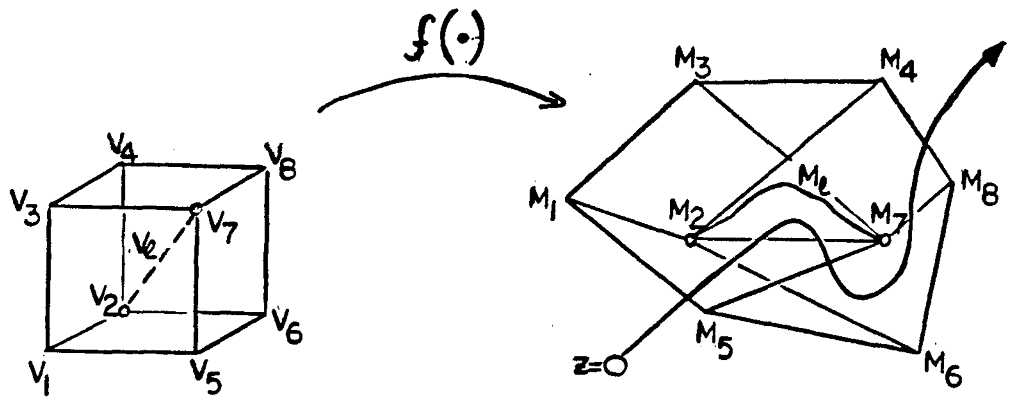


Figure 2.11: Origin crosses completely the convex hull of generalized constrained edge, an even number of times $\overline{M_2M_7}$, but not M_l.

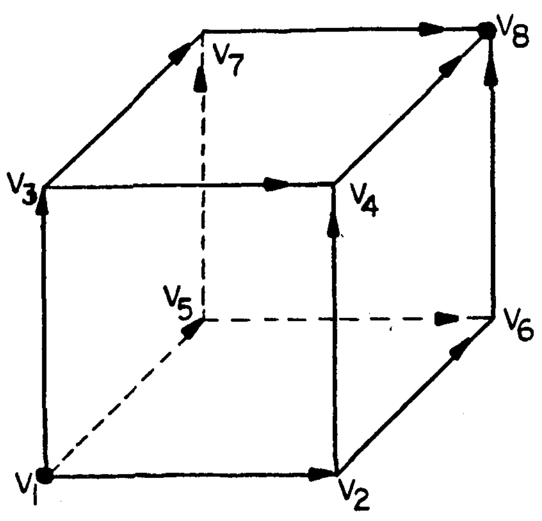


Figure 2.12: Example for defining neighbor vertices in all minimal paths.

Chapter 3

Eliminate frequency search: Theory and implementation

3.1 INTRODUCTION

In this chapter we propose a new method for computing k_m for real parametric uncertainty. By using the well-known Routh-Hurwitz stability criterion[32,33] we transform the search over all frequencies involved in the computation of k_m to a finite number of real-valued conditions that can be checked by using the method of [11-13]. Moreover the fact that these conditions are real-valued entails several simplifications in the algorithm and a significantly faster computation of $k_{min} \overset{\text{def}}{=} \min_{\omega} k_m(\omega)$.

The contents of this chapter are as follows. In section 3.2, we present the main result and in section 3.3, we discuss an example which has been worked out with both the new algorithm and the one in Chapter 2 which demonstrates the previous points.

3.2 MAIN RESULTS

3.2.1 Theory

The algorithmic procedure described in Chapter 2 allows the computation of the Multivariable Stability Margin $k_m(\omega)$ for real parametric uncertainty at each fixed frequency ω. Therefore a frequency search for computing

$$k_{min} = \min_{\omega} k_m(\omega) \tag{3.1}$$

to verify that $k_{min} > 1$ and thus check robust stability, is necessary.

In this chapter such a search is avoided by using the well known Routh-Hurwitz stability criteria[32,33]. The resulting algorithm is applicable to both the independent and related parameter cases.

Consider the closed loop characteristic polynomial $f(s;\delta)$ as a function of the Laplace variable s and the unknown parameter vector $\delta = [\delta_1, \ldots, \delta_n]^t$. Thus

$$f(s;\delta) = s^n + a_1(\delta)s^{n-1} + \ldots + a_{n-1}(\delta)s + a_n(\delta) \tag{3.2}$$

We assume that the functions $a_i(\delta)$, $i = 1, \ldots, n$ are multivariate rational functions in the δ_i's bounded for $\delta \in \mathcal{D}$. This assumption is usually satisfied by most systems of interest, but in more general cases our technique can be applied by first approximating the $a_i(\delta)$ by multivariate rational functions on the hypercube $\mathcal{D}$.

Consider now the Routh array for $f(s;\delta)$, and by means of it define the functions $g_i(\delta)$, $i = 1, \ldots, n$

$\mathbf{s}^n$	1	a_2	a_4	$\ldots$
$\mathbf{s}^{n-1}$	$a_1 = g_1(\delta)$	a_3	a_5	$\ldots$
$\mathbf{s}^{n-2}$	$b_1 = g_2(\delta)$	b_2	b_3	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathbf{s}^0$	$g_n(\delta)$	$\ldots$	$\ldots$	$\ldots$

In general the coefficient of s^n is $a_0(\delta)$ whose positiveness can be tested at the beginning; thus we can assume without loss of generality that $a_0(\delta) = 1$. Also note that all entries in the array are multivariate rational functions in the δ_i's. Therefore we can write:

$$g_i(\delta) = \frac{n_i(\delta)}{d_i(\delta)} \qquad i = 1, \ldots, n \tag{3.3}$$

and define

$$f_i(\delta) = n_i(\delta)d_i(\delta) \qquad i = 1, \ldots, n \tag{3.4}$$

where the $n_i(\delta), d_i(\delta)$ are multivariate polynomials in the δ_i's. From the Routh-Hurwitz stability criterion we obtain:

Lemma 3.2.1:

The closed-loop system is robustly stable if and only if

$$f_i(\delta) > 0 \quad \forall \delta \in \mathcal{D} \quad \text{and} \quad i = 1, \ldots, n \tag{3.5}$$

Proof

For a fixed $\delta \in \mathcal{D}$, $f_i(\delta) > 0$, $i = 1, \ldots, n$ is equivalent with $g_i(\delta) > 0$, $i = 1, \ldots, n$ since $d_i(\delta) \neq 0$ from the boundedness of the $a_i(\delta), i = 1, \ldots, n$ for $\delta \in \mathcal{D}$. Therefore from the Routh-Hurwitz stability criterion, (3.5) is a necessary and sufficient condition for robust stability.

■

The Multivariable Stability Margin k_{min} defined in (3.1) can be computed as follows:

Theorem 3.2.1:

Define

$$k_{mi} \overset{\text{def}}{=} \min_{\delta \in \mathcal{D}} \{k \in [0, \infty) \mid f_i(k\delta) = 0 \quad \text{for} \quad \text{some} \quad \delta \in \mathcal{D}\} \tag{3.6}$$

then

$$k_{min} = \min_{i=1,\ldots,n} k_{mi} \tag{3.7}$$

Proof

For $k = \min_{i=1,\ldots,n} k_{mi}$ there exists by assumption a $\delta^* \in \mathcal{D}$ such that $f_{i^*}(k\delta^*) = 0$ for some $i^* \in \{1, 2, \ldots, n\}$.

Lemma 3.2.1 implies that for the parameter values $k\delta^*$ the closed loop system is unstable and from definition (1.1) we obtain

$$k_{min} \leq \min_i k_{mi} \tag{3.8}$$

Conversely, for $k = k_{min}$ there exists $\delta^* \in \mathcal{D}$ such that for $k_{min}\delta^*$ the closed loop system is unstable. This implies that for some i^*,

$$f_{i^*}(k_{min}\delta^*) \leq 0 \tag{3.9}$$

Note next that $f_i(0) > 0 \; \forall i = 1, \ldots, n$ where $\delta = 0$ is the nominal parameter vector. This follows by nominal closed loop stability, which is a prerequisite for checking robust stability.

Since $f_i(\delta)$ are continous real valued functions of δ, $f_i(k_{min}\mathcal{D})$ must be an interval on the real axis, and therefore (3.9) implies

$$k_{min} \geq k_{mi^*} \quad \Longrightarrow \quad k_{min} \geq \min_i k_{mi} \tag{3.10}$$

From (3.8) and (3.10) we obtain (3.7) and the proof is complete.

■

We remark that each of the k_{mi}'s can be computed by using the procedure for computing $k_m(\omega)$ at a particular ω in [11-13] and which was outlined in section 2.2.

It should be clear that the calculation of $k_m(\omega)$ for each ω is now replaced by n such calculations, where n is the degree of the closed loop characteristic polynomial.

This offers a clear advantage over the previous algorithm since accurate computation of k_{min} might require an extensive frequency search, and indeed given the fact that $k_m(\omega)$ can be a discontinous function[13] of ω.

The parameter combination and the frequencies where robust stability is worse or even fails can be obtained as follows. Let i^* be such that $k_{min} = k_{mi^*}$ in (3.7). As a byproduct of the computation of k_{mi^*} we obtain $\delta^* \in \mathcal{D}$ such that $f_{i^*}(k^*\delta^*) = 0$. Then δ^* satisfies (see (3.2)):

$$f(j\omega^*; k_{min}\delta^*) = 0 \quad \text{for} \quad \text{some} \quad \omega^* \tag{3.11}$$

and $f(j\omega; k\delta) \neq 0$ for $k < k_{min}$, $\delta \in \mathcal{D}$ and $\omega \in [0, \infty)$. Thus δ^* is a worst parameter combination. Solving the polynomial equation (3.11) in ω for its positive real roots, we obtain the frequencies at which robust stability first fails.

Another important point is that the functions $f_i(\delta)$ are real valued. In the following we show how this can be taken advantage of in order to greatly expedite the computation of k_{min}.

3.2.2 Algorithmic considerations

The main procedures in the algorithm of [11-13] (see Chapter 2) are the computation of lower and upper bounds for $k_m(\omega)$ and the subdivision of the parameter hypercube to improve these bounds.

The following simplifications in these procedures for computing k_{mi} result from the real valuedness of the functions $f_i(\delta)$.

First we extend the parameter hypercube $\mathcal{D}$ to $\bar{\mathcal{D}}$ by introducing fictitious parameters as in (2.5). The mapping functions $f_i(\delta)$ are transformed accordingly to multilinear functions $\bar{f}_i\left(\bar{\delta}\right)$, $\bar{\delta} \in \bar{\mathcal{D}}$ (see subsection 2.2.2).

Step 1 (Computation of the lower bound)

A lower bound l_i on k_{mi} is found as the minimum k for which a vertex of the hypercube $k\bar{\mathcal{D}}$ is mapped by $f_i(\cdot)$ onto zero.

The lower bound on k_{mi} is obtained as the minimum value of k for which $0 \in co\left\{\bar{f}_i\left(k\bar{V}\right)\right\}$, where $\bar{V}$ is the set of vertices of $\bar{\mathcal{D}}$. Note that $co\left\{\bar{f}_i\left(k\bar{V}\right)\right\}$ is a real interval because of the real valuedness of $f_i(\delta)$ with ends being images of some vertices of $k\bar{\mathcal{D}}$.

Step 2 (Computation of the upper bound)

An upper bound u_i on k_{mi} is found as the minimum k for which a constrained vertex of the hypercube $\bar{\mathcal{D}}$ is mapped onto zero.

That $u_i \geq k_{mi}$ is clear since by definition, constrained vertices of $\bar{\mathcal{D}}$ correspond to the actual vertices of $\mathcal{D}$. In the procedure of [13] an upper bound is found as the minimum k for which the image of a constrained edge or generalized constrained edge cross the origin. These cases all reduce here to checking for constrained vertices of $\bar{\mathcal{D}}$ mapping onto the origin.

Step 3 (Refinements of the bounds)

This procedure remains unchanged with respect to the algorithms of [11-13]

Thus in the computation of the lower and upper bounds on k_{mi}, we need only check the vertices of $\bar{\mathcal{D}}$ and not be concerned with the images of edges or generalized edges "skipping" the origin, which accounts for the complexity and the major part of the computations in the program. In the algorithm of this paper we need only to compute the smallest positive real root of the real valued functions $\bar{f}_i\left(k\bar{V}_j\right)$ in k, where $\bar{V}_j$ are the vertices of $\bar{\mathcal{D}}$.

3.3 EXAMPLE

In this section we demonstrate the method for computing k_m of section 3.2 by means of the following example.

Suppose that the closed loop characteristic polynomial is given by:

$$f(s;\delta) = \mathbf{s}^4 + \mathbf{s}^3[p_1^3 p_2] + \mathbf{s}^2[p_1^2 p_2^2 p_3] + \mathbf{s}[p_1 p_2^3 p_3^2] + [p_3^3] \tag{3.12}$$

The independent parameters p_1, p_2, p_3 and their variation is defined around the nominal values as follows,

$$
\begin{aligned}
p_1 &= 1.4 + k\delta_1 \quad ; \quad -0.25 \leq \delta_1 \leq 0.25 \\
p_2 &= 1.5 + k\delta_2 \quad ; \quad -0.20 \leq \delta_2 \leq 0.20 \\
p_3 &= 0.8 + k\delta_3 \quad ; \quad -0.20 \leq \delta_3 \leq 0.20
\end{aligned}
\tag{3.13}
$$

It is easy to see that all the coefficients of (3.12) are positive for the set of parameters defined in (3.13) and $k = 1$. Applying now the Routh-Hurwitz procedure, we obtain:

$\mathbf{s}^4$	1	$p_1^2 p_2^2 p_3$	p_3^3
$\mathbf{s}^3$	$p_1^3 p_2$	$p_1 p_2^3 p_3^2$	0
$\mathbf{s}^2$	$\frac{p_2^2 p_3 \left(p_1^4 - p_3\right)}{p_1^2}$	p_3^3	$-$
$\mathbf{s}^1$	$\frac{p_3^2 p_1}{p_2} \frac{\left[p_1^4 \left(p_2^4 - 1\right) - p_2^4 p_3\right]}{\left(p_1^4 - p_3\right)}$	0	$-$
$\mathbf{s}^0$	p_3^3	$-$	$-$

We should now find the smallest k that meets one of the following constraints with equality,

$$f_1 = p_1^3 p_2 > 0 \tag{3.14}$$

$$f_2 = p_3 \left(p_1^4 - p_3\right) > 0 \tag{3.15}$$

$$f_3 = p_1 p_2 \left[p_1^4 \left(p_2^4 - 1\right) - p_2^4 p_3\right] \left(p_1^4 - p_3\right) > 0 \tag{3.16}$$

$$f_4 = p_3^3 > 0 \tag{3.17}$$

where we have simplified all parameters with even order exponents. Conditions (3.14) and (3.17) are trivial to check. For both of them the smallest k is $k_{m4} = 4$ for which p_3 vanishes. For (3.15) and (3.16) we used the computer program implementing the algorithm of section 3.2 and found $k_{m2} = 1.5$ and $k_{m3} = 1.09$ respectively. Therefore $k_m = 1.09$ and this means that the system is robustly stable for the set of parameter variations described above. The worst parameter combination is outside the region described in (3.13) and it was found to be $p^* = [1.128 \quad 1.282 \quad 1.018]$ corresponding to condition (3.16) being equal to zero. For this parameter combination the closed loop characteristic polynomial has a pair of poles at $s = \pm j1.1$. The calculation time for all four conditions was in the order of seconds. For comparison the same example was solved by using the algorithm in [14]. We observed that the computation of k_m for certain frequency points ω was well in the order of minutes.

Chapter 4

Applications to coefficient perturbed polynomials

4.1 INTRODUCTION

As mentioned in subsection 1.3.3, there are basicallly two approaches in the robust analysis of uncertain polynomials. The first one gives a *qualitative* answer to the question of robust stability of the system[20,23–26] while the second offers a *quantitative* answer in terms of k_m or μ[11–15,21,22], although any of these two approaches can be easily reduced to the other.

The material in Chapters 2 and 3 falls into this last category and takes into consideration the most general class of coefficients $c(p)$ as functions of the set of parameters p (see (1.9)) that can arise in FDLTI control systems. The setting in this chapter is similar to the one in Chapter 3, in the sense that we use the Routh-Hurwitz procedure[32,33] to transform the stability test of a single n^{th} order CLCP of the form (1.9) to the positivity test of n real equations.

In terms of the computational effort it takes to do the analysis, it is natural to expect it will increase as we treat more general classes of polynomials. In [23] the positivity of $2, 6$ or 12 equations have to be tested to analyze 2nd, 3rd or 4th order polynomials with the set of coefficients varying inside an hypercube, which is equivalent to checking the stability of $1, 2$ or 3 fixed polynomials, respectively. For a general n^{th} order polynomial of the above class, by using Kharitonov's theorem[20–23], we need to check positivity of $(4 \cdot n)$ equations instead. In [24], the number of equations to be tested will increase to $(m \cdot n)$ for the class of polynomials having odd and even coefficients varying inside two independent polytopes, were m are the extreme points of these polytopes. In [25] the class of polynomials whose coefficients are linear in the parameters the latter varying within real intervals (i.e. **all** coefficients varying inside a polytope) is treated and positivity of $(l \cdot n)$ equations as functions of a variable t need to be tested, being l the number of edges of the coefficient polytope. This t-sweep has been transformed in [26] to an eigenvalue test of a certain matrix. In [34] linear programming has been used in determining k_m for the last class of problems.

In Chapters 2 and 3 the analysis is extended to the more general cases when the coefficients c are multilinear in the parameters (independent parameter case) or polynomial in the parameters (related or repeated parameter case). In this situation we need to test positivity of n equations as functions of all parameters p through the iterative procedure explained before. Some recent results are being investigated that would reduce this test to the set of parameters contained in only the 2-dimensional faces of the hypercube for the multilinear case[35]. We can compare in figure 4.1 how the family of polynomials in (1.9) increases the complexity of the image of the parameter hypercube for each of the cases mentioned above.

The classes of polynomials with coefficients linear, multilinear or polynomial in the set of parameters p depart from the more simpler cases in the sense that no longer a finite number of tests need to be done, but instead an iterative procedure should be applied to obtain the answer, due to their increased generality. In this chapter a theoretical characterization of the class of polynomials of the form (1.9) which can be analyzed by a finite number of tests (at most over all vertices of the parameter hypercube) is given. Furthermore sufficient conditions are derived to determine when this is the case. The procedure allows in many cases to consider functions which are polynomic in the parameters as if they where multilinear, not having to create fictitious parameters as in (2.5). We will also show how as we gradually increase the functional complexity of the polynomial we will necessarily have to go through iterative procedures to test edges, faces or in general hyperfaces of the parameter hypercube. Finally two examples will be discussed which demonstrate the above results.

4.2 MAIN RESULTS

4.2.1 Equivalent conditions to check only vertices

We need to check positivity of n functions $f_j(p)$ with each parameter included in the real interval $p_i \in [a_i, b_i]$; $(i = 1, \dots, r)$, with $a_i = p_{oi} - k\delta_i$, $b_i = p_{oi} + k\delta_i$ being p_{oi} the nominal values. These intervals vary with $k > 0$, the measure of robustness to be determined, being easily reduced to the *qualitative* approach by fixing $k = 1$. We will assume that for the nominal set of parameters $p = p_o$ we have $f_j(p_o) > 0$; $(j = 1, \dots, n)$, i.e. the nominal closed loop system is stable. (1).

In the procedure described in chapter 2 we reduce both the independent and related parameter cases to testing if the image of a multilinear function intersects the origin. This allows us to use a theorem by Zadeh and Desoer[37] and prove that under this condition the mapping of the CLCP will be included in the convex hull of the mapping of the vertices(2).

In our case, since the functions $f_j(\delta)$ are all real valued, we can relax the constraint of multilinearity (or monotonicity) in the parameters. As we only need to test **positivity** of each function, we do not need to have the whole mapping of

(1) A necessary condition that is easy to check is the positivity of the coefficients $c_i(p)$, $i = 1, \dots, n$ obtaining an upper bound k_u. If $k_u < 1$ the system is not robustly stable and there is no need to continue our search. Otherwise we can use the Liénard-Chipart criteria[36] which reduces to a half the number of equations to be tested.

(2) Actually for this to be true we only need monotonicity in each parameter for all values of the other ones.

$f_j(p)$ included in the hull of the image of the vertices. The only condition will be to have the lower bound of the image of $f_j(p)$ determined by one of the vertices of the hypercube. Before we discuss the main result, we will define precisely the above.

Definition 4.1:

We are given the function $f_j : \mathcal{R}^r \longrightarrow \mathcal{R}$, the vector of upper and lower bounds of all parameters $a, b \in \mathcal{R}^r$ and the extreme points $\{a_i, b_i\}$ of the parameter p_i. A triad (f_j, a, b) is called extreme low bounded (ELB) in the parameter p_i iff

$$f_j(p)\rfloor_{p_i=a_i} \leq f_j(p) \quad \text{or} \quad f_j(p)\rfloor_{p_i=b_i} \leq f_j(p) \tag{4.1}$$

for all $p_l \in [a_l, b_l]$; $(l = 1, \ldots, r)$.

Definition 4.2:

A triad (f_j, a, b) is called multi-ELB if it is ELB for each parameter p_i ; $(i = 1, \ldots, r)$.

Remarks:

(i) Multilinear and multi-monotonic functions are particular cases of multi-ELB ones.

(ii) The ELB property will not only depend on the function f but also on the particular intervals defined by a and b. This can be seen in figure 4.2 where $f(p_1, p_2)$ is ELB in $p_1 \in [a, b]$ but not in $p_1 \in [x, b]$.

A characterization of the general requirement to check robustness by a finite number of tests is given by the following

Theorem 4.1:

For the class of polynomial functions $f_j(\cdot)$; positivity of $f_j(p)$ for $p_i \in [a_i, b_i]$; $(i = 1, \ldots, r)$ can be determined by checking only the extreme values of each parameter (vertices) if and only if the triad (f_j, a, b) is multi-ELB.

Proof:

($\Rightarrow$): Assume a parameter p_i is not ELB, then if f_j, a and b are such that $f_j(p)\rfloor_{p_i=a_i} > 0$ and $f_j(p)\rfloor_{p_i=b_i} > 0$, it is possible that for the mapping of some intermediate point $f_j(p)\rfloor_{p_i=x} < 0$; $x \in [a_i, b_i]$, thus the test of all vertices will not guarantee positivity of $f_j(p) \quad \forall p$.

($\Leftarrow$): Take an arbitrary point x_1 inside the parameter hypercube and trace through it a line parallel to one of the axes, determining two new points y_2 and z_2 at the opposite hyperfaces. Defining $x_2 = \arg\{\min[f_j(y_2), f_j(z_2)]\}$ we obtain $f_j(x_2) \leq f_j(x_1)$ because of the ELB property. Proceeding in the same way through all the axis we obtain in a finite number of steps $f_j(v) \leq \ldots \leq f_j(x_1)$ where v is a vertex of the hypercube. Since x_1 is arbitrary and proceeding in the same way for any point, we see that only vertices of the hypercube need to be checked to test positivity.

■

Before we continue we will define on an hypercube in $\mathcal{R}^n$ the *hyperface* of the set of parameters $(p_1, \ldots, p_l)$ $(l \leq r)$ as the region defined by all $p_i \in [a_i, b_i]$; $(i = 1, \ldots, l)$ with all remaining parameters at one of their extreme values. In $\mathcal{R}^3$ it reduces to the usual faces or edges of a cube.

The main result now arises naturally from the above theorem.

Corollary 4.1:

Positivity of $f_j(p)$ for all $p_i \in [a_i, b_i]$; $(i = 1, \ldots, r)$ can be determined by checking at most all the hyperfaces corresponding to all non-ELB parameters.

Proof:

We can proceed in the same way as in the necessary part of the proof of the above lemma through all the ELB parameters. We will finally arrive to different regions only defined by non-ELB parameters (hyperfaces). For this reason the low bound of the mappings of these regions will not be determined by any of their vertices, but by the whole regions themselves.

■

E.g. For a function $f_j(p_1, p_2, p_3)$ being only ELB in p_1, we should check the faces of the cube perpendicular to the p_1 axis, as seen in figure 4.3.

Remarks:

(i) An important consequence of theorem 4.1 is the fact that in the most general case, this is when (f_j, a, b) $(j = 1, \ldots, n)$ are not ELB in any parameter p_i, we are forced to check over the whole hypercube of parameters. This can be done by using the procedure in chapter 3.

(ii) To find the Multivariable Stability margin k_m we compute $k_j \overset{\text{def}}{=} \min\{k \in [0, \infty) | f_j(\delta_1, \ldots, \delta_r) = 0\}$, then $k_m = \min_j k_j$, where the search will be carried out over the regions described by theorem 4.1.

4.2.2 Relation with other works

The conditions given in the main theorem can be easily related to other work in this area[24]. To do this we need to connect the Routh-Hurwitz (R-H)[32,33] stability criteria with the Hermite-Biehler (H-B)[36] conditions used in the proof of Kharitonov's theorem[20,38]. Although both are equivalent, being necessary and sufficient conditions for the polynomial in (1.9) to have roots in the left half complex plane, it will be instructive to see both conditions from a geometric perspective.

This is

$f_j(p) > 0 \quad ; \quad j = 1, \ldots, n$ (Routh-Hurwitz criteria).

$\Longleftrightarrow$ $\Delta \arg[f(j\omega, p)] = n\pi \quad \omega \in \mathcal{R}$ increasing.

$\Longleftrightarrow$ $\Delta \arg[f(j\omega, p)] = n\pi/2 \quad \omega \geq 0$ increasing.

$\Longleftrightarrow$ $f(j\omega, p)$ crosses n consecutive times real and imaginary axis for $\omega \geq 0$.

$\Longleftrightarrow$ Interlacing property of roots of $\mathrm{Re}[f(j\omega, p)]$ and $\mathrm{Im}[f(j\omega, p)]$ (Hermite-Biehler criteria).

The procedure given in [24] evaluates the stability of the class of polynomials with independent even and odd coefficients both belonging to sets $\mathcal{E}$ and $\mathcal{O}$ respectively, by checking at most all extreme points of the whole set $\Omega = \mathcal{E} \times \mathcal{O}$. In this paper we will show that the $\mathcal{E}$-sufficiency and $\mathcal{O}$-sufficiency(3) of the extreme points of $\Omega^{[24]}$ are equivalent to at least one of the $f_j(p)$ obtained through the Routh-Hurwitz procedure to be multi-ELB. For this we first prove the following,

Lemma 4.1:

If $\mathcal{E}$ and $\mathcal{O}$ are polytopes, without loss of generality we can consider all coefficients in $\mathcal{E}$ and $\mathcal{O}$ to be linear functions of two independent sets of parameters p^e and p^o with $p_i^e \in [a_i^e, b_i^e]; i = 1, \ldots, r_e$ and $p_l^o \in [a_l^o, b_l^o]; l = 1, \ldots, r_o$. Furthermore each extreme point of the whole set of coefficients Ω will be achieved at a vertex of the hypercube defined by the family of parameters p^e and p^o and vice versa.

Proof:

It is easy to see that if a coefficient $c_j \in \mathcal{E}$ $(\in \mathcal{O})$ is nonlinear in any parameter p^e (p^o) then $\mathcal{E}$ $(\mathcal{O})$ is not a polytope, which contradicts the assumption.

Now assume we are located at an extreme point of $\mathcal{E}$ $(\mathcal{O})$ and some p_{i*}^e is strictly

(3) For the polynomial in (19) with $s = j\omega$, the real and imaginary parts will depend on the even and odd coefficients c_i respectively. Then we define a subset $\mathcal{E}' \subset \mathcal{E}$ $(\mathcal{O}' \subset \mathcal{O})$ to be $\mathcal{E}$-sufficient ($\mathcal{O}$-sufficient) when the even (odd) coefficients which determine the maximum and minimum value of the real (imaginary) part of $f(c, j\omega)$ belong to $\mathcal{E}$' ($\mathcal{O}$').

inside the interval $[a^e_{i*}, b^e_{i*}]$, then by linearity and the definition of extreme point[24], moving p^e_{i*} to a^e_{i*} and b^e_{i*} the map will be a straight line part inside and part outside the polytope $\mathcal{E}$ ($\mathcal{O}$), again a contradiction. Being the extreme points of Ω all possible combinations of the ones of $\mathcal{E}$ and $\mathcal{O}$ the same argument applies. This means that any extreme point of the coefficient space Ω will be achieved by a vertex of the parameters (p^e, p^o) hypercube.

Similarly we can prove that any vertex of the hypercube will map to an extreme point of the coefficient space Ω.

■

Lemma 4.1 plus the equivalence between R-H and H-B stability theorems will allow us to prove the following:

Lemma 4.2:

For the class of polynomials with coefficients in Ω, the extreme points will be $\mathcal{E}$-sufficient and $\mathcal{O}$-sufficient iff $\exists$ $[f_j(p), a, b]$ obtained by the R-H procedure which is multi-ELB.

Proof:

Let us define the parameters inside the intervals $p^e_i \in [p^e_{0i} - k\delta^e_i, p^e_{0i} + k\delta^e_i]$ and $p^o_j \in [p^o_{0j} - k\delta^o_j, p^o_{0j} + k\delta^o_j]$ so we can vary k continuosly until we reach the instability boundary at k_m. Assuming $\mathcal{E}$ and $\mathcal{O}$-sufficiency this boundary will be achieved at one of the extreme points of Ω and by lemma 4.1 also at a vertex p^* of the parameter hypercube. The equivalence of the R-H and H-B procedures means that at least one of the $f_j(p^*)$ will be in the neighborhood of zero which in turn means that $[f_j, a(k_m), b(k_m)]$ is multi-ELB for the intervals $a(k_m)$ and $b(k_m)$ defined above. Being both arguments necessary and sufficient, the equivalence is satisfied.

■

As proved in [24] for the particular case when both $\mathcal{E}$ and $\mathcal{O}$ are hypercubes we can consider without loss of generality, Ω and the parameter space as coincident. Then by some simplifications it can be shown that the number of extreme points (now vertices) to be checked are reduced to four, which is the Kharitonov theorem.

These simplifications will mean in our case to determine first which of the $f_j(p)$'s are multi-ELB and next find which are the relevant extreme values of some of the parameters. In the next section we will derive simple sufficient conditions for a triad (f_j, a, b) to be ELB in a parameter p_i. The same procedure will allow us to find in certain cases which is the relevant bound of that particular parameter.

4.2.3 Practical considerations

To determine when a function f_j and a set of intervals (a, b) are ELB for a particular parameter p_i we should go through a two step procedure.

The first step would be to determine if f_j belongs to a class of functions which are ELB irrespective of the intervals (a, b). For the set of polynomials considered, certain general classes of such functions are:

(i) If $f_j(p_i)$ is of odd degree, the class of monotonic (increasing or decreasing) functions, i.e.

$$\frac{\partial f_j}{\partial p_i} \geq 0 \qquad \forall p \tag{4.2}$$

or

$$\frac{\partial f_j}{\partial p_i} \leq 0 \qquad \forall p \tag{4.3}$$

(ii) If $f_j(p_i)$ is of even degree, the class of concave functions, i.e.

$$\frac{\partial^2 f_j}{\partial p_i^2} \leq 0 \qquad \forall p \tag{4.4}$$

For a polynomial, first and second derivatives are simple functions of the coefficients and in many cases these will already determine if the function f_j is ELB. For example take

$$g(p) = p_1p_2 + p_1 - p_2^2 \qquad\qquad p_1, p_2 > 0 \tag{4.5}$$

then

$$\begin{aligned} \frac{\partial g}{\partial p_1} &= 1 + p_2 > 0 \qquad \text{(monotonically increasing)} \\ \frac{\partial^2 g}{\partial p_2^2} &= -1 < 0 \qquad\qquad \text{(concave)} \end{aligned} \tag{4.6}$$

then g is multi-ELB.

Conditions (4.2), (4.3) and (4.4) are only sufficient for f_j to be ELB and in case they fail, the next step would be to test whether they hold for the particular intervals (a, b) considered. This will again lead us to check positivity (or negativity) of functions of the parameters. The whole procedure can then be repeated over this new set of equations until we can determine for which parameters is f_j ELB. Being all conditions (4.2), (4.3) and (4.4) only sufficient, if some of the positivity (or negativity) conditions are not met, the analysis will be inconclusive.

Remarks:

(i) An important consequence can be obtained from this analysis. If we can determine from (4.2) or (4.3) that a parameter p_i is monotonic, then we will only need to check a_i if it is increasing or b_i if decreasing. This will reduce the number of regions to be tested as described by theorem 4.1 and in particular will reduce the number of vertices if the theorem applies. A simple example follows. Take the CLCP:

$$f = s^3 + p_2 \cdot s^2 + p_1 \cdot s + p_0 \quad ; \quad p_i > 0 \tag{4.7}$$

By applying the Routh-Hurwitz stability criterion we need to test whether $f_1(p) = p_1p_2 - p_0 > 0$. But $f_1(p)$ is multilinear in the parameters and in particular decreasing in p_0 and increasing in p_1 and p_2. Then we only need to check the vertex $p = [b_o \quad a_1 \quad a_2]^t$, a particular subset of the four Kharitonov vertices as described in [23].

(ii) Another important consequence is that once determined the offending value of one parameter say $p_i = a_i$ as above, the procedure will be repeated over other parameters p_l but **only** at the value $p_i = a_i$. This can be seen from the following example. Take the CLCP

$$f = s^4 + p_3 \cdot s^3 + p_2 \cdot s^2 + p_1 \cdot s + p_0 \quad ; \quad p_i > 0 \tag{4.8}$$

By applying Liénard-Chipart[36] criteria we need to test only

$$f_1(p) = p_1 p_2 p_3 - p_1^2 - p_0 p_3^2 > 0 \tag{4.9}$$

From the analysis we obtain f_1 is linearly decreasing in p_0 and increasing in p_2 and concave in p_1 and p_3. This means we should fix $p_0 = b_0$ and $p_2 = a_2$ and apply the second step to check if for the particular intervals we have, the quadratic equations in p_1 and p_3 are in the increasing or decreasing region. This simplifies by the fact p_0 and p_2 are fixed, so we only need to check if

$$\frac{\partial f_1}{\partial p_1}\rfloor_{b_0,a_2} = a_2 p_3 - 2p_1 \quad \text{and} \quad \frac{\partial f_1}{\partial p_3}\rfloor_{b_0,a_2} = a_2 p_1 - 2b_0 p_3 \tag{4.10}$$

are positive or negative. This suggests checking at both combinations (a_1, b_3) and (b_1, a_3) which means only the vertices

$$p_A = \begin{bmatrix} b_0 \\ b_1 \\ a_2 \\ a_3 \end{bmatrix} \qquad p_B = \begin{bmatrix} b_0 \\ a_1 \\ a_2 \\ b_3 \end{bmatrix} \tag{4.11}$$

as suggested in [23].

(iii) Although we will reduce the number of vertices to be checked when the situation in (i) applies for some of the parameters, there can be situations in which we will unnecessarily check some extra vertices. This is because (4.2), (4.3) and

(4.4) are sufficient conditions for a triad to be ELB in a certain parameter, but not necessary.

4.3 EXAMPLES

4.3.1 Crane model

This example appeared initially in [39] and recently in [29] where some dynamic uncertainty has been added to the model and treated in the framework of parametric uncertainty. By using the CLCP of the system in this last paper we have:

$$\begin{aligned} f(s,\mu,m_l) = \mu s^6 + 3\mu s^5 + (1e-3\mu m_l + 3.116\mu + 10m_l) + \\ + (2.31e-4\mu m_l + 9.47\mu + 27.69m_l)s^3 + \\ + (1e-2m_l^2 - 8.8e-3\mu m_l + 36.06m_l + 25)s^2 + \\ + (27.69m_l - 1.5e-3\mu m_l)s + 5m_l \end{aligned} \tag{4.12}$$

being

$\mu \in [0, 13.7]$: a function of cable mass and length.

$m_l \in [50, 2395]$: mass load.

By applying the Liénard-Chipart[36] criteria to $f(\cdot)$ we obtain four polynomials $f_1(\mu, m_l)$ to $f_4(\mu, m_l)$ for which we should test positivity. The procedure in this paper has been applied to all four of them, but for brevity we will just show it for the first one only. This is,

$$f_1(\mu, m_l) = 2.31m_l + 27.69e-3m_l\mu - 0.12\mu \tag{4.13}$$

$$\text{with} \quad \begin{cases} \frac{\partial f_1}{\partial m_l} = 2.31 + 27.69e-3\mu > 0 \\ \frac{\partial f_1}{\partial \mu} = 27.69e-3m_l - 0.12 > 0 \end{cases} \tag{4.14}$$

both linear increasing functions, which means we only need to check at $\mu = 0$ and $m_l = 50$.

Similarly for f_2, f_3 and f_4 we obtained they where monotonically increasing functions of m_l and monotonically decreasing in μ. This implies we should also check f at $\mu = 13.7$ and $m_l = 50$. In fact the worst combination of parameters in the sense of robust stability is the last one for which the CLCP of the system will have roots at the imaginary axis.

In this way only two vertices have been checked to determine this result and the answer is nonconservative. In [29] instead a complete check of all edges of a polytope containing the coefficient space had to be done. Depending on how the actual coefficient space fits this polytope, either the answer will be conservative or a big number of edges will have to be tested.

4.3.2 Combined real parametric and complex dynamic uncertainty

This example consists of an uncertain plant and a controller such that the loop transfer function $L(s)$ will have an uncertain gain and right half plane pole of the form:

$$L(s) = \frac{3p_2\,(s+2)}{(s+1)(s-3p_3)} \tag{4.15}$$

with

$$p_2 \in [3,5] \qquad \text{and} \qquad p_3 \in [0,2] \tag{4.16}$$

We also want to limit the peak of the sensitivity function so we have a complex unstructured uncertainty $|p_1| < 1$. By the method described in next chapter, the robustness test reduces to check positivity of a function of the parameters but now also of the frequency ω. This function is:

$$f(\omega, p) = x\omega^4 + [9p_2^2 + 9p_3^2 x - 18p_2p_3 - 6p_2 + x]\omega^2 + 9[4(p_2^2 - p_2p_3) + p_3^2 x] \tag{4.17}$$

with $x < 1$ and $\omega \in [0, \infty)$. The first derivatives at p_2 and p_3 are

$$\begin{aligned} \frac{\partial f}{\partial p_2} &= \omega^2(p_2 - p_3 - 1/3) + 2(2p_2 - p_3) > 0 \\ \frac{\partial f}{\partial p_3} &= \omega^2(xp_3 - p_2) + (xp_3 - 2p_2) < 0 \end{aligned} \tag{4.18}$$

being in both cases only necessary to check at $p_2 = 3$ and $p_3 = 2$ because f is linearly increasing in p_2 and decreasing in p_3. In this way a single vertex test is sufficient to determine robust stability.

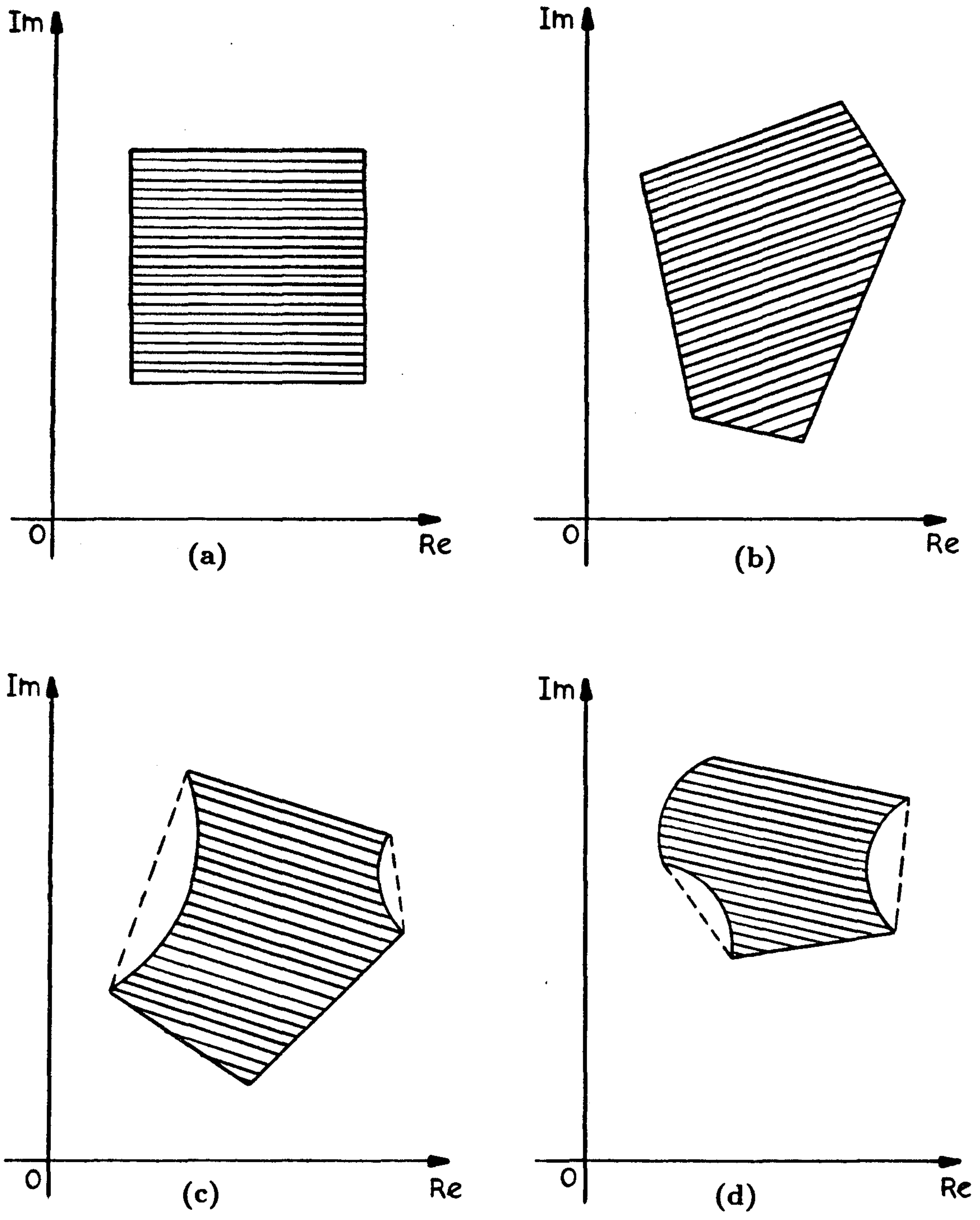


Figure 4.1: Comparison of images of different classes of uncertain polynomials. (a) Independent coefficients. (b) Coefficients linear in p. (c) Coefficients multilinear in p. (d) Coefficients polynomic in p.

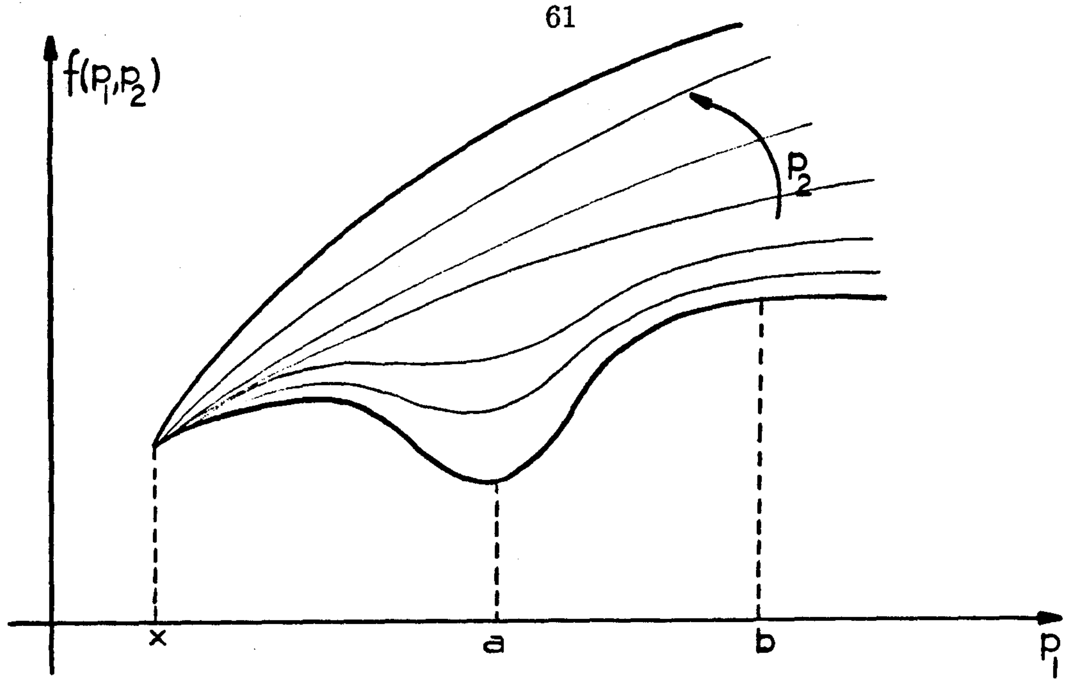


Figure 4.2: The ELB property for a function f and two sets of intervals.

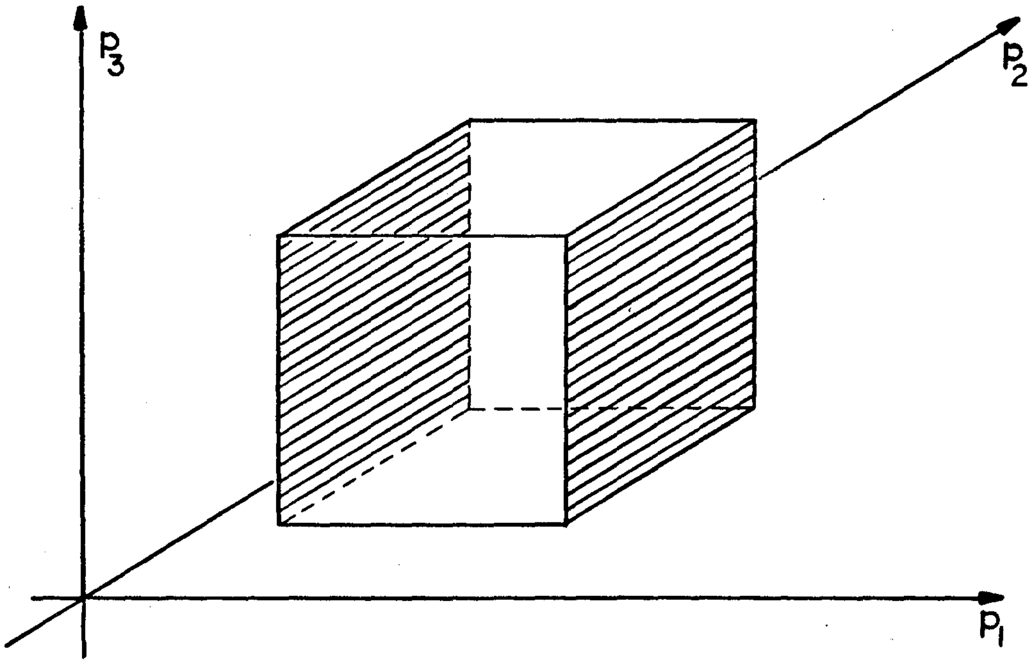


Figure 4.3: Example of regions to be checked for an ELB function in p_1.

Chapter 5

Stability margin for combined parametric and unstructured dynamic uncertainty

5.1 INTRODUCTION

In this chapter we consider the problem of Robust Stability in systems with simultaneous real uncertain parameters and unmodeled dynamics which is given by one complex block. In other words in the general uncertainty structure described by (1.2), any number of real parameters which can be repeated and one complex block are allowed.

For such uncertainty structure we define the robustness margin $r_m(\omega)$ and characterize robust stability in terms of it. The robustness margin is closely related to the multivariable stability margin given in [10]. The main contribution here is the exact computation of $r_m(\omega)$ and thus a nonconservative robustness test for the uncertainty structure considered. The computation of $r_m(\omega)$ is reduced to a finite number of multivariable stability margin calculations which can be accomplished by the algorithm described in Chapter 3. The exact calculation of $k_m(\omega)$ and $\mu(\cdot)$ for the uncertainty structure considered is also possible through an iterative procedure.

Although the uncertainty structure of any real parameters and one complex dynamic block is not the most general possible, there are many important cases that can be handled through this particular uncertainty structure. We discussed already the robust stability of systems where model uncertainty is expressed in terms of unknown real parameters and unmodeled dynamics that are lumped in one location in the feedback path. A second application is testing the robust performance of systems with real parametric uncertainty by taking the complex block allowed in (1.2) to be the performance block[16].

Yet another application of these results is the robustness of systems with real time varying parameters. A sufficient condition for robust stability can be stated in terms of the SSV of a certain constant matrix with respect to the uncertainty structure considered here[40].

Other results that consider real and complex uncertainties simultaneously are reported in [28,29,30].

In the next section the problem is precisely formulated and the robustness margin r_m is defined and related to the k_m and μ. In section 5.3 the main results are presented on the exact computation of $r_m(\omega)$ and $k_m(\omega)$ for the uncertainty structure considered. In section 5.4 we apply these results in two examples.

5.2 PROBLEM FORMULATION

We first define the uncertainty structure that is used throughout the chapter (see figure 5.1).

$$\mathbf{\Delta} \overset{\text{def}}{=} \{diag\,(\Delta_b, \Delta_p) \mid \Delta_b \in \mathbf{\Delta_b} \quad ; \quad \Delta_p \in \mathbf{\Delta_p}\} \tag{5.1}$$

$$\mathbf{\Delta_b} \overset{\text{def}}{=} \{\Delta \in \mathcal{C}^{mxm} \mid \bar{\sigma}\,(\Delta) < 1\} \tag{5.2}$$

$$\mathbf{\Delta_p} \overset{\text{def}}{=} \{diag\,(\delta_1, \ldots, \delta_i, \ldots, \delta_n) \mid \delta_i \in \mathcal{R} \quad ; \quad |\delta_i| < 1\} \tag{5.3}$$

The nominal system is given by the rational transfer matrix $M(s)$ which can be subdivided into four blocks conformably with the dimensions of Δ_b and Δ_p. We thus obtain figure 5.2. The last step allows us to work with a new system which depends on the parameters $\delta_1, \ldots, \delta_n$ in $\mathbf{\Delta_p}$. More specifically the latter is expressed by the following Linear Fractional Transformation:

$$\tilde{M}(s,\Delta_p) \overset{\text{def}}{=} M_{11} - M_{12}\Delta_p (I + M_{22}\Delta_p)^{-1} M_{21} \tag{5.4}$$

The following theorem gives necessary and sufficient conditions for robust stability and it is an easy extension of a result in [3] for the case of unstructured uncertainty.

Theorem 5.1:

The feedback system of figure 5.1 is robustly stable (i.e. it remains closed-loop stable for all $\Delta_p \in \mathbf{\Delta}_p$ and all $\Delta \in \mathbf{\Delta}_b$ if and only if

(i) $\tilde{M}(s,\Delta_p)$ is stable for every $\Delta_p \in \mathbf{\Delta}_p$.

(ii) $\bar{\sigma}\left[\tilde{M}(j\omega,\Delta_p)\right] < 1 \quad \forall \quad \Delta_p \in \mathbf{\Delta}_p \quad , \quad \omega \in [0,\infty)$.

Condition (i) in the previous theorem simply states that the closed-loop system must remain stable for all variations of the real parameters, when there is no complex uncertainty. It can be checked by evaluating

$$k_r(\omega) \overset{\text{def}}{=} \inf\{k \in [0,\infty) \mid \det[I + k\Delta_p M_{22}(j\omega)] = 0 \quad \text{for} \quad \text{some} \quad \Delta_p \in \mathbf{\Delta}_p\} \tag{5.5}$$

Evaluation of k_r is equivalent with calculating the stability margin for real parametric uncertainty and can be done by the results of Chapter 3.

Condition (ii) in theorem 5.1 states that the system resulting by fixing the real parameters at any combination allowed by (5.3) must be robustly stable for the model perturbation allowed by (5.2). In particular we obtain the following necessary conditions for robust stability.

Corollary 5.1:

The feedback system of figure 5.1 is robustly stable only if:

(i) $k_r(\omega) \geq 1 \quad ; \quad \forall\omega$

(ii) $\bar{\sigma}[M_{11}(j\omega)] \leq 1 \quad ; \quad \forall\omega$

The following theorem gives an alternative expresion for the Multivariable Stability Margin with respect to the uncertainty structure of (5.1),(5.2) and (5.3).

<u>Theorem 5.2</u>:

Let

$$\hat{k}_m(\omega) \overset{\text{def}}{=} \inf\left\{k \in [0,\infty) \mid \bar{\sigma}\left[\tilde{M}(j\omega, k\Delta_p)\right] \geq \frac{1}{k} \quad \text{for} \quad \text{some} \quad \Delta_p \in \mathbf{\Delta}_p\right\} \tag{5.6}$$

then

$$k_m(\omega) = \min\{\hat{k}_m(\omega), k_r(\omega)\} \tag{5.7}$$

<u>Proof</u>:

We calculate:

$$\begin{aligned}\det(I + k\Delta M) &= \det \begin{bmatrix} (I + k\Delta_b M_{11}) & (k\Delta_b M_{12}) \\ \\ (k\Delta_p M_{21}) & (I + k\Delta_p M_{22}) \end{bmatrix} \\ &= \det(I + k\Delta_p M_{22}) \cdot \\ &\qquad \det\left\{I + k\Delta_b\left[M_{11} - M_{12}\left(I + k\Delta_p M_{22}\right)^{-1} k\Delta_p M_{21}\right]\right\} \\ &= \det(I + k\Delta_p M_{22}) \det\left[I + k\Delta_b \tilde{M}(j\omega, k\Delta_p)\right]\end{aligned}$$

Therefore from (1.1)

$$\begin{aligned}k_m(\omega) = \inf\{k \in [0,\infty) \mid \det(I + k\Delta_p M_{22}) = 0 \quad \text{or} \\ \det\left[I + k\Delta_b \tilde{M}(j\omega, k\Delta_p)\right] = 0 \quad \text{for} \quad \text{some} \quad \Delta_p \in \mathbf{\Delta}_p \quad ; \quad \Delta_b \in \mathbf{\Delta}_b\}\end{aligned} \tag{5.8}$$

But

$$\begin{aligned}&\inf\left\{k \in [0,\infty) \mid \det\left[I + k\Delta_b \tilde{M}(j\omega, k\Delta_p)\right] = 0 \quad \text{for} \quad \text{some} \right. \\ &\qquad\qquad\qquad\qquad \left. \Delta_p \in \mathbf{\Delta}_p \quad ; \quad \Delta_b \in \mathbf{\Delta}_b\right\} \\ &\Longleftrightarrow \inf\left\{k \in [0,\infty) \mid \bar{\sigma}\left[k\tilde{M}(j\omega, k\Delta_p)\right] \geq 1 \quad \text{for} \quad \text{some} \quad \Delta_p \in \mathbf{\Delta}_p\right\} \\ &\Longleftrightarrow \inf\left\{k \in [0,\infty) \mid \bar{\sigma}\left[\tilde{M}(j\omega, k\Delta_p)\right] \geq \frac{1}{k} \quad \text{for} \quad \text{some} \quad \Delta_p \in \mathbf{\Delta}_p\right\} = \hat{k}_m\end{aligned} \tag{5.9}$$

by comparison with (5.6). Finally (5.7) is immediate from (5.8) and (5.9).

■

Next we define the robustness margin $r_m(\omega)$ with respect to the model uncertainty structure in (5.1), (5.2) and (5.3) by:

$$r_m(\omega) \overset{\text{def}}{=} \min\{\hat{r}_m(\omega), k_r(\omega)\} \tag{5.10}$$

where

$$\hat{r}_m(\omega) = \inf\left\{k \in [0, \infty) \mid \bar{\sigma}\left[\tilde{M}(j\omega, k\Delta_p)\right] \geq 1 \quad \text{for} \quad \text{some} \quad \Delta_p \in \boldsymbol{\Delta}_p\right\} \tag{5.11}$$

and $k_r(\omega)$ was defined in (5.5). The next theorem characterizes robust stability in terms of r_m in much the same way that k_m does. However, as it will be apparent in the next section, $r_m(\omega)$ is easier to calculate than $k_m(\omega)$.

<u>Theorem 5.3</u>:

The system of figure 5.1 is robustly stable for the model uncertainty structure defined by (5.1), (5.2) and (5.3) if and only if $r_m(\omega) \geq 1 \quad \forall\omega$.

<u>Proof</u>:

From the definitions of $k_m(\omega)$ and $r_m(\omega)$ observe that if

$$k_m(\omega) > 1 \qquad \Longrightarrow \qquad r_m(\omega) > 1 \tag{5.12}$$

and if

$$k_m(\omega) \leq 1 \qquad \Longrightarrow \qquad r_m(\omega) \leq 1 \tag{5.13}$$

Therefore if the system is robustly stable, then $k_m(\omega) > 1$ and (5.12) gives $r_m(\omega) > 1$. On the other hand if the system is not robustly stable, then $k_m(\omega) \leq 1$ and (5.13) gives $r_m(\omega) \leq 1$.

■

5.3 MAIN RESULTS - Computation of r_m and k_m

In this section we compute for the model uncertainty structure of (5.1), (5.2) and (5.3) the robustness margin r_m defined by (5.10) and next the multivariable stability margin k_m given in (1.1). The calculation of $r_m(\omega)$ is reduced to a finite number of multivariable stability margin calculations for real parametric uncertainty. Before stating this as theorem 5.4 we need to go through certain preliminary manipulations. First it holds:

$$\bar{\sigma}\left[\tilde{M}(j\omega,\Delta_p)\right] < 1 \Longleftrightarrow \rho\left[\tilde{M}^*(j\omega,\Delta_p)\tilde{M}(j\omega,\Delta_p)\right] < 1 \tag{5.14}$$

where $\rho(M)$ denotes the spectral radius of a matrix M. Let

$$p(z,\omega,\Delta_p) \overset{\text{def}}{=} \det\left[zI - \tilde{M}^T(-j\omega,\Delta_p)\tilde{M}(j\omega,\Delta_p)\right] \tag{5.15}$$

be the characteristic polynomial of $\tilde{M}^T(-j\omega,\Delta_p)\tilde{M}(j\omega,\Delta_p)$. Note that if we express $p(z,\omega,\Delta_p)$ as

$$p(z,\omega,\Delta_p) = a_l z^l + a_{l-1}z^{l-1} + \ldots + a_1 z + a_0 \tag{5.16}$$

the coefficients $a_i(\omega,\Delta_p)$, $i = 0,\ldots,l$ are real rational functions of ω and the parameters in Δ_p. Condition (5.14) is equivalent with verifying that all the roots of $p(z,\omega,\Delta_p)$ with respect to z for a fixed $s = j\omega$ and $\Delta_p \in \mathbf{\Delta}_p$ are inside the unit disc.

There is a known algorithm that checks if the roots of a polynomial are inside the unit disc. This is the Jury[41] test, an application of the more general Schur-Cohn[42,43] procedure, which is frequently found in digital control analysis. If we apply this procedure to $p(z, \omega, \Delta_p)$ we obtain [1]

1	a_{l-1}	$\dots$	$\dots$	a_0
a_0	a_1	$\dots$	$\dots$	1
a_l^1	a_{l-1}^1	$\dots$	$\dots$	a_1^1
a_1^1	a_2^1	$\dots$	$\dots$	a_l^1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
a_l^{l-1}				

with

$$a_i^k = a_i^{k-1} - \alpha_k a_{k-i}^k \quad ; \quad \alpha_k = a_k^k / a_l^k$$

Let us define for convenience $f_i(\omega, \Delta_p) \equiv a_l^i$, $i = 1, \dots, l-1$.

A necessary and sufficient condition for $p(z, \omega, \Delta_p)$ to have all its roots inside the unit disc is then

$$f_i(\omega, \Delta_p) > 0 \quad ; \quad i = 1, \dots, l-1 \tag{5.17}$$

Next let

$$f_i(\omega, \Delta_p) = \frac{n_i(\omega, \Delta_p)}{d_i(\omega, \Delta_p)} \quad ; \quad i = 1, \dots, l-1 \tag{5.18}$$

where $n_i(\omega, \Delta_p)$ and $d_i(\omega, \Delta_p)$ are real valued multivariate polynomials in ω and the δ_i's.

Define

$$g_i(\omega, \Delta_p) = n_i(\omega, \Delta_p) \cdot d_i(\omega, \Delta_p) \quad ; \quad i = 1, \dots, l-1 \tag{5.19}$$

[1] We can assume with no loss of generality that $p(z, \omega, \Delta_p)$ is monic by dividing throughout by a_l without changing the roots of $p(z, \omega, \Delta_p)$ in z and the form of the other coefficients.

The main result then follows:

Theorem 5.4:

Consider

$$k_i(\omega) \overset{\text{def}}{=} \inf\{k \in [0,\infty) \mid g_i(\omega, k\Delta_p) = 0 \quad ; \quad \Delta_p \in \mathbf{\Delta}_p\} \quad ; \quad i = 1, \dots, l-1 \quad (5.20)$$

and

$$k_r(\omega) \overset{\text{def}}{=} \inf\{k \in [0,\infty) \mid \det[I + k\Delta_p M_{22}(j\omega)] = 0 \quad ; \quad \Delta_p \in \mathbf{\Delta}_p\} \quad (5.21)$$

Assume that

$$\bar{\sigma}\,[M_{11}(j\omega)] < 1 \quad (5.22)$$

Then

$$r_m(\omega) = \min\{k_r(\omega), k_1(\omega), \dots, k_{l-1}(\omega)\} \quad (5.23)$$

Proof:

From (5.14) and (5.15), $\bar{\sigma}\left[\tilde{M}(j\omega, k\Delta_p)\right] < 1$ if and only if the roots of $p(z, \omega, \Delta_p)$ are inside the unit disc and this, by (5.17), is equivalent with $f_i(\omega, k\Delta_p) > 0$, $i = 1, \dots, l-1$.

From (5.19) we obtain that

$$\bar{\sigma}\left[\tilde{M}(j\omega, k\Delta_p)\right] < 1 \quad \Longleftrightarrow \quad g_i(\omega, k\Delta_p) > 0 \quad ; \quad i = 1, \dots, l-1 \quad (5.24)$$

This implies that

$$\begin{aligned} \hat{r}_m(\omega) &= \inf\left\{k \in [0,\infty) \mid \bar{\sigma}\left[\tilde{M}(j\omega, k\Delta_p)\right] \geq 1 \quad ; \quad \Delta_p \in \mathbf{\Delta}_p\right\} \\ &= \inf\{k \in [0,\infty) \mid g_i(\omega, k\Delta_p) = 0 \quad ; \quad \Delta_p \in \mathbf{\Delta}_p \quad ; \quad i = 1, \dots, l-1\} \\ &= \min\{k_i(\omega) \quad ; \quad i = 1, \dots, l-1\} \end{aligned} \quad (5.25)$$

Where the last equality follows from definitions (5.20) and from the fact that for $k = 0$, $g_i(\omega, 0) > 0$ as the latter is equivalent with (5.22).

From the definition of $r_m(\omega)$ in (5.10) and (5.25) the result follows inmediately.

■

Theorems 5.3 and 5.4 reduce the problem of robust stability for the feedback system of figure 5.1 to testing condition (5.22) which by corollary 5.1 is necessary for robust stability and to a finite number of real stability margin calculations.

The following remarks are in order.

(i) The procedure should be started by first checking the neccessary condition (5.22). This corresponds to checking robust stability for complex perturbations without real parametric uncertainty. If this test is positive, one then continues with the computation of $k_r(\omega)$ in (5.5) and $k_i(\omega)$, $i = 1, \ldots, l-1$ in (5.20).

(ii) The procedure for the computation of $k_i(\omega)$ in (5.20) will produce a frequency ω_o for which instability occurs, if this is the case, and also the worst combination of real parameters and complex perturbations. This information should be useful for the designer on how to modify the compensator so that robust closed-loop stability is obtained.

(iii) From (5.15) and also from (5.19) it is apparent that the $g_i(\omega, \Delta_p)$ are in general not multilinear in the δ_j's even if $\det\left\{I + \begin{bmatrix} \Delta_b & 0 \\ 0 & \Delta_p \end{bmatrix} M(j\omega)\right\}$ originally is. Therefore the algorithm in [11,12] cannot be applied and the one in [13] must be used.

(iv) It was noted already that the $g_i(\omega, \Delta_p)$ are real valued multivariate polynomials in ω and the δ_j's in Δ_p. This greatly simplifies the algorithm in [11-13] and increases significantly its speed as compared to the general case of Chapter 2. This has been explained in greater detail in Chapter 3.

(v) For high dimensional problems, obtaining the polynomials $g_i(\omega, \Delta_p)$,

$i = 1, \ldots, l-1$ in (5.19) can be tedious and even risky when done by hand. The development in the recent years of programs for symbolic manipulations should resolve this situation.

We now turn to the computation of k_m for the system of figure 5.1. To this end we define

$$r_m(\kappa, \omega) \overset{\text{def}}{=} \inf \left\{ k \in [0, \infty) \mid \bar{\sigma}\left[\tilde{M}(j\omega, k\Delta_p)\right] \geq \frac{1}{\kappa} \quad ; \quad \Delta_p \in \boldsymbol{\Delta}_p \right\} \tag{5.26}$$

The computation of $r_m(\kappa, \omega)$ for a given κ is accomplished by applying theorem 5.4. We next prove the following lemma.

<u>Lemma 5.1</u>:

The function $r_m(\kappa) : \mathcal{R}^+ \longrightarrow \mathcal{R}^+$ defined in (5.26) (and where dependence on ω is omitted) is nonincreasing.

<u>Proof</u>:

Let $\kappa_1 > \kappa_2$, $r_1 \equiv r_m(\kappa_1)$ and $r_2 \equiv r_m(\kappa_2)$. Assume that $r_1 > r_2$. Then there is no $\Delta_p \in \boldsymbol{\Delta}_p$ such that

$$\bar{\sigma}\left[\tilde{M}(j\omega, r_2\Delta_p)\right] \leq \frac{1}{\kappa_1} \tag{5.27}$$

by the definition of $r_m(\kappa)$ in (5.26). By assumption $\frac{1}{\kappa_1} < \frac{1}{\kappa_2}$ and (5.27) gives a contradiction since by (5.26) there exists a $\Delta_p^* \in \boldsymbol{\Delta}_p$ such that $\bar{\sigma}\left[\tilde{M}(j\omega, r_2\Delta_p^*)\right] \geq \frac{1}{\kappa_2}$. Therefore $\kappa_1 > \kappa_2$ implies $r_1 \leq r_2$ and $r_m(\kappa)$ is a nonincreasing function.

■

Lemma 5.1 allows the computation of $k_m(\omega)$ by means of a simple iteration.

<u>Theorem 5.5</u>:

Consider the sequence $\kappa_{n+1} = r_m(\kappa_n)$, where κ_0 is arbitrary and $r_m(\kappa)$ is defined by (5.26) and with respect to the uncertainty structure for the system of figure 5.1.

Then

$$\lim_{n \longrightarrow \infty} \kappa_n(\omega) = k_m(\omega) \tag{5.28}$$

the multivariable stability margin at frequency ω.

Proof:

By the definition of $k_m(\omega)$ in (5.7) and of the function $r_m(\kappa)$ in (5.26), we have that $k_m(\omega)$ is the smallest fixed point $\kappa^* = r_m(\kappa^*)$. By lemma 5.1 the function $r_m(\kappa) : \mathcal{R}^+ \longrightarrow \mathcal{R}^+$ has a unique fixed point and therefore $\lim_{n \longrightarrow \infty} \kappa_n = \kappa^* = k_m(\omega)$.

■

A good candidate for κ_0, the starting value for the iterative procedure to find k_m, is provided by k_r given in (5.5).

In summary, we have shown that the robust stability of a system with any number of real uncertain parameters and a complex perturbation can be checked by computing the robustness margin $r_m(\omega)$ or the multivariable stability margin $k_m(\omega)$. The calculation of $r_m(\omega)$ is accomplished by a finite number of multivariable stability margin calculations for real parametric uncertainty and the calculation of $k_m(\omega)$ is done by iterating on the procedure that gives $r_m(\omega)$.

In the introduction we discussed that other robustness problems are also addressed by these results. This point is further illustrated by the following examples.

5.4 EXAMPLES

5.4.1 Example 1

We consider a feedback system with loop transfer function $L(s)$ given by:

$$L(s) = \frac{g \quad 10^3 (1 + s/30)^2}{(p+s)(1+10s)(1+s/100)^2(1+s/300)} [1 + \delta_1(s)] \tag{5.29}$$

We assume that the pole p and the gain g are uncertain with variations

$$\begin{cases} g = 1 \pm \delta_2 & |\delta_2| < 0.5 \\ p = 1 \pm \delta_3 & |\delta_3| < 0.2 \end{cases} \tag{5.30}$$

and that $\delta_1(s)$ represents fast neglected dynamics that satisfies:

$$|\delta_1(s)| < 0.2\sqrt{1 + (\omega/10)^2} \tag{5.31}$$

The closed loop system is redrawn in figure 5.3 with weights and nominal system given by

$$\begin{aligned} L_o(s) &= \frac{10^3(1+s/30)^2}{(1+s)(1+10s)(1+s/100)^2(1+s/300)} \\ W_1(s) &= \frac{(1+s/10)}{5} \\ W_2(s) &= L_o(s) \\ W_3(s) &= \frac{-1}{(s+1)} \end{aligned} \tag{5.32}$$

This system is next rearranged in the form of figure 1.1 where Δ contains two real δ_2 and δ_3 and one complex parameter δ_1. For this example we obtained

$$r_m = 10.52 \qquad \text{and} \qquad k_m = 1.65$$

achieved at 81 and 13.5 rad/s respectively. Therefore the system satisfies our performance criteria for all possible parameter variations. The worst combination of perturbations is $\delta_2 = 0.5$ and $\delta_3 = -0.2$ in both cases. For this example we also checked robustness by calculating the Structured Singular value μ and obtained $k_m = 1/\mu = 0.64$ at 10 rad/s, which would indicate that the system is not robustly stable. This proves how covering real parameter variations with complex uncertainty can be arbitrarily conservative.

5.4.2 Example 2

We apply the previous procedure on a satellite attitude control design described in [44].

The plant is modeled as two masses connected by a spring with torque constant k and viscous damping d. The transfer function is,

$$p(s) = \frac{10\,(ds + k)}{s^2\,(s^2 + 11ds + 11k)} \tag{5.33}$$

with nominal values $k_0 = 0.09$ and $d_0 = 0.003$. The controller derived in [44] is

$$c(s) = \frac{(1.4s + 1)\left[(s/.9)^2 + 1\right]}{2\left[(s/25) + 1\right]^2} \tag{5.34}$$

which although improper gives a strictly proper loop transfer function $l(s) = p(s)c(s)$. We express k and d as $k = \delta_1^2$, $d = \delta_1\delta_2$ where

$$\begin{aligned} 0.3 \leq \delta_1 \leq 0.63 \\ 0.01 \leq \delta_2 \leq 0.06 \end{aligned} \tag{5.35}$$

It can be shown that the closed-loop system remains stable for all parameter variations in (5.35). The performance objective is to obtain a good compromise between tracking speed and accuracy of the pointing body.

We formulate this by requiring that the transfer function from command r to tracking error e is small (in the ∞-norm sense). We also want to avoid possible saturation of the actuators by sensor noise and we require that the transfer function from n to u is also small (see figure 5.4). The frequency dependent weights are chosen to reflect our knowledge on the frequency content of the signals r and n. Here we take $W_r = 1/s$ and $W_n = s/(2s+1)$.

The feedback system of figure 5.4 is next rearranged as in figure 5.5, and the robust performance theorem in [16] is applied by considering a complex block Δ_b around $\tilde{M}(s, \delta)$ as in figure 5.2.

A measure of performance in this framework is given by

$$\eta(\delta) = \max_{\omega} \bar{\sigma}\left[\tilde{M}(j\omega, \delta)\right] \tag{5.36}$$

Smaller $\eta(\delta)$ reflects better performance. We compute the worst performance for δ allowed by (5.35) by calculating η^* such that

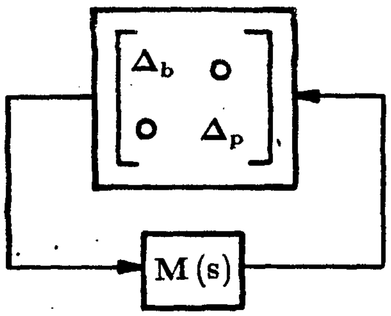
$$r_m(\eta^*) \stackrel{\text{def}}{=} \inf_{\omega}\left\{\inf\left[k \in [0,\infty) \mid \det\left(I + \frac{\tilde{M}(j\omega, k\delta)}{\eta^*}\right) = 0 \quad \text{for} \quad \text{some} \quad \delta \quad \text{in} \quad (5.35)\right]\right\} = 1 \tag{5.37}$$

(see figure 5.6). We thus obtained $\eta^* = 4.15$ at frequency ω_0 and for a combination of parameters $k^* = 0.397$ and $d^* = 0.038$. To check performance degradation due to parametric uncertainty, we also calculate the nominal performance:

$$\eta_0 = \max_{\omega} \bar{\sigma}\left[\tilde{M}(j\omega, \delta_0)\right] = 3 \tag{5.38}$$

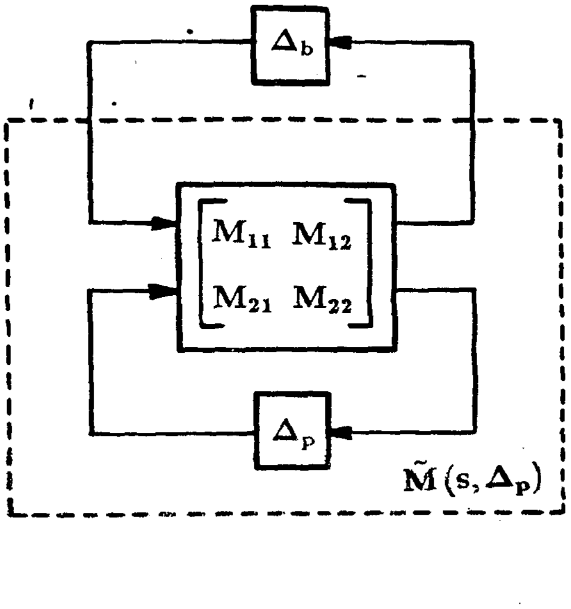
The percentage of nominal performance degradation is then:

$$\frac{\eta - \eta_o}{\eta_o}\text{x}100 = 38\%$$



$\begin{bmatrix} \Delta_b & 0 \\ 0 & \Delta_p \end{bmatrix}$
M (s)

Figure 5.1: General Δ structure with real parametric and complex block uncertainty.



Δ_b
$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$
Δ_p
$\tilde{M}(s, \Delta_p)$

Figure 5.2: Separation of parametric and complex uncertainty.

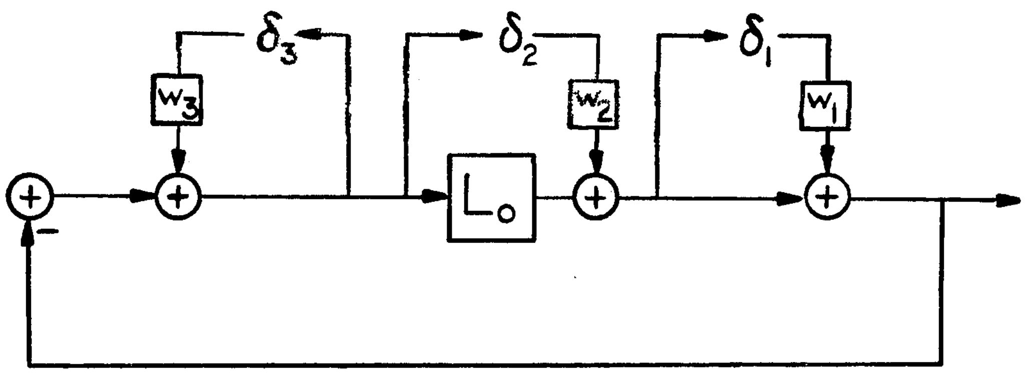


Figure 5.3: Feedback system with two uncertain parameters and unstructured dynamic uncertainty.

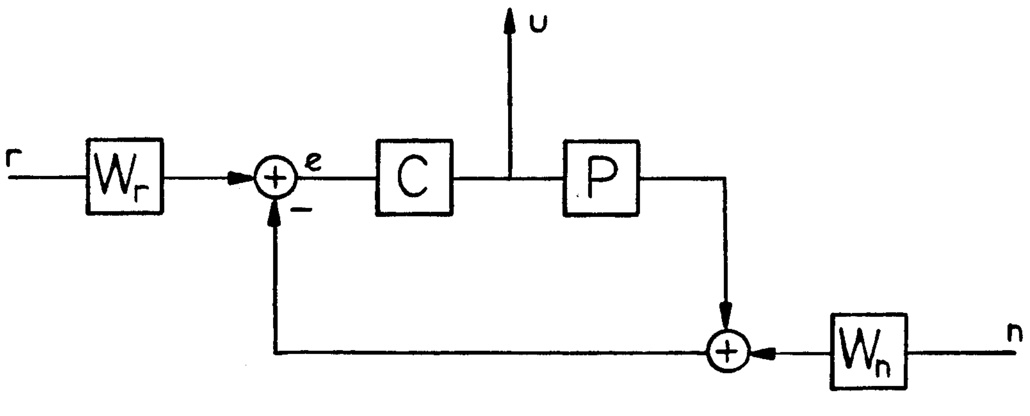


Figure 5.4: Feedback system and performance weights.

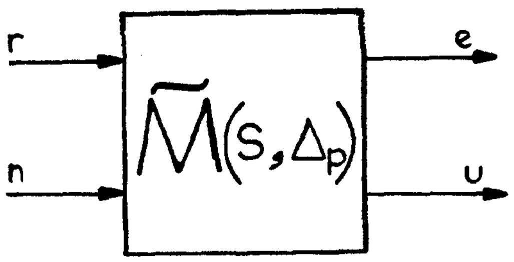


Figure 5.5: General structure for robust performance analysis.

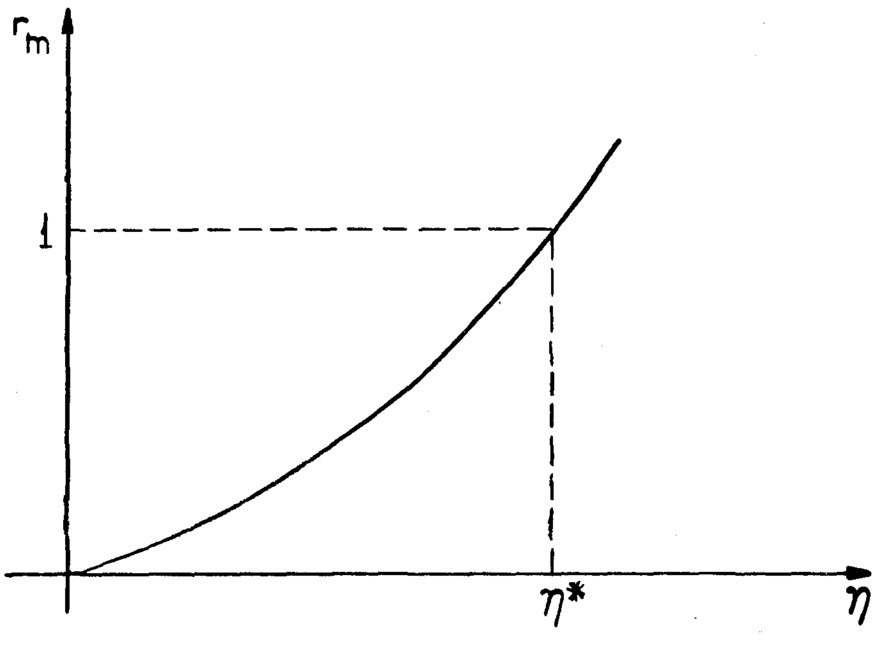


Figure 5.6: Robustness margin as function of the performance measure.

Chapter 6

General uncertainty analysis

6.1 INTRODUCTION

In this chapter we extend the results of chapter 5 for the general uncertainty structure of n real (possibly related) uncertain parameters and simultaneous structured dynamic uncertainty taking the form of m complex blocks.

More specifically we develop a procedure to compute the robustness margin r_m (see section 5.2) and through it the multivariable stability margin k_m and structured singular value μ. The computation of these quantities is exact for the case of three or less complex blocks and any number of real parameters. This is the same restriction that applies to μ as explained in subsection 1.3.2 .

As a result, a powerful methodology for analyzing robust stability and performance in linear time invariant feedback systems is obtained. Performance can be defined in terms of the induced norm from certain external input signals to certain output error signals of the system and it is treated by adding an extra complex "performance" block[16]. Yet another description of performance can be the requirement that the closed loop poles remain in a certain region of the left half plane for all model perturbations. The latter can also be checked by this procedure by searching over the boundary of this region instead of the usual frequency search.

The problem formulation is similar in spirit to the one given in section 5.2 and is described in section 6.2. In section 6.3 we describe the main results, which are applied to an example in section 6.4. This chapter is concluded in section 6.5 with some final remarks.

6.2 PROBLEM FORMULATION

The statement of the problem will be similar to the one in last chapter, although we are considering now a more general class of plants. The uncertainty structure used throughout the thesis (see figure 5.1) is defined next:

$$\mathbf{\Delta} \overset{\text{def}}{=} \{diag\,(\Delta_b, \Delta_p) \mid \Delta_b \in \mathbf{\Delta_b} \quad ; \quad \Delta_p \in \mathbf{\Delta_p}\} \tag{6.1}$$

$$\mathbf{\Delta_b} \overset{\text{def}}{=} \{diag\,(\Delta_1, \ldots, \Delta_m) \mid \Delta_i \in \mathcal{C}^{r_i x r_i} \quad ; \quad \bar{\sigma}\,(\Delta_i) \leq 1\} \tag{6.2}$$

$$\mathbf{\Delta_p} \overset{\text{def}}{=} \{diag\,(\delta_1, \ldots, \delta_n) \mid \delta_i \in \mathcal{R} \quad ; \quad |\delta_i| \leq 1\} \tag{6.3}$$

The nominal system is given by the rational transfer matrix $M(s)$ which can be subdivided into four blocks conformably with the dimensions of Δ_b and Δ_p. We thus obtain figure 5.2 in which $\tilde{M}\,(s, \Delta_p)$ is described by the following Linear Fractional Transformation:

$$\tilde{M}\,(s, \Delta_p) \overset{\text{def}}{=} M_{11} - M_{12}\Delta_p\,(I + M_{22}\Delta_p)^{-1}\,M_{21} \tag{6.4}$$

For the system in figure 5.1 to be robustly stable it is necessary and sufficient to check:

(i) $\tilde{M}(s, \Delta_p)$ is stable for every $\Delta_p \in \mathbf{\Delta}_p$.

(ii) $\mu_{\mathbf{\Delta}_b}\left[\tilde{M}(j\omega, \Delta_p)\right] < 1 \qquad \forall \quad \Delta_p \in \mathbf{\Delta}_p \quad , \quad \omega \in [0, \infty)$.

Condition (i) states that the nominal plant with only real parametric uncertainty must be robustly stable. It can be checked by evaluating

$$k_r(\omega) \overset{\text{def}}{=} \inf\{k \in [0, \infty) \mid \det\,[I + k\Delta_p M_{22}(j\omega)] = 0 \quad \text{for} \quad \text{some} \quad \Delta_p \in \mathbf{\Delta}_p\} \tag{6.5}$$

and verifying that $k_r(\omega) > 1 \quad \forall\omega$. Condition (ii) is necessary and sufficient[16] for the robust stability of the nominal plant with the real uncertain parameters fixed at some values allowed by (6.3) and the complex uncertainty structure allowed by (6.2). Checking condition (ii) is the main objective of this chapter.

From (i) and (ii) we can also derive the following expression for the multivariable stability margin $k_m(\omega)$:

$$k_m(\omega) = \min\{\hat{k}_m(\omega), k_r(\omega)\} \tag{6.6}$$

where

$$\hat{k}_m(\omega) \overset{\text{def}}{=} \inf\left\{k \in [0,\infty) \mid \mu_{\Delta_b}\left[\tilde{M}(j\omega, k\Delta_p)\right] \geq \frac{1}{k} \quad \text{for} \quad \text{some} \quad \Delta_p \in \mathbf{\Delta}_p\right\} \tag{6.7}$$

Robust stability is equivalently expressed in terms of the robustness margin $r_m(\omega)$, defined as:

$$r_m(\omega) \overset{\text{def}}{=} \min\{\hat{r}_m(\omega), k_r(\omega)\} \tag{6.8}$$

where

$$\hat{r}_m(\omega) = \inf\left\{k \in [0,\infty) \mid \mu_{\Delta_b}\left[\tilde{M}(j\omega, k\Delta_p)\right] \geq 1 \quad \text{for} \quad \text{some} \quad \Delta_p \in \mathbf{\Delta}_p\right\} \tag{6.9}$$

It has been shown in last chapter that k_m and r_m are equivalent in the following sense:

$$\text{Robust Stability} \iff k_m(\omega) > 1 \iff r_m(\omega) > 1 \tag{6.10}$$

The calculation of μ_{Δ_b} is achieved by computing the upper bound in (1.5), which is exact for the case of three or fewer blocks ($m \leq 3$) and tight in more general cases[16]. This in turn leads to a lower bound $\underline{r_m}$ for r_m defined by

$$\underline{r_m}(\omega) \overset{\text{def}}{=} \min\{\underline{\hat{r}_m}(\omega), k_r(\omega)\} \tag{6.11}$$

with

$$\underline{\hat{r}_m}(\omega) \overset{\text{def}}{=} \inf\left\{k \in [0,\infty) \mid \inf_{D\in\mathcal{D}} \bar{\sigma}\left[D\tilde{M}(j\omega, k\Delta)D^{-1}\right] \geq 1 \quad \text{for} \quad \text{some} \quad \Delta \in \mathbf{\Delta}_p\right\} \tag{6.12}$$

and $\mathcal{D}$ is defined in (1.8).

Similarly we can define $\underline{k_m}$ and $\underline{\hat{k}_m}$ by changing the bound in (6.12) for the maximum singular value from unity to $1/k$. Since $\underline{r_m}(\omega)$ is a lower bound on $r_m(\omega)$ (equal to $r_m(\omega)$ when $m \leq 3$), robust stability is assured if $\underline{r_m}(\omega) > 1$ for all ω. The computation of $\underline{\hat{r}_m}(\omega)$ and by (6.11) of $\underline{r_m}(\omega)$ is explained in the next section.

We will focus our attention on $\underline{\hat{r}_m}$, since $\underline{\hat{k}_m}$ (and thus $\underline{k_m}$) can be computed from $\underline{\hat{r}_m}$ by means of an iteration (see theorem 5.5).

6.3 MAIN RESULTS

In this section we derive an equivalent characterization of $\underline{\hat{r}_m}(\omega)$ defined in (6.12) which allows its systematic computation.

We first define:

$$\underline{\hat{r}_m}(\omega, D) \overset{\text{def}}{=} \inf\left\{k \in [0, \infty) \mid \bar{\sigma}\left[D\tilde{M}(j\omega, k\Delta)D^{-1}\right] \geq 1 \quad \text{for} \quad \text{some} \quad \Delta \in \mathbf{\Delta}_p\right\} \tag{6.13}$$

then $\underline{\hat{r}_m}(\omega)$ is the minimum $k \in [0, \infty)$ such that:

$$\inf_{D \in \mathcal{D}} \bar{\sigma}\left[D\tilde{M}(j\omega, k\Delta_*)D^{-1}\right] = 1$$

for some $\Delta_* \in \mathbf{\Delta_p}$. Let $D_* \in \mathcal{D}$ and such that

$$\bar{\sigma}\left[D_*\tilde{M}(j\omega, \underline{\hat{r}_m}(\omega)\Delta_*)D_*^{-1}\right] = 1 \tag{6.14}$$

Then the following is true,

<u>Theorem 6.1</u>:

For $\underline{\hat{r}_m}(\omega, D)$ defined in (6.13) and $\underline{\hat{r}_m}(\omega)$ and D_* satisfying (6.14) it holds:

$$\underline{\hat{r}_m}(\omega) = \sup_{D \in \mathcal{D}} \underline{\hat{r}_m}(\omega, D) = \underline{\hat{r}_m}(\omega, D_*) \tag{6.15}$$

The proof of theorem 6.1 requires a number of results. We first prove the following characterization of convex functions.

Lemma 6.1:

Given a convex function $f : \mathcal{R}^m \longrightarrow \mathcal{R}$, and $\forall a, b \in \mathcal{R}^m$, let $x_\lambda \overset{\text{def}}{=} (1-\lambda)a + \lambda b$. Then either

$$\{f(x_\lambda) < f(a) \qquad \text{or} \qquad f(x_\lambda) < f(b)\} \qquad\qquad \forall \lambda \in (0,1) \tag{6.16}$$

or $\exists \lambda \in [0,1]$ such that x_λ is a global minimum of f, i.e.

$$f(x_\lambda) \leq f(x) \qquad \forall x \in \mathcal{R}^m \tag{6.17}$$

Proof:

Assume that none of the conditions (6.16) or (6.17) hold. Then $\exists \lambda^* \in (0,1)$ such that:

$$f(x_{\lambda^*}) \geq f(a) \qquad \text{and} \qquad f(x_{\lambda^*}) \geq f(b) \tag{6.18}$$

From the convexity of f and from (6.18):

$$\begin{aligned} f(x_{\lambda^*}) \leq (1-\lambda^*)f(a) + \lambda^* f(b) &\leq (1-\lambda^*)f(x_{\lambda^*}) + \lambda^* f(b) \\ &\leq (1-\lambda^*)f(a) + \lambda^* f(x_{\lambda^*}) \end{aligned} \tag{6.19}$$

$$\Longleftrightarrow \qquad f(x_{\lambda^*}) \leq f(b) \qquad \text{and} \qquad f(x_{\lambda^*}) \leq f(a) \tag{6.20}$$

and by (6.18) then $f(x_{\lambda^*}) = f(a) = f(b)$. If x_{λ^*} is not a global minimum of f, $\exists \quad c \in \mathcal{R}^m$ such that $f(x_{\lambda^*}) > f(c)$. Assuming that $x_{\lambda^*} \in (a,c)$ by convexity,

$$\begin{aligned} f(x_{\lambda^*}) &\leq (1-\rho)f(a) + \rho f(c) \\ &< (1-\rho)f(x_{\lambda^*}) + \rho f(x_{\lambda^*}) = f(x_{\lambda^*}) \end{aligned} \tag{6.21}$$

for some $\rho \in (0,1)$, which is clearly a contradiction. The same contradiction is obtained if $x_{\lambda^*} \in (b,c)$.

■

We can now prove the following:

Lemma 6.2:

Let $D_1, D_2 \in \mathcal{D}$ and $D_\lambda = (1-\lambda)D_1 + \lambda D_2$ for $\lambda \in (0,1)$. The function $\underline{\hat{r}_m}(\omega, D)$ defined in (6.13) satisfies [1]

$$\underline{\hat{r}_m}(D_1) < \underline{\hat{r}_m}(D_\lambda) \qquad \text{or} \qquad \underline{\hat{r}_m}(D_2) < \underline{\hat{r}_m}(D_\lambda) \tag{6.22}$$

or

$$\underline{\hat{r}_m}(D) \le \underline{\hat{r}_m}(D_\lambda) \qquad \forall D \in \mathcal{D} \tag{6.23}$$

Proof:

Take $\Delta_\lambda \in \mathbf{\Delta}_p$ such that:

$$\bar{\sigma}\left[D_\lambda \tilde{M}\left(j\omega, \underline{\hat{r}_m}(D_\lambda)\Delta_\lambda\right) D_\lambda^{-1}\right] = 1 \tag{6.24}$$

and define

$$f(D) = \bar{\sigma}\left[D \tilde{M}\left(j\omega, \underline{\hat{r}_m}(D_\lambda)\Delta_\lambda\right) D^{-1}\right] \tag{6.25}$$

It is known that $f(D)$ is convex in D[45]. Then by applying lemma 6.1 we obtain,

$$f(D_\lambda) < f(D_1) \qquad \text{or} \qquad f(D_\lambda) < f(D_2) \tag{6.26}$$

$$\text{or} \qquad f(D_\lambda) \le f(D) \qquad \forall D \in \mathcal{D} \tag{6.27}$$

From (6.24) and (6.26) we obtain

$$f(D_1) > 1 \qquad \text{or} \qquad f(D_2) > 1 \tag{6.28}$$

(1) Variable ω has been droped for simplicity.

The function $f_k(D) \overset{\text{def}}{=} \bar{\sigma}\left[D\tilde{M}(j\omega, k\Delta_\lambda)D^{-1}\right]$ is continuos with respect to k. For $k = 0$ we can assume without loss of generality that $f_0(D_i) < 1$ [(2)] where i can be 1 or 2. Since for $k = \underline{\hat{r}_m}(D_\lambda) = l$ it holds $f_l(D_i) > 1$ (see (6.28)), we have that $f_r(D_i) = 1$ for some $r < l = \underline{\hat{r}_m}(D_\lambda)$. By the definition of $\underline{\hat{r}_m}(D)$ in (6.13) and from (6.28) we obtain

$$\underline{\hat{r}_m}(D_i) \leq r < \underline{\hat{r}_m}(D_\lambda) \tag{6.29}$$

Also (6.27) and definition (6.13) readily imply

$$\underline{\hat{r}_m}(D) \leq \underline{\hat{r}_m}(D_\lambda) \qquad \forall D \in \mathcal{D} \tag{6.30}$$

■

We are now in a position to prove theorem 6.1 .

<u>Proof of Theorem 6.1</u>:

Consider any $D \in \mathcal{D}$ and let $D' = 2D_* - D$. Then $D_* = \frac{1}{2}(D + D')$ and by identifying $D_1 = D$ and $D_2 = D'$ in lemma 6.2 we obtain:

$$\underline{\hat{r}_m}(D) \leq \underline{\hat{r}_m}(D_*) \tag{6.31}$$

From (6.31) and D_* defined in (6.14), we obtain the desired result.

■

We next prove the following important property of the function $\underline{\hat{r}_m}(\omega, D)$.

<u>Theorem 6.2</u>:

The function $\underline{\hat{r}_m}(\omega, D)$ is strictly quasiconcave [(3)].

---

(2) If $f_0(D_i) \geq 1$, then $\underline{\hat{r}_m}(D_i) = 0$ and it can be shown that $\underline{\hat{r}_m}(D) = 0$ for all D. This in turn implies that $\inf_{D \in \mathcal{D}} \bar{\sigma}\left[D\tilde{M}(j\omega, 0 \cdot \Delta)D^{-1}\right] \geq 1$. If $m \leq 3$, then the system with only complex perturbations (i.e. assuming that there is no real parametric uncertainty) is unstable and no further testing is necessary. If $m > 3$, robust stability with only complex perturbations cannot be inferred by the upper bound on μ_{Δ_b} given in (1.5). Therefore our test which is exact with respect to real parameter variations can not be conclusive either.

(3) As a reminder, a function $f : \mathcal{R}^m \longrightarrow \mathcal{R}$ is called *strictly quasiconcave* iff

$$f(b) > f(a) \quad \Longrightarrow \quad f[(1-\lambda)a + \lambda b] > f(a)$$

Proof:

Assume that $\underline{\hat{r}_m}(D)$ is not strictly quasiconcave. Then $\exists\lambda^* \in (0,1)$ and $D_1, D_2 \in \mathcal{D}$ such that

$$\underline{\hat{r}_m}(D_2) > \underline{\hat{r}_m}(D_1) \geq \underline{\hat{r}_m}(D_{\lambda^*}) \tag{6.32}$$

But (6.32) contradicts lemma 6.2 and therefore theorem 6.2 is true.

■

The computation of $\underline{\hat{r}_m}(\omega)$ has been reduced in view of theorem 6.1 to a search for the maximum of the quasiconcave function $\underline{\hat{r}_m}(\omega, D)$. For a fixed D, $\underline{\hat{r}_m}(\omega, D)$ is computed by a technique explained in section 5.3. Briefly the procedure is the following,

(i) Transform the condition $\bar{\sigma}\left[D\tilde{M}(j\omega, k\Delta)D^{-1}\right] < 1$ to a condition on the roots of a polynomial with coefficients being polynomic in the real parameters in $\mathbf{\Delta}_\mathrm{p}$.

(ii) Apply Jury's test[41] to find necessary and sufficient conditions for all roots of this polynomial to be inside the unit disc.

In this way $\underline{\hat{r}_m}$ is obtained as

$$\underline{\hat{r}_m}(\omega, D) = \min\{k_1(\omega, D), \ldots, k_{l-1}(\omega, D)\} \tag{6.33}$$

where

$$k_i(\omega, D) \overset{\text{def}}{=} \inf\{k \in [0,\infty) \mid g_i(\omega, D, k\Delta_p) = 0 \quad ; \quad \Delta_p \in \mathbf{\Delta}_p\} \quad ; \quad i = 1, \ldots, l-1 \tag{6.34}$$

and the g_i's are polynomial functions in $\mathbf{\Delta}_\mathrm{p}$, obtained from the Jury procedure.

(iii) Calculate $k_i(\omega, D), i = 1, \ldots, l-1$ by the procedure of chapter 3. For details see chapter 5.

$\forall a, b \in \mathcal{R}^m$ and $\lambda \in (0,1)$[46]. Strictly quasiconcave functions have the property that any local maxima are also global[47]. Therefore the computation of $\underline{\hat{r}_m}$ can be achieved by a straightforward search procedure in view of theorems 6.1 and 6.2.

The procedure to compute $\underline{k_m}$ is similar to the one described in last chapter and it is based on iterating on the calculation of $\underline{r_m}$. More specifically let $\underline{r_m}(\kappa) \overset{\text{def}}{=} \min\{\underline{\hat{r}_m}(\kappa), k_r(\omega)\}$, where now

$$\underline{\hat{r}_m}(\kappa) \overset{\text{def}}{=} \inf\left\{k \in [0,\infty) \mid \inf_{D\in\mathcal{D}} \bar{\sigma}\left[D\tilde{M}(j\omega, k\Delta)D^{-1}\right] \geq 1/\kappa \quad \text{for} \quad \text{some} \quad \Delta \in \mathbf{\Delta}_p\right\} \tag{6.35}$$

It can be shown that $\underline{\hat{r}_m}(\kappa)$ is a nonincreasing function of κ and that the iteration $\kappa_{n+1} = \underline{r_m}(\kappa_n)$ converges to $\underline{k_m}$ which is a lower bound on the multivariable stability margin:

$$\underline{k_m} = \lim_{n\longrightarrow\infty} \kappa_n \tag{6.36}$$

for an arbitrary κ_0.

<u>Remarks:</u>

(i) The calculation of $\underline{r_m}$ and $\underline{k_m}$ requires the computation of $\underline{\hat{r}_m}$ as defined in (6.34) and (6.35). The functions $g_i(\cdot)$ in the last equation are polynomic in the parameters, therefore the procedure described in [13] rather than the one in [11,12] should be applied. The fact that these functions are real valued, simplifies this algorithm and increases significantly the speed as compared to the general case. This can be found in detail in chapter 3.

(ii) The procedure to compute $\underline{r_m}$ is reduced to a finite number of real stability margin calculations at each fixed D. The problem in (6.15) can be solved by means of a nonlinear optimization program.

(iii) A frequency search over the $j\omega$-axis is required when analyzing robust stability as well as performance when the latter is treated by adding an extra "performance" complex block to the uncertainty structure[16]. If instead performance is defined as the inclusion of the closed loop poles in a certain region of the left-half complex plane, we should search over the boundary of this region.

(iv) For high dimensional problems, obtaining the polynomials $g_i(\omega, D, \Delta_p)$, $i = 1, \ldots, l-1$ in (6.35) can be tedious and even risky when done by hand. The development in the recent years of programs for symbolic manipulations should resolve this situation.

6.4 EXAMPLE

We consider a feedback system with loop transfer function $L(s)$ given by:

$$L(s) = \frac{g \quad 10^3 (1+s/30)^2}{(p+s)(1+10s)(1+s/100)^2(1+s/300)} [1+\delta_2(s)] \tag{6.37}$$

We assume that the pole p and the gain g are uncertain with variations

$$\begin{cases} g = 1 \pm \delta_3 & |\delta_3| < 0.5 \\ p = 1 \pm \delta_4 & |\delta_4| < 0.2 \end{cases} \tag{6.38}$$

and that $\delta_2(s)$ represents fast neglected dynamics that satisfies:

$$|\delta_2(s)| < 0.1\sqrt{1+(\omega/10)^2} \tag{6.39}$$

We also require a minimum amount of disturbance rejection, so we add a "performance" complex block with weight $W_1(s)$ to meet this specification. The closed loop system is redrawn in figure 6.1 with weights and nominal system given by

$$\begin{aligned} L_o(s) &= \frac{10^3 (1+s/30)^2}{(1+s)(1+10s)(1+s/100)^2(1+s/300)} \\ W_1(s) &= \frac{(1+s)}{10(s+0.01)} \\ W_2(s) &= \frac{(1+s/10)}{10} \\ W_3(s) &= 0.5 L_o(s) \\ W_4(s) &= \frac{-0.2}{(s+1)} \end{aligned} \tag{6.40}$$

This system is next rearranged in the form of figure 1.1 where Δ contains two complex δ_1 and δ_2 and two real parameters δ_3 and δ_4. For this example we obtained being $m = 2$,

$$r_m = \underline{r_m} = 1.97 \qquad \text{and} \qquad k_m = \underline{k_m} = 1.70$$

achieved at low frequencies and 3.88 rad/s respectively, with scalings $D_r = \text{diag}[3.1, 1]$ and $D_k = \text{diag}[0.98, 1]$. Therefore the system satisfies our performance criteria for all possible parameter variations. The worst combination of real perturbations is $(\delta_3, \delta_4) = (-0.5, 0.2)$ and $(-0.5, -0.2)$ respectively.

For this example we also checked robustness by calculating the Structured singular value μ and obtained $k_m = 1/\mu = 0.67$ at 10 rad/s. Although μ was computed for $m = 4$ complex blocks (δ_1 to δ_4) the upper and lower bounds on μ were tight, which would indicate that we do not have robust performance. This shows that treating real parameters as complex can be very conservative.

6.5 CONCLUSIONS

The problem of robust stability and performance in feedback control systems with n real (possibly related) uncertain parameters and structured unmodeled dynamics, the latter taking the form of m complex blocks dynamic uncertainties has been considered. We have shown that that the robust stability of a system with any number of real uncertain parameters which can be related to each other and complex blocks can be checked by computing lower bounds $\underline{r_m}$ and $\underline{k_m}$ for the robustness margin r_m or the multivariable stability margin k_m respectively. The calculation of $\underline{r_m}$ is accomplished by a finite number of multivariable stability margin computations for real parametric uncertainty and a nonlinear programming search over the scaling matrices D. The calculation of $\underline{k_m}$ is done by iterating on the procedure that gives $\underline{r_m}$.

The main contribution of this chapter and of the thesis is the generalization of the uncertainty structure described in subsection 1.3.2 by introducing any number of real parameters without adding any conservatism to the problem. In some sense it combines the procedures in subsections 1.3.1 and 1.3.2 to solve the most general case of uncertain FDLTI systems.

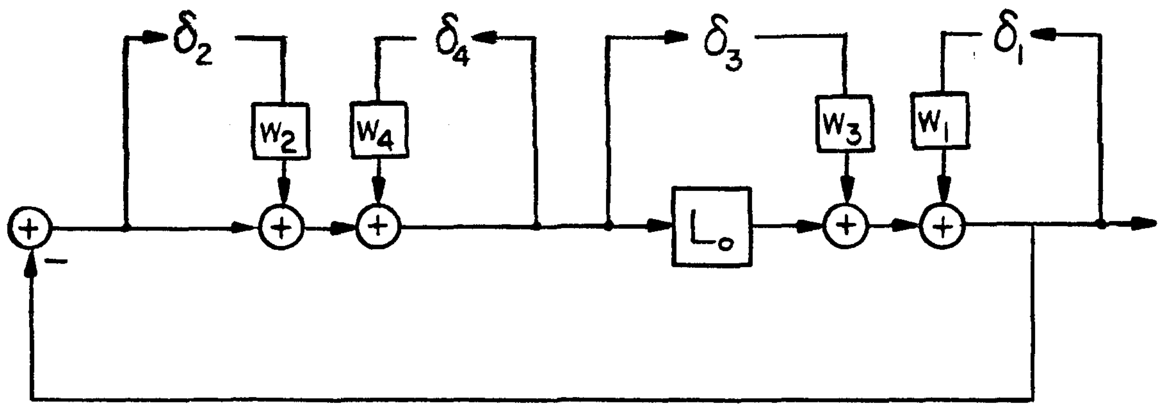


Figure 6.1: Feedback system with two uncertain parameters and structured dynamic uncertainty.

Chapter 7

Program statistics and performance

7.1 INTRODUCTION

This chapter presents the results obtained from the computation of the Multivariable stability margin over a great number of random examples using the programs described in Chapters 2 and 3. Different questions concerning the performance of the programs are considered and a comparison of certain characteristics for the two different implementations of the code are made.

As we have described before, the program to compute the multivariable stability margin can be implemented in two different ways. One of them is by taking the mapping function to be the closed loop characteristic polynomial of the system the robust stability of which is analyzed. The second implementation was described in Chapter 3 and uses as mapping functions, the polynomials obtained by applying the Routh-Hurwitz stability criterion to the closed loop characteristic polynomial. These functions are now real-valued, which simplifies significantly the code, besides eliminating the required search over frequencies.

Furthermore in both implementations we can find cases of independent and related parameters which lead to multilinear and polynomial functions of the uncertain parameters respectively. Although conceptually both situations are handled in a similar way, the amount of computation in the latter case increases as it should be expected. This leads to four different situations which should be analyzed and compared with each other.

The implementation of the program with a real valued mapping can arise either when analyzing robust stability for real parameter variations using the Routh-Hurwitz criteria or when doing the analysis for a combination of real and complex perturbations. It is very unlikely that in any of these situations we will obtain a mapping function which is multilinear in the uncertain parameters, even when the closed loop characteristic polynomial is so. Furthermore a multilinear real function will always achieve the origin at the image of a vertex of the hypercube. It makes sense then to do the analysis only for the real mapping function in the case of related uncertain parameters.

In summary we consider next the following three cases:

(1) Independent parameter case (multilinear function) with complex mapping.

(2) Related parameter case (polynomial function) with complex mapping.

(3) Related parameter case (polynomial function) with real mapping.

For all of them we have chosen a general polynomial structure whose coefficients (real or complex) have been selected at random. The criteria to choose a particular structure (i.e. fix the number of parameters and terms) was a compromise between having a sufficient number of parameters so that the structure of the mapping function could be considered *generic* and on the other hand to keep computational time for each example reasonable so that an ensemble of good size could be acquired and many different features could be checked.

Thus a general multilinear function of 5 parameters was choosen to be the mapping function for case (1) and a general polynomic function of 3 parameters for cases (2) and (3) above. In this last structure, the first parameter is cubed, the second squared and the last is linear, being equivalent to 6 parameters subject to three equality constraints. More specifically the mapping function in these situations is the following:

$$f(p) = \sum_{i=1}^{5} c_i \quad p_i + \sum_{\substack{i,j \\ i \neq j \leq 5}} c_{ij} \quad p_i p_j + \sum_{\substack{i,j,k \\ i \neq j \neq k \leq 5}} c_{ijk} \quad p_i p_j p_k + \sum_{\substack{i,j,k,l \\ i \neq j \neq k \neq l \leq 5}} c_{ijkl} \quad p_i p_j p_k p_l$$
$$+ c_{32} \quad p_1 p_2 p_3 p_4 p_5 \tag{7.1}$$

in (1) and

$$\begin{aligned} f(p) &= c_0 + c_1 p_1 + c_2 p_2 + c_3 p_3 + c_4 p_1^2 + c_5 p_1 p_2 + c_6 p_1 p_3 \\ &+ c_7 p_2 p_3 + c_8 p_2^2 + c_9 p_1 p_2 p_3 + c_{10} p_1^3 + c_{11} p_1^2 p_2 + c_{12} p_1^2 p_3 \\ &+ c_{13} p_1 p_2^2 + c_{14} p_2^2 p_3 + c_{15} p_1^3 p_2 + c_{16} p_1^3 p_3 + c_{17} p_1^2 p_2^2 + c_{18} p_1^2 p_2 p_3 \\ &+ c_{19} p_1 p_2^2 p_3 + c_{20} p_1^3 p_2 p_3 + c_{21} p_1^2 p_2^2 p_3 + c_{22} p_1^3 p_2^2 + c_{23} p_1^3 p_2^2 p_3 \end{aligned} \tag{7.2}$$

which is transformed by doing $p_1^3 = \bar{p}_1 \bar{p}_2 \bar{p}_3$, $p_2^2 = \bar{p}_4 \bar{p}_5$ and $p_3 = \bar{p}_6$ subject to

$$p_1 = \bar{p}_1 = \bar{p}_2 = \bar{p}_3 \qquad p_2 = \bar{p}_4 = \bar{p}_5 \qquad p_3 = \bar{p}_6 \tag{7.3}$$

in (2) and (3). Although most of the examples had the structures shown above, other structures with different number of parameters have also been tested. The real parameters vary in the range $p_i = 1 \pm 0.5$ and similarly for $\bar{p}$. All coefficients $c_i \in \mathcal{C}$ (or $\mathcal{R}$) have real and imaginary parts equally distributed inside the interval $[-5, 5]$ and are obtained as outputs of the F77 random number routine, each example being generated by a different "seed".

7.2 MULTILINEAR COMPLEX MAPPING

The number of random examples considered in this case was in the order of 2000. The probabilities of occurrence of an event are simply taken as (No. of ocurrences/Total No. of samples). In the following the data is presented and comments over the different issues that have been analyzed are made.

(i) The combination of parameters that will actually achieve the origin of the complex plane can be either a vertex, an edge or in general an m-dimensional face, where $m \leq 5$. The data obtained for all samples is shown in Table VII.1 and a brief explanation follows. It has been proved in [35] that at most the image of a 2-dimensional face will map to the origin, a generic result for polynomials that are multilinear in the parameters. If we assume that any point on the boundary of the image of the hypercube has the same probability of first reaching the origin, then this probability will be proportional to the amount of perimeter covered by each type of perturbation. Then we only need to analyze the perimeter of the image of a 2-dimensional face and relate it to the different perturbations, as seen in figure 7.1. A subset of the vertices of the hypercube of parameters will map to the "vertices" of the image, a set of measure zero over the whole perimeter which would make the probability of vertices to be zero. There are two reasons why we obtain a small but nonzero probability, one is the finite precision of the machine that assigns a finite nonzero part of this perimeter to the image of the vertices. The second is that by the form we have set the function in (7.1) it is similar to having a multilinear function in the parameters at a randomly distributed set of frequencies. We have found that the percentage of occurrence of vertices increases drastically for particular frequencies as will be explained in section 7.4. By distributing these high probabilities over the whole frequency range we will obtain as a result this nonzero percentage in this case. From the same figure we can see that most of the perimeter is the image of edges of the hypercube, which is supported by the data. This is important in the sense that for a first approach we only need to check the images of these edges, obtaining in that way a good upper bound estimation of k_m. The image of the inside of a 2-dimensional face maps to the boundary of the image in cases where there is a "crossing" of edges (figure 7.1) and gives a small part of the total perimeter. The fact that there are no higher order faces that will reach the origin supports the result in [35]. The same percentages have been obtained when changing the number of parameters to $3, 5$ and 9.

(ii) As explained in Chapter 2, when the lower bound has been reached either by the image of a vertex or an edge, upper and lower bounds coincide and no partitioning is necessary. The probability of this situation to arise before any partitioning has been made is 55.7 %, so in more than half of the samples we obtain the exact answer at the first stage of the computations. Under the same assumption we had in (i), this percentage tells us which is the relation between "exposed" and "hidden" perimeter of the image of the hypercube as seen in figure 7.1. Of course this does not include all cases where after partitioning we also obtain an edge, which explains the difference between the 90.5% of Table VII.1 and the percentage given above.

(iii) In the search for the lower bound of k_m we proceed by selecting with a certain criteria two possible candidates for the *critical vertices* and later checking if they are so. By carrying out a statistics of the number of iterations needed at this stage to find the actual critical vertices, we can conclude that the selection strategy is very efficient. This is shown in figure 7.2 where we can see how in most of the cases, the actual pair of *critical vertices* were obtained at the first choice.

(iv) At a certain partition r of the hypercube, if the lower bound k_{l_i} of domain i is higher than the upper bound of the last partition $k_{u(r-1)}$ we can eliminate this domain. This avoids the upper bound computation of k_{u_i} on this type of domains, which are called *unbounded.* Two different statistics have been made over this issue. The first issue is the probability of having *unbounded* domains on a partition and is plotted in figure 7.5 as the percentage of cases over the *ensemble* of samples that contain *unbounded* domains, as a function of the partition. We can see that this probability increases to 100 % as we continue partitioning. A possible explanation of this fact is that when dividing the hypercube, the upper and lower bounds of the next partition get tighter, then the domains not containing the offending perturbation(1) have higher probabilities of being rejected because the difference between their bounds and the complete partition bounds will increase. The second issue applies to the partitions that have *unbounded* domains and gives the percentage of these over the total number of domains on that partition. This is also plotted as a function of the partition in figure 7.3 where we can see that around half of the domains are *unbounded* at these partitions, a fact that reduces significantly the amount of computation.

(v) When all the domains of a partition have been evaluated and all *unbounded* domains eliminated, we proceed to determine the new upper and lower bounds of partition r. At this point we can still eliminate domains by comparing their lower bounds with the upper bound of the new partition k_{ur}. The difference between these, defined as *discarded* domains and the *unbounded* ones in (iv) is that the former are eliminated after computing both upper and lower bounds, while the latter are eliminated without the need to compute their upper bounds. *Discarded* domains can arise when the upper bound of the partition r decreases with respect to partition $(r-1)$, otherwise no new domains will be eliminated other than the *unbounded* ones. The same statistics as in (iv) are performed here. In figure 7.4 we can see the percentage of domains that have been eliminated at each partition both *discarded* and *unbounded.* By comparing with

(1) The offending perturbation is defined as the combination of parameters that first reaches the origin while increasing k.

figure 7.3, we can see that actually most of them have been eliminated in the first stage (i.e. are *unbounded*). In figure 7.5, we compare the probabilities of having *discarded* and *unbounded* domains as a function of the partition number. As before the difference among them is very low. From these facts we can conclude the following. Most of the domains are eliminated when computing their lower bounds, which saves more than half of the work. Second, as we continue partitioning we have seen that the number of new *discarded* domains are reduced almost to zero. This implies that after several partitions, the upper bound tends to remain constant while the lower bound will increase, a fact that has also been observed independently.

(vi) The opposing phenomena of partitioning and elimination of domains at each stage will result in an increase in the amount of computations as we look for the convergence of lower and upper bounds of k_m, so it is important to know the rate at which the number of domains increase. In figure 7.6 we compare the average number of domains at each partition with a linear growth at unit rate. We can see that the whole algorithmic procedure could be computed in polynomial time if we can find a way to compute both upper and lower bounds in this way, not being the growth of domains an issue.

(vii) If we only want to compute the lower bound of the whole hypercube with no further subdivisions, it would be convenient to know how good an estimate this is for k_m. In the average, considering all cases there is a high probability of having k_{low} 2 % below the actual k_m. This is mainly due to the fact that as seen in (ii) more than a half of the samples find exactly the stability margin by computing only the first lower bound. Even when excluding these cases the average error increases only to 6 % below the actual k_m. In all cases considered, the peak error was found to be 38 % below the actual margin.

(viii) In terms of the time to compute lower and upper bounds, the first computation takes 8 seconds while the second one 14.5 in the average over all examples considered, on a SUN-2/350. Also total computation times have been plotted as a function of the number of parameters in figure 7.14, the increase being near exponential in average, as expected. In this last test special care has been taken so that the timing differences would not be influenced by the different number of terms of the mapping function for every different number of parameters. A constant number of terms where chosen at random in defining the mapping function for all 3,5,7 and 9 parameters.

Some final remarks for the multilinear case follow. The testing of a large number of random samples has stressed the following: the strategy of selecting *a priori* critical vertices in the low bound search and eliminating domains before the calculation of the upper and lower bound of the partition proves to be efficient in terms of computations. The linear growth of the number of domains shows that the increase in domains is not an issue in the search for a polynomial time algorithm to compute k_m. Furthermore in many cases (55.7%) these domains have coincident upper and lower bounds which means that not only they will not undergo further partitions, but also that they will not need to be recomputed. Finally an empirical justification of the statistical results based on the fact that *generically* only vertices, edges or 2-dimensional faces can reach the origin[35] was discussed. Some typical plots of the image of the multilinear complex polynomial function with a vertex, an edge and a face as the offending perturbation, can be seen in figures 7.7, 7.8 and 7.9 respectively.

Vertex	**Edge**	**Face**	***n*-dimensional face ($n > 2$)**
0.9 %	90.5 %	8.6 %	None

Table VII.1

7.3 POLYNOMIAL COMPLEX MAPPING

The number of random examples considered in this case was on the order of 250. The mapping function is the one in equation (7.2) with random complex coefficients $(c_1, \ldots, c_{23})$.The same issues of section 7.2 have been analyzed and the results are presented next.

(i) The combination of parameters that can achieve the origin of the complex plane can be a vertex, an edge or a face as in the multilinear case, but also in general any n-dimensional face. We cannot apply the generic result for multilinear mappings[35] to this situation although the mapping function is multilinear in the generalized set of parameters, because we now have the set of constraints (7.3) that have to be met. For the mapping function (7.2) this means that the origin can be reached ranging from vertices to 5-dimensional faces. This is supported by the results shown in Table VII.2.

(ii) The situation in which the first intersection of the origin is by a vertex or an edge of the constrained set of parameters p, such that lower and upper bounds of the whole hypercube are coincident arises only in 10.5 % of the cases. This means that in most of the samples we had to partition to obtain the exact value of k_m. This is due to the fact that the set of constraints (7.3) adds a new requirement in obtaining this kind of situations, reducing its probability of occurrence as compared with the multilinear case. This is also one of the reasons why, in the average, the computation of the multivariable stability margin in the related parameter case is slower.

(iii) The search for the lower bound of k_m is exactly the same as in the multilinear case. Then the same comment applies to the selection of the critical vertices we had in section 7.2 (iii).

(iv) The probability of having unbounded domains increases as we continue partitioning although it reaches a high probability at a latter stage. The percentage of unbounded to total number of domains at a partition is now smaller than in the multilinear case and goes from 15 % to 40 %, depending on the partition. Both these facts also contribute to increase the computation time, the reason being the following. Since there are not as many unbounded domains as in the multilinear case and if the probability of even having unbounded domains at a given partition is lower, then the number of domains to be computed will be larger than in the independent parameter case.

(v) The same as in (iv) can be said concerning the regions discarded after evaluating upper and lower bounds of all domains in a partition. The only similarity with the multilinear case is that we still have most of the domains eliminated being of the unbounded type. This again supports the strategy of eliminating the domains at the early stage of the lower bound computation.

(vi) For the reasons given in (iv), we have a higher rate of increase in the domains than in the multilinear case. Although this fact, we can see in figure 7.10 that the increase in the number of domains is above the linear growth but still well below the parabolic and exponential increase. Ideally the linear growth of domains is achieved when (in average) half the domains need further subdivision and no domains are eliminated. The exponential growth occurs when all domains are subdivided and kept after each partition. The related parameter case seems to be in between these two situations.

(vii) The discrepancy between k_m and the lower bound of the whole hypercube is in the average around 22 % with a peak value of 62 %. This is due to the facts that only in 10.5 % of the cases we find $k_m = k_{low} = k_{upper}$ at the first computation of the lower bound and also that a larger variety of perturbations can achieve the origin (see Table VII.2).

(viii) In terms of the time to compute the lower bound it was found similar to the multilinear case being both procedures identical[(2)]. As explained in Chapter 2, the main difference between the independent and related parameter cases is in the upper bound computation. In the latter the search is carried out over edges of the constrained set, but to guarantee that one of these edges has crossed the origin, we need to check that a particular convex set that contains that edge (see subsection 2.3.3.2) has crossed completely the point $z = 0$. This is the main reason why there is a big difference in the computation time of the upper bound k_u with respect to the independent parameter case.

The main conclusion for the related parameter case is that although the speed is slower than in the multilinear one, this is mostly due to the increase in the number of computations to achieve the upper bound and lesser to the growth in the number of domains after each partitioning. Also there is a significant difference in the kind of perturbations that can first reach the origin. This can be appreciated by looking at the image of a sample function $f(p_1, p_2, p_3)$ in figure 7.11 as compared with figures 7.7, 7.8 and 7.9. We can see now that the convex hull determined by the vertices of this figure no longer contains the complete image of $f(\cdot)$.

(2) The different number of parameters in the multilinear and related parameter cases analyzed was taken into account to draw this conclusion.

Vertex	Edge	Face	3-D face	4-D face	5-D face
1.1 %	29.1 %	24.5 %	36 %	8.1 %	1.2 %

Table VII.2

7.4 POLYNOMIAL REAL MAPPING

When the mapping function is real valued (see chapter 3) although the algorithm remains the same, the implementation changes considerably. The main modification is that now instead of looking for lines connecting pairs of vertices intersecting the origin, we check only for individual vertices mapping onto the origin. This eliminates many troublesome situations (see section 2.3.2 and 2.3.3.2) in the lower bound search and also the need for an algorithm that generates neighbor vertices in the upper bound search as the one described in Appendix 2.5. The logic that checks if the convex hull of a generalized constrained edge goes completely over the origin is also eliminated, being this the main reason for the difference in computation time between independent and related parameter cases under a complex mapping, as explained in section 7.3-(viii). Also the computation time of lower and upper bounds will now be related by the ratio of all 2^n vertices to the subset of these vertices we check to find k_{upper}, as opposed to the complex mapping cases where the lower bound is achieved faster than the upper one.

Finally there is a very important reason that establishes the main difference in computation time for the real mapping case, that is the following. If we take a general multilinear CLCP as a function of frequency and perform the Routh-Hurwitz stability test, we obtain general real polynomic functions of the parameters that are products and sums of general multilinear real functions. The positivity of these functions is evaluated by the real mapping implementation of the program which will determine the stability margin, i.e. the lowest value of k at **all** frequencies for which the complex image of the CLCP reaches the origin. By experimenting with a general CLCP it was found that the percentage of vertices being the offending perturbations increased drastically when analyzing the function at the most critical frequency as compared with the result obtained from Table VII.1 which is general for any frequency. This means that not only we eliminate the frequency search but also the mapping functions we analyze have a much greater probability of having a vertex as the offending point. In fact many of them will be real multilinear functions which will always reach the origin at a vertex. Otherwise we will search over frequencies with greater probabilities of having to go through partitions or costful computations.

The number of random examples considered in this case was on the order of 2000. The mapping function is the one in equation (7.2) with random real coefficients $(c_1, \dots, c_{23})$. Other structures and number of parameters have also been tested. The results are presented next.

(i) The combination of parameters that can first achieve the origin of the complex plane can be a vertex, an edge, a face or in general any n-dimensional face. It was found that almost **all** perturbations where at the vertices of the hypercube, with a lower percentage of faces and 3-dimensional faces. No edges or higher order faces where found. The explanation of the difference with the complex mapping of section 7.3 has been detailed above. The percentages of each type of perturbation are shown in Table VII.3. This same test was performed over other structures and number of parameters. With the same number as in (7.2) but where each parameter is squared we found that the percentage of vertices obtained was also around 90%. When the number of parameters was reduced to two, both squared, this percentage was 66%.

(ii) The situation in which we first reach the origin by a vertex of the constrained set with no further partitions needed, appears in 33.7 % of the cases. This is better than in the complex mapping case, although not as good as in the multilinear complex map case. The reason as before is that the set of constraints (7.3) adds a requirement which reduces the probability with respect to the multilinear case of stopping the search at the first stage. This result was the same when we changed the structure and number of parameters as in (i).

(iii) In the search for the lower bound of k_m we now only need to find a single *critical vertex*. The same strategy has been used in selecting *a priori* this vertex, and the results are even better than in both complex mapping situations. There were no samples in which the selected vertex would have to be changed more than 4 times, being 83 % of the times the first choice the correct one. The plot can be seen in figure 7.12.

(iv) The pattern in the analysis of domains that are eliminated at the k_{low} search stage (i.e. unbounded) and the ones eliminated after all domains in a partition have been evaluated (i.e. discarded), is similar to the one in section 7.3. This means that this behavior is inherent to related parameter cases, which in general tends to increase the amount of computations as compared to the multilinear case. This issue seems not to be an important drawback in this case though.

(v) The same can be said concerning the growth of domains at each partition, being a direct consequence of the analysis in point (iv). The plot is shown in figure 7.13 compared with the linear, parabolic and exponential growths.

(vi) The difference between k_m and the lower bound of the whole hypercube is in average around 10.5 % with a peak value of 56 %. This seems to be much better than the related parameter case with complex mapping, but this is only because we now have a higher percentage of cases that obtain the exact k_m at the first stage of computations, as compared with the percentage in section 7.3.

(vii) Another reason why the speed of computation of the stability margin is dramatically increased is because the speed to compute lower and upper bounds is much higher than in the complex mapping cases. The different implementation described in Chapter 3 and at the beginning of this section support the results of 1.13 seconds to compute k_{low} and 0.62 seconds for k_{upper} in average. Also due to the arguments given at the beginning of this section, the upper bound takes less time than the lower one. The total time for different number of parameters is plotted in figure 7.15 and can be compared with the computation time in the multilinear complex case of figure 7.14. We can appreciate that it is ten times faster than the computation of a single frequency point of the multilinear case. The same procedure to allow a constant number of terms in the mapping function has been taken into account as mentioned in section 7.2-(viii).

The main conclusion for the related parameter case under a real mapping function is that not only it is much faster than both complex mapping cases, but it also offers the possibility of obtaining a very good upper bound of k_m by checking the vertices of the constrained set. Taking into account that also the computation of the lower bound k_{low} is in average 10.5% below the actual stability margin, an efficient computation for this case would be to check all vertices of the hypercube without further subdivisions.

Vertex	**Face**	**3-D face**	**Other**
90.3 %	1.65 %	8.07 %	None

Table VII.3

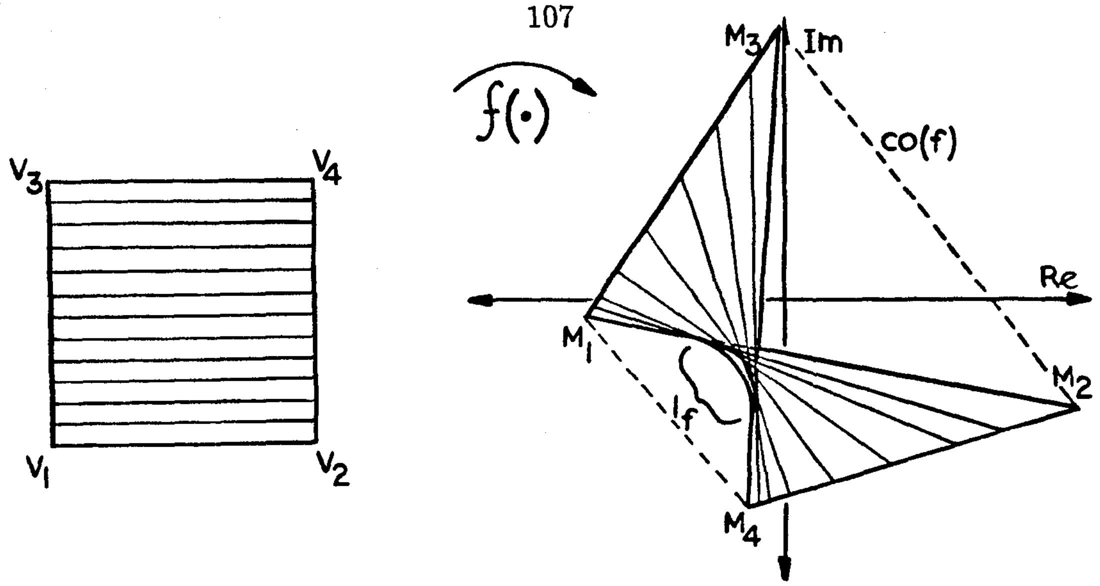


Figure 7.1: Image of a square showing "exposed" perimeter (in the convex hull boundary), "hidden" perimeter (strictly inside convex hull) and proper face image over the boundary (l_f).

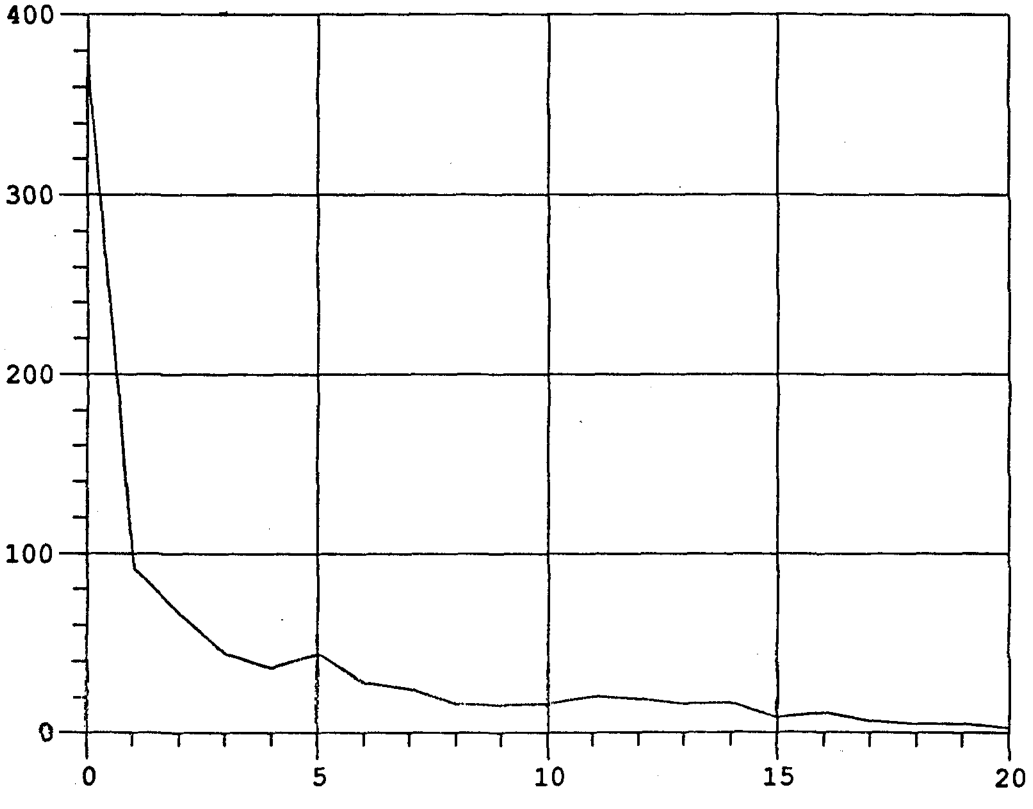


Figure 7.2: No. of samples vs. No. of iterations to find *critical* vertices.

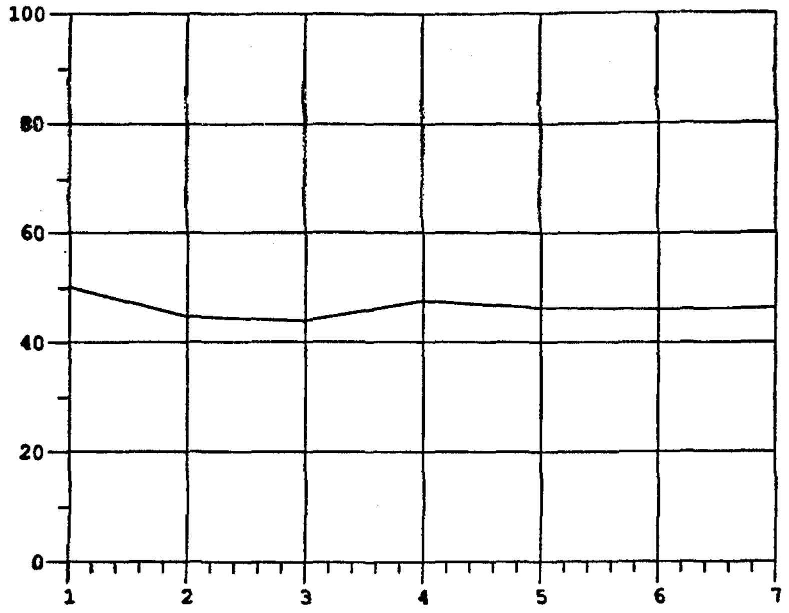


Figure 7.3: Unbounded domains/total domains in a partition (%) vs. Partition.

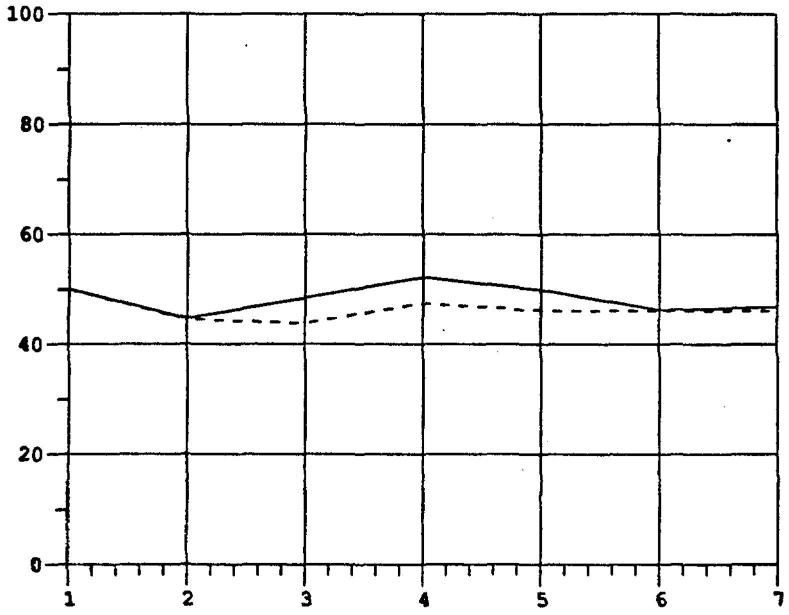


Figure 7.4: Thick line: Domains eliminated/total domains in a partition (%) vs. Partition. Dashed line: same as figure 7.3.

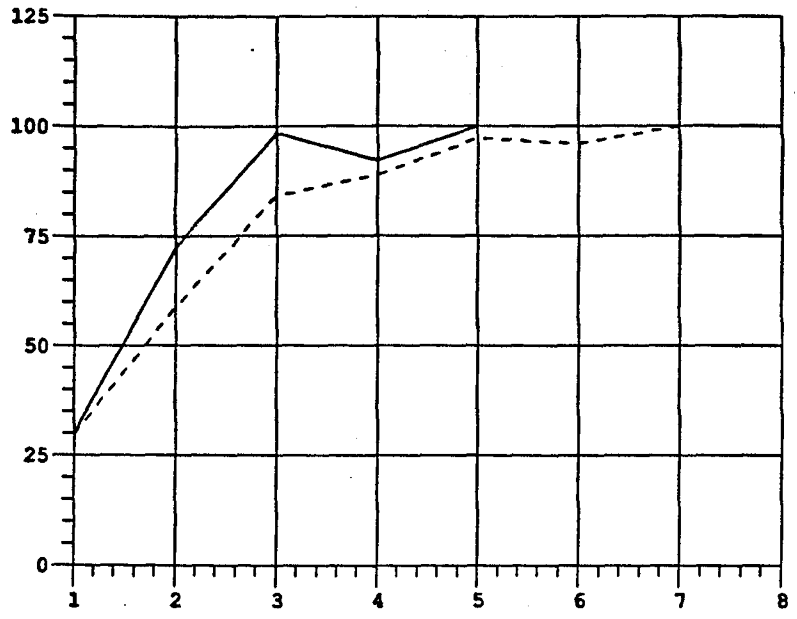


Figure 7.5: Thick line: Samples with domains eliminated/total samples (%) vs. Partition. Dashed line: Samples with unbounded domains/total samples (%) vs. Partition.

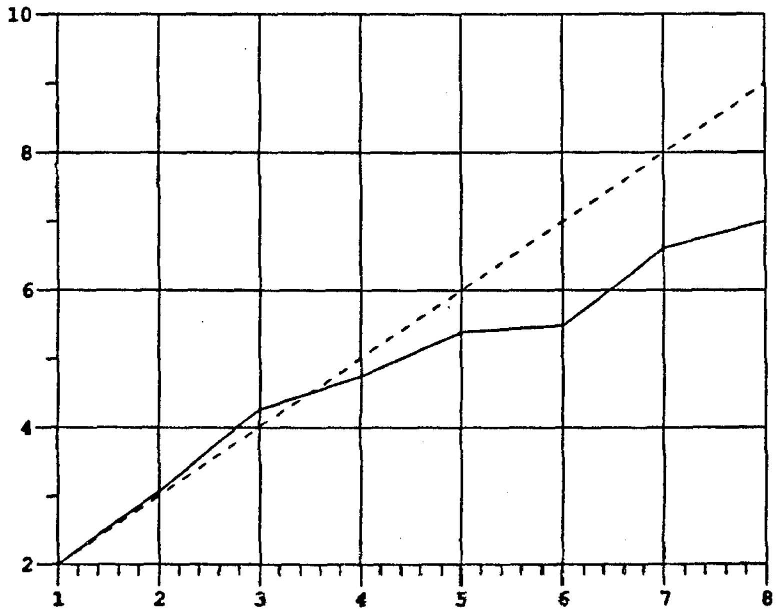


Figure 7.6: Thick line: Average increase in domains vs. Partition. Dashed line: Linear growth vs. Partition.

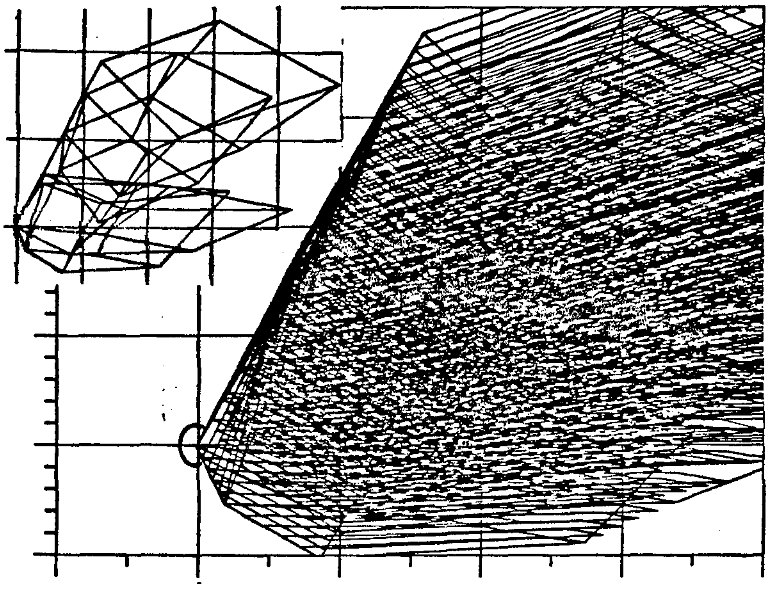


Figure 7.7: Multilinear complex function image, *offending* perturbation at a vertex. Upper-left: Image of edges of hypercube.

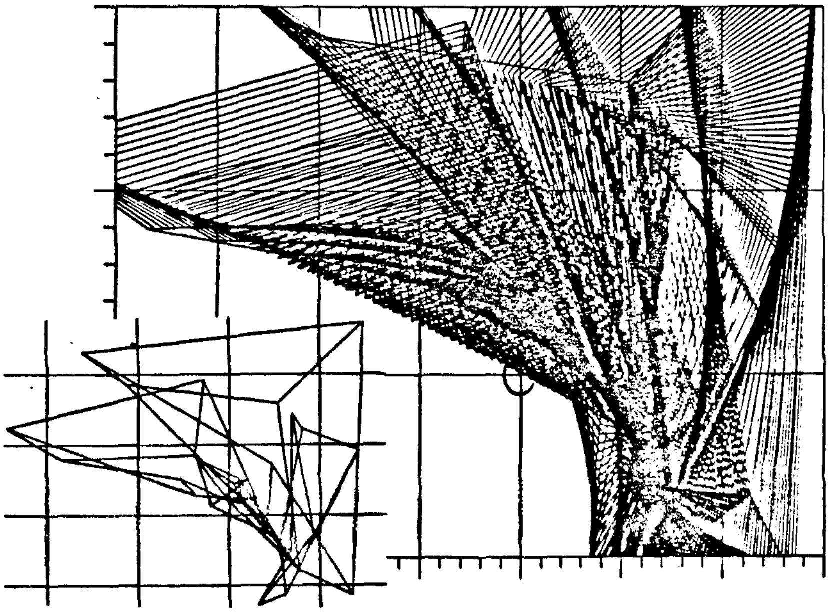


Figure 7.8: Multilinear complex function image, *offending* perturbation at an edge. Lower-left: Image of edges of hypercube.

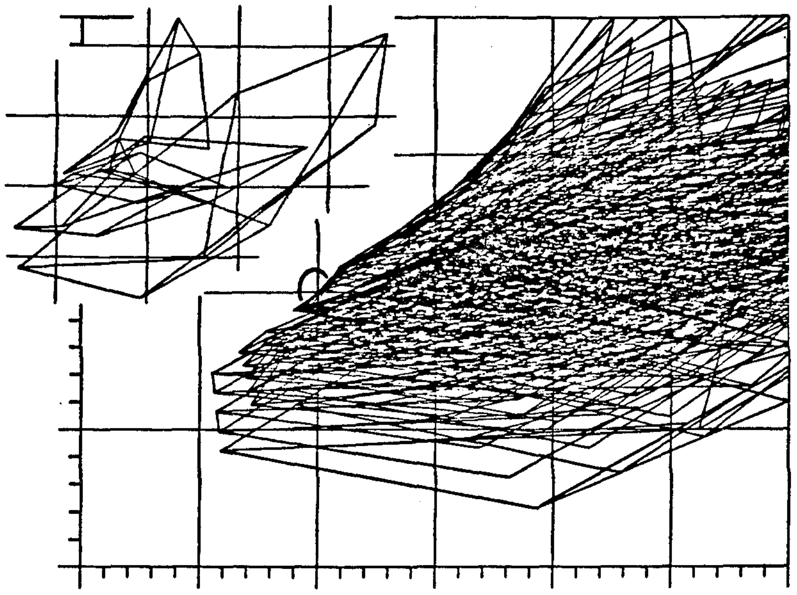


Figure 7.9: Multilinear complex function image, *offending* perturbation at a proper face. Upper-left: Image of edges of hypercube.

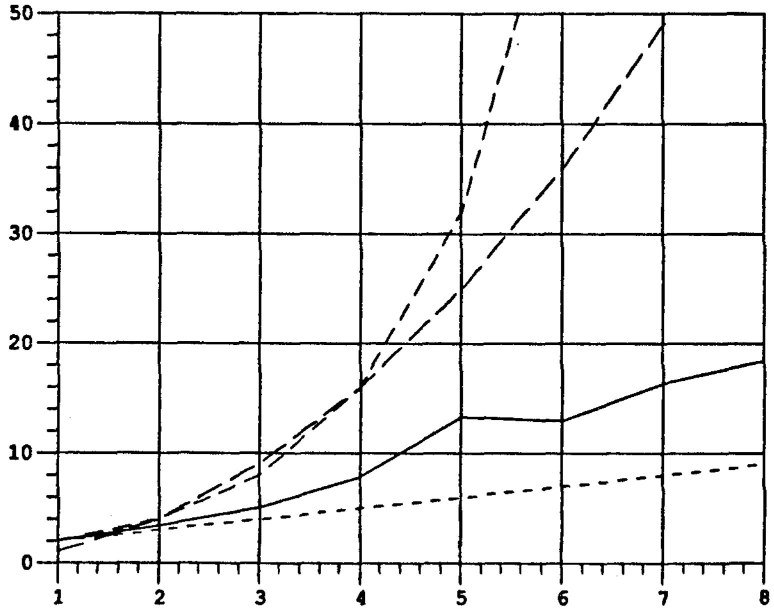


Figure 7.10: Thick line: Average increase in domains **vs.** Partition. Small dash line: Linear growth **vs.** Partition. Medium dash line: Exponential growth **vs.** Partition. Large dash line: Parabolic growth **vs.** Partition.

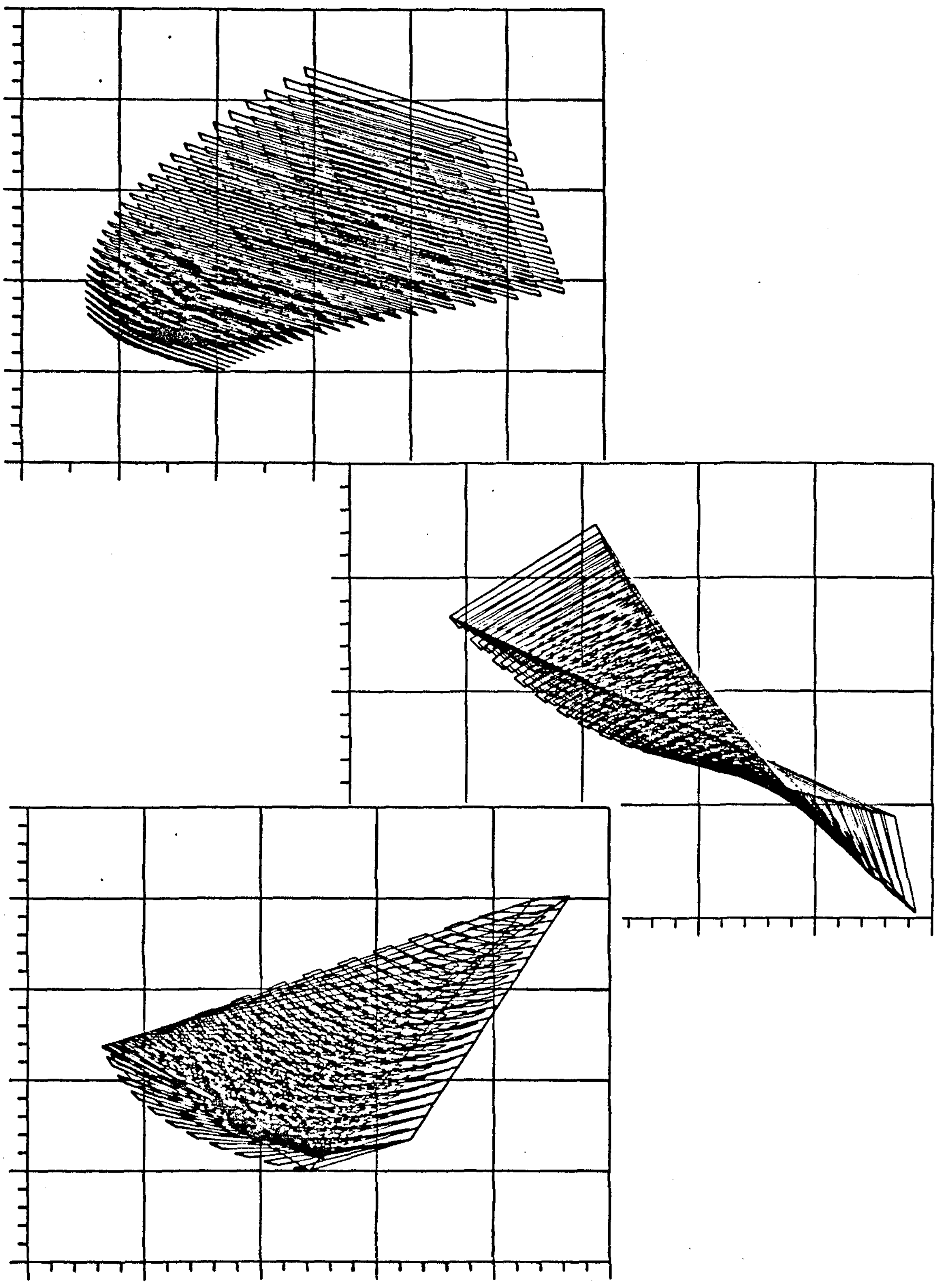


Figure 7.11: Polynomial complex function images.

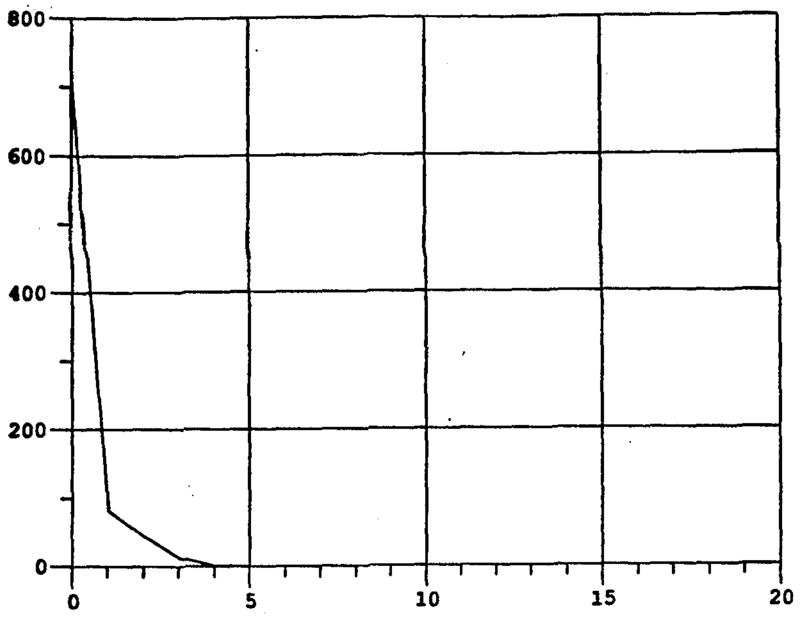


Figure 7.12: No. of samples **vs.** No. of iterations to find *critical* vertices.

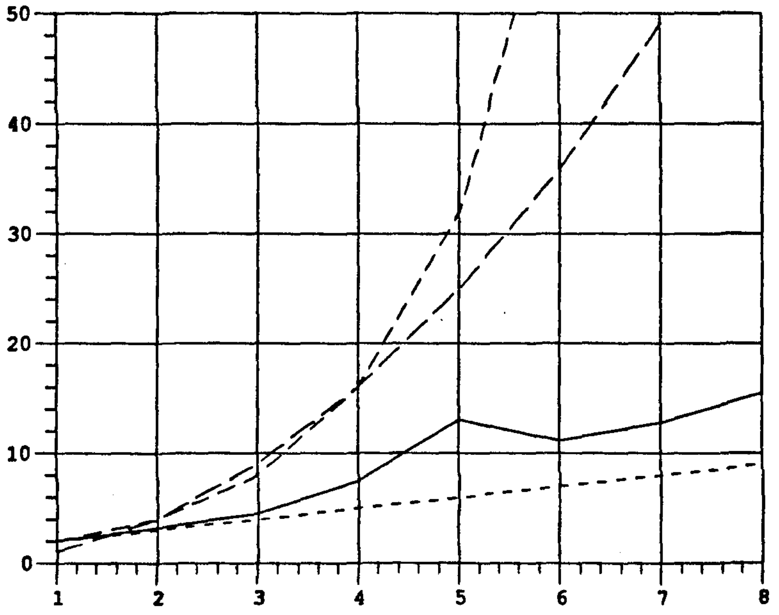


Figure 7.13: Thick line: Average increase in domains **vs.** Partition. Small dash line: Linear growth **vs.** Partition. Medium dash line: Exponential growth **vs.** Partition. Large dash line: Parabolic growth **vs.** Partition.

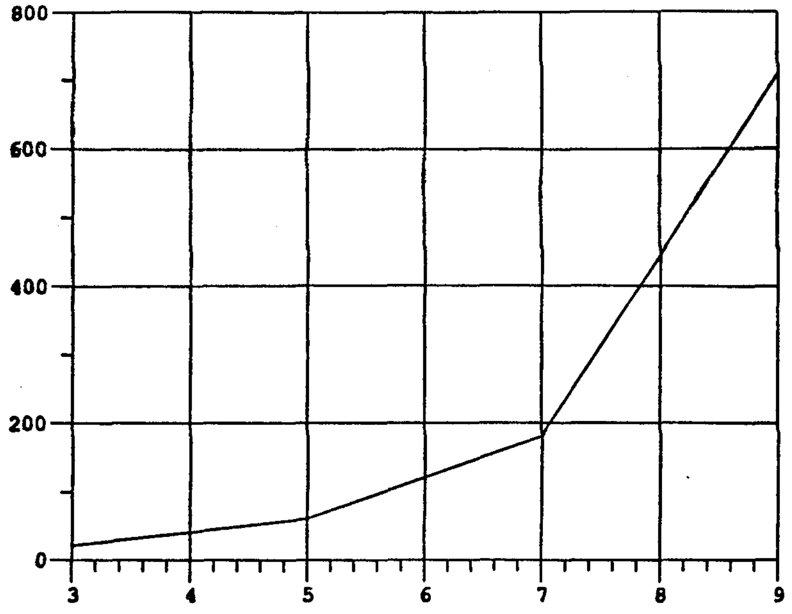


Figure 7.14: Average total computation time **vs.** No. of parameters. Multilinear complex case.

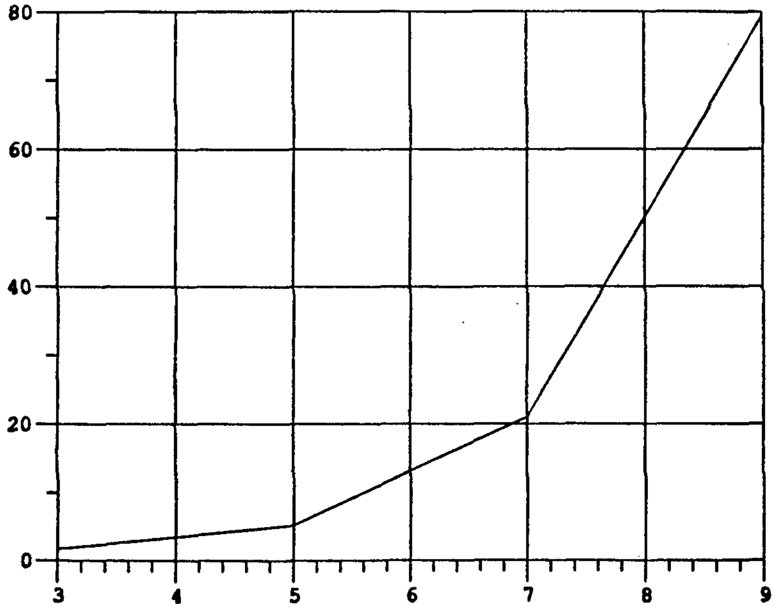


Figure 7.15: Average total computation time **vs.** No. of parameters. Related real case.

Chapter 8

X29 aircraft example

8.1 INTRODUCTION

In this chapter the program that computes the Multivariable Stability Margin has been applied to the analysis of the NASA X29 experimental aircraft. The lateral-directional control system design has been selected for this analysis because it represents a true multivariable control loop with coupling between roll and yaw axis. The longitudinal axis control system instead consists basically of two independent SISO loop systems.

This aircraft was analyzed in Chapter 2 using a simplified model description at Mach 0.6 and $15,000$ ft. flight condition. Now the complete model of plant and controller have been taken into account to test for robust stability and will be detailed in section 8.2. In Section 8.3 we explain the results of this analysis.

8.2 MODEL

Before presenting the equations of plant and controller, the terminology[48] throughout this chapter is defined according to figure 8.1.

a_Y	:	Lateral acceleration (g)
b	:	Reference span (ft)
C_l, C_n	:	Coefficients of roll and yaw moments
C_Y	:	Coefficient of lateral force
g	:	Acceleration of gravity (32.2 ft/s^2)

I_x, I_y, I_z :	Roll, pitch and yaw axis moments of inertia (slug-ft^2)
I_{xz} :	Roll and yaw axis cross inertia (slug-ft^2)
H :	Altitude (feet)
m :	Mass (slugs)
p, q, r :	Roll, pitch and yaw rates (rad/s)
$\bar{q}$:	Dynamic pressure (lb/ft^2)
S :	Reference area (ft^2)
T :	Sampling period (0.025 s)
V :	Velocity (ft/s)
α :	Angle of attack (rad)
β :	Sideslip angle (rad)
δ_a, δ_r :	Aileron and rudder deflection (rad)
θ, ϕ :	Pitch and roll attitude (rad)

The lateral-directional model for this aircraft was obtained from NASA[48] and is described by the following equations:

$$\begin{aligned}
\dot{\beta} &= \frac{\bar{q}S}{mV}C_Y + p\sin\alpha - r\cos\alpha + \frac{g}{V}\sin\phi\cos\theta \\
\dot{p}I_x - \dot{r}I_{xz} &= \bar{q}SbC_l + rq(I_y - I_z) + pqI_{xz} \\
\dot{r}I_z - \dot{p}I_{xz} &= \bar{q}SbC_n + pq(I_x - I_y) - rqI_{xz} \\
\dot{\phi} &= p + q\tan\theta\sin\phi + r\tan\theta\cos\phi \\
a_y &= \frac{\bar{q}S}{mg}C_Y - \frac{Z_{a_y}}{g}\dot{p} + \frac{X_{a_y}}{g}\dot{r}
\end{aligned} \tag{8.1}$$

where

$$\begin{aligned}
C_Y &= C_{Y_\beta}\beta + C_{Y_{\delta a}}\delta_a + C_{Y_{\delta r}}\delta_r \\
C_l &= C_{l_\beta}\beta + C_{l_{\delta a}}\delta_a + C_{l_{\delta r}}\delta_r + C_{l_p}\frac{pb}{2V} + C_{l_r}\frac{rb}{2V} \\
C_n &= C_{n_\beta}\beta + C_{n_{\delta a}}\delta_a + C_{n_{\delta r}}\delta_r + C_{n_p}\frac{pb}{2V} + C_{n_r}\frac{rb}{2V}
\end{aligned} \tag{8.2}$$

are the aerodynamic coefficient equations. The subscripts β, δ_a and δ_r are the variables with respect to which the derivatives of C_Y, C_l and C_n are taken, e.g. C_{Y_β} represents the derivative of the lateral force coefficient with respect to sideslip angle. These derivatives are obtained by usual parameter identification methods applied to the experimental data. From this procedure we will obtain an error bound on some of the parameters, in particular in the C_l and C_n equations.

The state space form of the above equations considering the input from the differential flap and aileron as $\mathbf{u} = [\delta_a \quad \delta_r]^t$ and the output as $\mathbf{y} = [p \quad r \quad \phi \quad a_Y]^t$ is:

$$\begin{aligned} E\dot{\mathbf{x}} &= \tilde{A}\mathbf{x} \quad + \quad \tilde{B}\mathbf{u} \\ \mathbf{y} &= C\mathbf{x} \quad + \quad D\mathbf{u} \end{aligned} \tag{8.3}$$

the realization being $\left[E^{-1}\tilde{A}, E^{-1}\tilde{B}, C, D\right]$. These matrices contain explicitly the uncertain parameters $C_{l_\beta}, C_{n_\beta}, C_{l_{\delta a}}, C_{n_{\delta a}}, C_{l_{\delta r}}$ and $C_{n_{\delta r}}$ as follows:

$$\begin{aligned} E &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & I_x & -I_{xz} & 0 \\ 0 & -I_{xz} & I_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \tilde{A} &= \begin{bmatrix} q_m C_{Y_\beta} & \sin\alpha & -\cos\alpha & \frac{g}{V}\cos\theta \\ q_s C_{l_\beta} & -\tilde{a}_{22} & \tilde{a}_{23} & 0 \\ q_s C_{n_\beta} & \tilde{a}_{32} & -\tilde{a}_{33} & 0 \\ 0 & 1 & \tan\theta & \tilde{a}_{44} \end{bmatrix} \\ \tilde{B} &= \begin{bmatrix} q_m C_{Y_{\delta a}} & q_m C_{Y_{\delta r}} \\ q_s C_{l_{\delta a}} & q_s C_{l_{\delta r}} \\ q_s C_{n_{\delta a}} & q_s C_{n_{\delta r}} \\ 0 & 0 \end{bmatrix} \\ C &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ q_g C_{Y_\beta} & \frac{-Z_{a_Y}}{g} & \frac{X_{a_Y}}{g} & 0 \end{bmatrix} \\ D &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ q_g C_{Y_{\delta a}} & q_g C_{Y_{\delta r}} \end{bmatrix} \end{aligned} \tag{8.4}$$

The state is $\mathbf{x} = [\beta \quad p \quad r \quad \phi]^t$ and the following definitions hold,

$$q_m = \bar{q}S/mV \qquad q_g = \bar{q}S/mg \qquad q_s = \bar{q}Sb \qquad q_v = \bar{q}Sb^2/2V \tag{8.5}$$

$$\tilde{a}_{22} = -\left(q_v C_{l_p} + qI_{xz}\right) \qquad \tilde{a}_{23} = q_v C_{l_r} + q\left(I_y - I_z\right) \tag{8.6}$$

$$\tilde{a}_{32} = q_v C_{n_p} + q\left(I_x - I_y\right) \qquad \tilde{a}_{33} = -\left(q_v C_{l_r} - qI_{xz}\right) \qquad \tilde{a}_{44} = q\tan\theta \tag{8.7}$$

The four open loop poles we obtain from the above model are the following. The "dutch roll" is a complex conjugate pair of poles that apply to both the yaw and roll axis. The "roll mode" is a stable real pole and finally an unstable low frequency pole, defined as the "spiral mode".

The controller structure for all flight conditions is the following (see figure 8.2):

$$K(s) = K_a(s) \cdot K_b(s) \tag{8.8}$$

with

$$K_a(s) = \begin{bmatrix} A_f(s)Z(s)\left[(X_{kp3} + X_{ki3}T/2) + X_{ki3}/s\right] & 0 \\ 0 & A_r(s)Z(s)X_{kp4} \end{bmatrix} \tag{8.9}$$

and

$$\begin{aligned}
(K_b)_{11} &= P_1(s) \cdot S_p(s) \cdot [K_2 - \alpha \cdot \text{Blend} \cdot L(s) \cdot K_3] \\
(K_b)_{12} &= K_3 \cdot S_r(s) \cdot F_r(s) \cdot P_1(s) \cdot L(s) \\
(K_b)_{13} &= -K_3 \cdot P_2(s) \cdot L(s) \cdot g \cdot \text{Blend}/V \\
(K_b)_{14} &= K_4 \cdot F_a(s) \cdot P_1(s) \\
(K_b)_{21} &= P_1(s) \cdot S_p(s) \cdot [K_{16} - \alpha \cdot \text{Blend} \cdot L(s)] \\
(K_b)_{22} &= K_{17} \cdot S_r(s) \cdot F_r(s) \cdot P_1(s) \cdot L(s) \\
(K_b)_{23} &= -K_{17} \cdot P_2(s) \cdot L(s) \cdot g \cdot \text{Blend}/V \\
(K_b)_{24} &= K_{18} \cdot F_a(s) \cdot P_1(s)
\end{aligned} \tag{8.10}$$

The elements of each matrix are the following:

$A_f(s)$	:	4th order differential flap actuator model.
$A_r(s)$	:	4th order rudder actuator model.
$Z(s)$	:	Padé approximation for time delay due to sampler & hold.
$P_1(s)$	:	Roll, yaw and lateral acceleration channels prefilter.
$P_1(s)$	:	Roll attitude channel prefilter.
$S_p(s)$	:	2nd order roll rate sensor model.
$S_r(s)$	:	2nd order yaw rate sensor model.
$F_r(s)$	:	2nd order yaw channel notch filter.
$F_a(s)$	:	2nd order latteral acceleration channel notch filter.
$L(s)$	:	Digital filter L(z) transformed via Tustin to analog model.

8.3 RESULTS

The robust stability analysis was made over two different flight conditions. The first one is the nominal linearization point for designing the controller structure. The second one is a critical flight situation at sea level and with high angle of attack. This last condition has only been checked through simulations, but there has not yet been any flight tests. Both conditions can be seen in Table VIII.1.

	No. 1	**No. 2**
Mach No.:	0.9	0.4
Altitude:	30,000 feet	sea level
α:	3.78°	30°
V:	895 ft/s	447 ft/s
θ:	3.81°	−60°
q:	0 °/s	24.62 °/s
$\bar{q}$:	357 lb/ft^2	237 lb/ft^2

Table VIII.1

A different linear model is obtained at each flight condition and although the structure of the plant and controller models does not change, their parameter values do. In the controller these parameters are K_2 through $K_{18}, X_{kp3}, X_{ki3}, X_{kp4}$ and Blend. In the plant there are also changes in all aerodynamic coefficients derivatives. Among them, the uncertain parameters will have different nominal values and error bounds.

To apply the program computing the stability margin to this example, we find the closed loop characteristic polynomial $f(s,\Delta)$ as a function of the uncertain parameters mentioned before. This is:

$$\begin{aligned} G(s,\Delta) &= D + C\left[sE - \tilde{A}(\Delta)\right]^{-1}\tilde{B}(\Delta) \\ f(s,\Delta) &= \det\left[I + K(s)G(s,\Delta)\right] \\ \Delta &= \operatorname{diag}\left[\delta_1 \ldots \delta_n\right] \end{aligned} \tag{8.11}$$

were δ_i are the error bounds of parameter p_i around the nominal value p_{oi}. In this procedure we need to keep track of the uncertain parameters in a symbolic way, to follow the exponents of each parameter in the final equation. This is important in defining the canonical constraints as in equation (2.7). In this case the function $f(\cdot)$ is linear in all parameters except the first two ones which are squared. The generalized set of parameters and equality contraints are:

$$\begin{aligned} p_1 = p_2 = C_{l_\beta} &= p_{o1} \pm \delta_1 = p_{o2} \pm \delta_2 \\ p_3 = p_4 = C_{n_\beta} &= p_{o3} \pm \delta_3 = p_{o4} \pm \delta_4 \\ p_5 = C_{l_{\delta a}} &= p_{o5} \pm \delta_5 \\ p_6 = C_{n_{\delta a}} &= p_{o6} \pm \delta_6 \\ p_7 = C_{l_{\delta r}} &= p_{o7} \pm \delta_7 \\ p_8 = C_{n_{\delta r}} &= p_{o8} \pm \delta_8 \end{aligned} \tag{8.12}$$

The nominal values will be different at each flight condition and the variations around the nominals will be:

$$\begin{aligned} &\delta_1 = \delta_2 = 33.3\% \qquad \delta_3 = \delta_4 = 13.7\% \\ &\delta_5 = 12.8\% \quad \delta_6 = 58.0\% \quad \delta_7 = 18.9\% \quad \delta_8 = 14.3\% \end{aligned} \tag{8.13}$$

for the first linearization point and

$$\begin{aligned} &\delta_1 = \delta_2 = 33.3\% \qquad \delta_3 = \delta_4 = 11.1\% \\ &\delta_5 = 32.0\% \quad \delta_6 = 63.6\% \quad \delta_7 = 685\% \quad \delta_8 = 22.6\% \end{aligned} \tag{8.14}$$

for the second one.

The multivariable stability margin $k_m(\omega)$ was computed in both cases at several frequency points. The plots of k_m versus frequency can be seen in figures 8.3 and 8.4. The minimum value was $k_m = 3.15$ achieved at $\omega = 3.5$ rad/s in the first case and $k_m = 2.89$ at $\omega = 10$ rad/s in the second one, thus obtaining a robustly stable closed loop system in both situations. The mapping of the closed loop characteristic polynomial for these two frequencies can be seen in figures 8.5 and 8.6 respectively.

From the program we can also obtain the combination of parameters p^* whose image first reaches the origin of the complex plane. This is a very important information in the analysis and redesign (if necessary) of the feedback system. Ideally, by designing the controller for a nominal plant that has the set of parameters p^*, the achievement of nominal stability will guarantee robust stability as well.

The critical combination of Δ's in each case is:

$$\begin{aligned} \Delta_1 &= \operatorname{diag}[\delta_1 \quad \delta_2 \quad \delta_3 \quad \delta_4 \quad -(0.86\cdot\delta_5) \quad -\delta_6 \quad -\delta_7 \quad \delta_8] \\ \Delta_2 &= \operatorname{diag}[\delta_1 \quad \delta_2 \quad -\delta_3 \quad -\delta_4 \quad -\delta_5 \quad (0.18\cdot\delta_6) \quad \delta_7 \quad \delta_8] \end{aligned} \tag{8.15}$$

The actual design of $K(s)$ was made in both cases for the set of parameters $p_N = p_o + \Delta_N$, where Δ_N is

$$\Delta_N = \operatorname{diag}[-\delta_1 \quad -\delta_2 \quad \delta_3 \quad \delta_4 \quad -\delta_5 \quad -\delta_6 \quad \delta_7 \quad -\delta_8] \tag{8.16}$$

As we can see this disagrees with the values obtained in (8.15) at both flight conditions. This is due to the fact that it is difficult to develop a physical "intuition" on how the variation of parameters will influence the stability of the feedback system, mainly in systems with a large number of parameters like this one. Another important reason is that, for simplicity, the design of the controller at each linearization point is made for a nominal set of parameters of the plant chosen at the same extreme point of the uncertainty interval.

Some final remarks follow. At both flight conditions the nominal closed loop transfer function has unstable poles. In the first case, the unstable pole is the usual "spiral" mode which is slow enough to be controlled by the pilot. In the second linearization point, the closed loop **linear** system has a fast unstable mode but through simulations it was found that the **nonlinear** model around this linearization point has slow diverging states which again are controllable by the pilot.

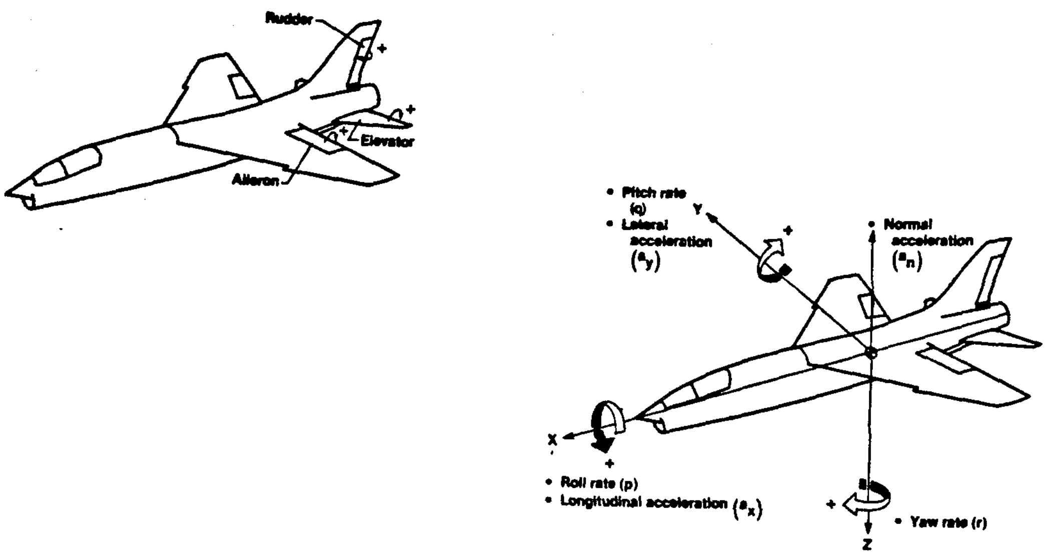

Rudder
Elevator
Aileron
Pitch rate (q)
Lateral acceleration (a_y)
Y
Normal acceleration (a_n)
X
Roll rate (p)
Longitudinal acceleration (a_x)
Yaw rate (r)
Z

Figure 8.1: General aircraft body axis system and control surfaces.

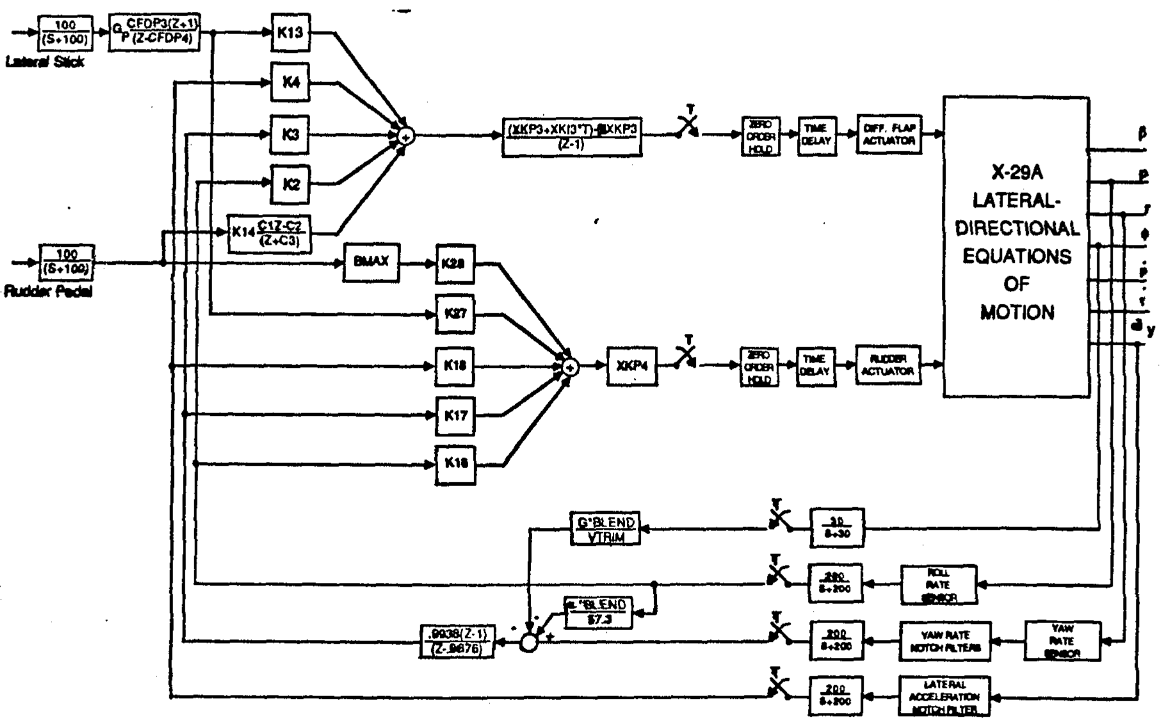


Figure 8.2: Control system block diagram.

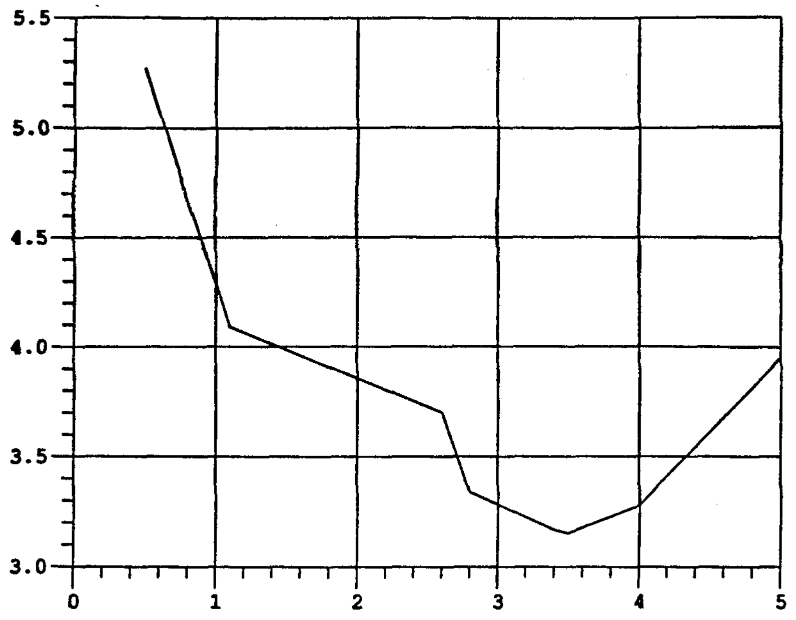


Figure 8.3: First linearization point. Multivariable stability margin vs. frequency.

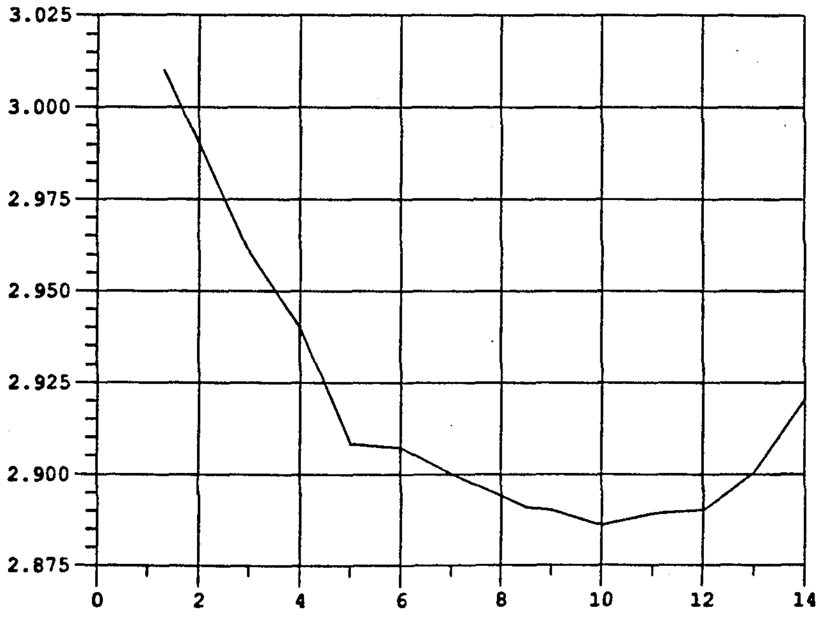
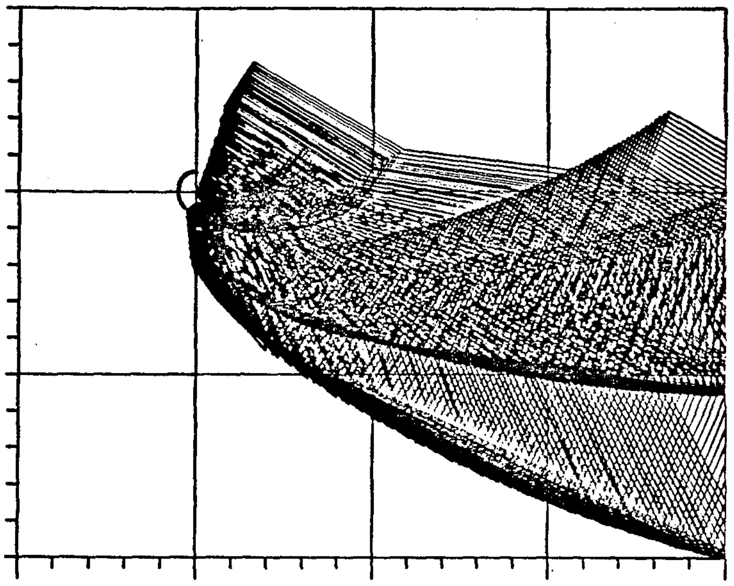


Figure 8.4: Second linearization point. Multivariable stability margin vs. frequency.



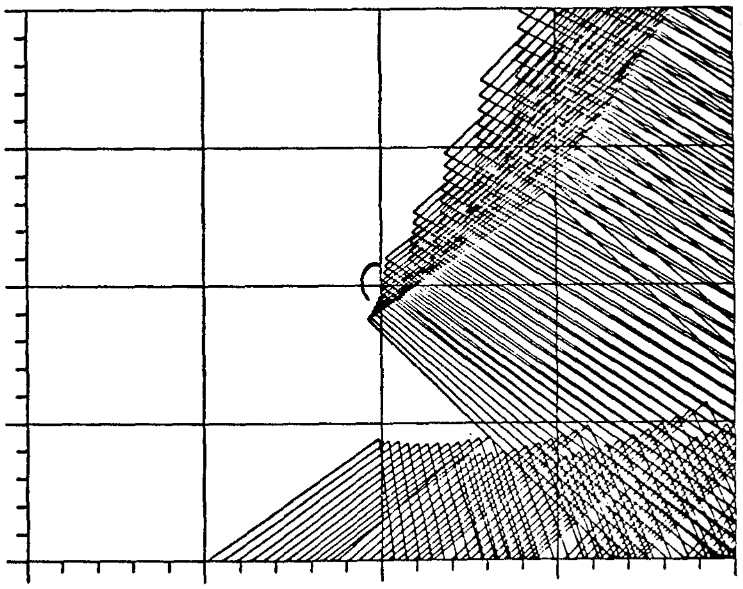
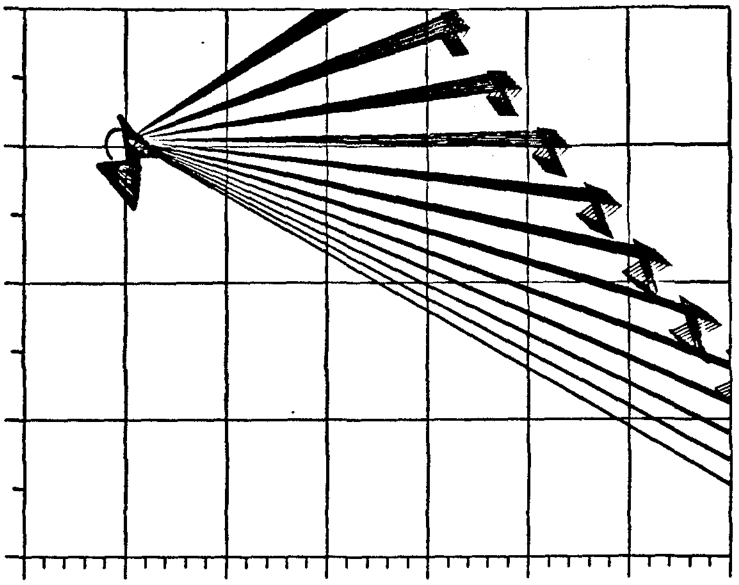


Figure 8.5: First linearization point. Above: Image of Closed loop characteristic polynomial at minimum $k_m(j\omega)$. Below: Detailed version.



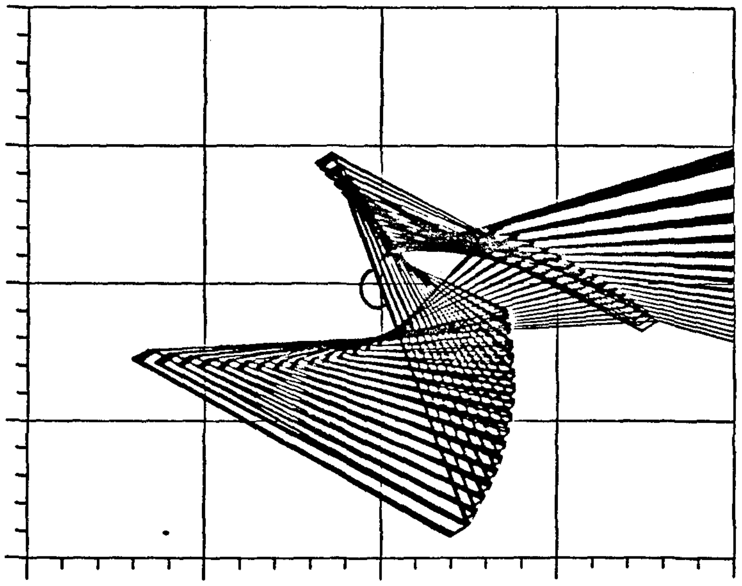


Figure 8.6: Second linearization point. Above: Image of Closed loop characteristic polynomial at minimum $k_m(j\omega)$. Below: Detailed version.

Chapter 9

Future research

The main objective of this thesis is to develop the theoretical and computer implementation aspects of the analysis of FDLTI systems with general uncertainty structures. A general algorithm that computes the multivariable stability margin has been implemented and an alternative version has been developed which computes the same margin at a lower computational cost. This increase in the speed is obtained not only by eliminating the frequency search, but also because the computation over the particular real functions obtained from the Routh-Hurwitz procedure will have a high probability of obtaining vertices of the parameter hypercube as the offending perturbations.

Through a statistical analysis on the performance of the algorithm in both implementations it has also been determined that the partition of domains is not the main factor responsible for having an exponential time procedure in the parameters. Although we obtain a significant improvement in the computational speed, the problem of obtaining a polynomial time algorithm that could perform the same task still remains unsolved. This seems to be a nontrivial problem because even eliminating the upper bound search, we remain with the computation of all vertices which by itself makes the overall time exponential in the parameters. An open research problem would be to find a method that would eliminate part of these vertices by a judicious analysis (in polynomial time) of the function to be mapped. This is the main objective of chapter 4 where a method is developed to determine a smaller number of variables over the whole hypercube among which we can find the actual offending parameter combination. Although a great simplification can be obtained in many cases through that analysis, practical necessary and sufficient conditions that could determine when a set of parameters are ELB (see definition 4.1) and which among them are the relevant ones to determine robust stability has not yet been solved.

In terms of general uncertainty structures involving not only parametric, but also dynamic structured uncertainty, a search method is proposed that uses the improved algorithm mentioned before as the main tool for computations. Although the method gives the exact result for three or less dynamic blocks, the statement and computation of the problem for high order systems can be tedious. An automated procedure should be developed that would use symbolic manipulation programs as a means to improve this situation.

Finally, the problem of synthesis is yet unsolved for these general uncertainty structures which seems to be also a nontrivial question.

References

[1] J. C. Doyle, "Robustness of multiloop feedback systems", *Proceedings of the CDC*, San Diego, 1978.

[2] M. G. Safonov, M. Athans,"Gain and phase margins for multiloop LQG regulators", *IEEE Transactions on Automatic Control*, vol. AC-22, 1977.

[3] J. C. Doyle, G. Stein,"Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis", *IEEE Transactions on Automatic Control*, vol. AC-26, 1981.

[4] M. G. Safonov, A. J. Laub, G. L. Hartmann, "Feedback Properties of Multivariable Systems: The Role and Use of the Return Difference Matrix", *IEEE Transactions on Automatic Control*, vol. AC-26, 1981.

[5] G. Zames, "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms and approximate inverses", *IEEE Transactions on Automatic Control*, vol. AC-26, 1981.

[6] M. G. Safonov, M. S. Verma, "Multivariable L^∞ sensitivity optimization and Hankel approximation", *Proceedings of the ACC*, 1983.

[7] K. Glover, "All optimal Hankel norm approximations of linear multivariable systems and their L^∞ error bounds", *International Journal of Control*, vol. 39, 1984.

[8] J. C. Doyle, "Lecture Notes on Advances in Multivariable Control", *ONR Honeywell Workshop*, Minneapolis, 1984.

[9] J. C. Doyle, K. Glover, P. Khargonekar, B. A. Francis, "State-space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems", *Proceedings of the ACC*, Georgia, 1988.

[10] M. G. Safonov,"Tight Bounds on the Response of Multivariable Systems with

Component Uncertainty",*Proceedings of the.* 16^{th} *Allerton Conference*, Illinois, 1978.

[11] R. R. de Gaston, *Nonconservative Calculation of the Multiloop Stability Margin*, Ph.D. Dissertation, University of Southern California, 1985.

[12] R. R. de Gaston, M. G. Safonov, "Exact Calculation of the Multiloop Stability Margin", *IEEE Transactions on Automatic Control*, vol. AC-33, 1988.

[13] A. Sideris, R. R. de Gaston,"Multivariable Stability Margin Calculation with Uncertain Correlated Parameters", *Proceedings of the CDC*, Greece, 1986.

[14] R. S. Sánchez Peña, A. Sideris,"A general program to compute the Multivariable Stability Margin for systems with real parametric uncertainty", *Proceedings of the ACC*, Georgia, 1988.

[15] R. M. Biernacki, H. Hwang, S. P. Bhattacharyya, "Robust Stability with Structured Real Parameter Perturbations", *IEEE Transanctions on Automatic Control*, vol. AC-32, 1987.

[16] J. C. Doyle,"Analysis of Feedback Systems with Structured Uncertainty", *IEE Proc.*, Vol. 129, pt. D, No. 6, 1982.

[17] M. K. H. Fan, A. L. Tits,"Characterization and efficient computation of the Structured Singular Value", *IEEE Transactions on Automatic Control*, vol. AC-31, 1986.

[18] M. K. H. Fan, A. L. Tits,"m-Form numerical range and the computation of the Structured Singular Value", *IEEE Transactions on Automatic Control*, vol. AC-33, 1988.

[19] A. Packard, *What's new with μ: Structured uncertainty in multivariable control*, Ph.D. Dissertation, University of California at Berkeley, 1988.

[20] V. L. Kharitonov, "Asymptotic stability of an equilibrium position of a family of linear differential equations", *Differencialnye Uravneniya*, vol. 14, 1978.

[21] B. R. Barmish, "Invariance of the strict Hurwitz property for polynomials with perturbed coefficients", *IEEE Transactions on Automatic Control*, vol. AC-29, 1984.

[22] S. Bialas, J. Garloff, "Stability of polynomials under coefficient perturbations", *IEEE Transactions on Automatic Control*, vol. AC-30, 1985.

[23] B. D. O. Anderson, E. I. Jury and M. Mansour, "On robust Hurwitz polynomials", *IEEE Transactions on Automatic Control*, vol. AC-32, 1987.

[24] E. R. Panier, M. K. H. Fan, A. L. Tits, "On the stability of polynomials with uncoupled perturbations in the coefficients of even and odd powers", submitted to *Systems and Control Letters*, 1987.

[25] A. C. Bartlett, C. V. Hollot, H. Lin, "Root locations of an entire polytope of polynomials: It suffices to check the edges", *Mathematics of Control, Signals and Systems*, Springer-Verlag 1988.

[26] M. Fu, B. R. Barmish, "A generalization of Kharitonov's interval polynomial framework to handle linearly dependent uncertainties", submitted for publication, 1987.

[27] B. R. Barmish, "Kharitonov's theorem and its extensions and applications: An introduction", *Proceedings of the CDC*, Los Angeles, 1987.

[28] K. H. Wei, R. K. Yedavalli, "Robust stabilizability for linear systems with both parameter variation and unstructured uncertainty", *Proceedings of the CDC*, Los Angeles, 1987.

[29] C. V. Hollot, D. P. Looze, A. C. Bartlett, "Unmodelled dynamics: Performance and stability via parameter space methods", *Proceedings of the CDC*, Los Angeles, 1987.

[30] M. Fan, A. Tits, J. C. Doyle, "On Robustness under parametric and dynamic uncertainties", *Proceedings of the ACC*, Georgia, 1988.

[31] J. R. Rice, The Approximation of Functions, Reading, MA: Addison-Wesley, 1969.

[32] E. J. Routh, Dynamics of a System of Rigid Bodies, McMillan, New York, 1892.

[33] A. Hurwitz, "On the conditions under which an equation has only roots with negative real parts", *Matematische Annalen*, vol. 46, 1895.

[34] A. Sideris, "An efficient procedure to check the robust stability of polynomials with coefficients in a polytope", in preparation.

[35] M. Fan, A. Sideris, R. Sánchez Peña, J. C. Doyle, "Generic property of uncertain polynomials multilinear in the parameters", in preparation.

[36] F. R. Gantmacher, Matrix Theory - vol. II, N.Y. Chelsea, 1959.

[37] L. Zadeh, C. A. Desoer, Linear System Theory, N.Y. McGraw-Hill, 1963.

[38] K. S. Yeung, S. S. Wang, "A simple proof of Kharitonov's Theorem", *IEEE Transactions on Automatic Control*, vol. AC-32, 1987.

[39] J. Ackermann, "Parameter design of robust control systems", *IEEE Transactions on Automatic Control*, vol. AC-25, 1980.

[40] J. C. Doyle, A. Packard,"Uncertain Multivariable Systems from a State Space perspective", *Proceedings of the ACC*, Minnesota, 1987.

[41] E. I. Jury, J. Blanchard, "A Stability test for linear discrete time systems in table form", *Proceedings of the IRE*, vol. 49, 1961.

[42] J. Schur, "Über Potenzreihen, die im Inneren des Einheitskreises beschänkt sind, II", *Zeitschrift für di4e reine angewandte Matematik*, vol. 148, 1918.

[43] A. Cohn, Über die Anzahl der Wurzeln einer algebraischen Gleichung in einem Kreise", *Matematische Zeitschrift*, vol. 14, 1922.

[44] Franklin, Powell, Emami-Naeini, "Feedback Control of Dynamic Systems", Addison-Wesley, 1986.

[45] M. G. Safonov, J,,. C. Doyle, "Minimizing conservativeness of robustness sin-

gular values", in Multivariable Control, *D. Reidel Publishing Company*, 1984.

[46] J. Jahn, E. Sachs, "Generalized quasiconvex mappings and vector optimization", *SIAM J. Control & Optimization*, vol. 24, 1986.

[47] O. L. Mangasarian, Nonlinear Programming, McGraw-Hill, N.Y., 1969.

[48] R. E. Maine, K. W. Iliff, "Application of Parameter Estimation to Aircraft Stability and Control: The Output error approach", *NASA Reference Publication 1168*, 1986.

[49] A. Sideris, R. S. Sánchez Peña, "Fast computation of the Multivariable Stability Margin for real interrelated uncertain parameters", *Proceedings of the ACC*, Georgia, 1988 and submitted to the IEEE Transactions on Automatic Control, 1987.

[50] A. Sideris, R. S. Sánchez Peña, "Robustness margin calculation with dynamic and real parametric uncertainty", *Proceedings of the ACC*, Georgia, 1988 and submitted to the IEEE Transactions on Automatic Control, 1988.

[51] R. S. Sánchez Peña, A. Sideris, "Robustness with real parametric and structured complex uncertainty", submitted to the 27th *CDC*, Texas, 1988 and to the IEEE Transactions on Automatic Control, 1988.