

AD-A200 408

A TRIDENT SCHOLAR PROJECT REPORT

NO. 150

DTIC FILE COPY

DEVELOPMENT OF STABILITY/ROBUSTNESS CONSIDERATIONS
FOR CONTROL SYSTEM DESIGN WITH MULTIPLE
INPUT/MULTIPLE OUTPUT PLANTS



UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

This document has been approved for public
release and sale; its distribution is unlimited.

DTIC
ELECTE
NOV 03 1988
S H D

06 11 02 004

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER U.S.N.A. - TSPR; no. 150 (1988)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DEVELOPMENT OF STABILITY/ROBUSTNESS CONSIDERATIONS FOR CONTROL SYSTEMS DESIGN WITH MULTIPLE INPUT/ MULTIPLE OUTPUT PLANTS.		5. TYPE OF REPORT & PERIOD COVERED Final 1987/88
7. AUTHOR(s) Daniel J. Hurdle		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS United States Naval Academy, Annapolis.		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS United States Naval Academy, Annapolis.		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE 10 June 1988
		13. NUMBER OF PAGES 167
		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) This document has been approved for public release; its distribution is UNLIMITED.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Accepted by the U.S. Trident Scholar Committee.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Automatic control Feedback control systems		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) An investigation into stable, robust control system design with multiple input/multiple output (MIMO) plants was conducted. Stability/robustness is identified as the first and primary source of concern in MIMO control system design and thus is the focus of research. Performance/robustness requirements and the meeting of additional performance specifications are largely left for future research. A design example is presented, however, which incorporates the (OVER)		

DD FORM 1473
1 JAN 73

EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

meeting of certain performance criteria into the overall framework of achieving MIMO stability/robustness.

Methods which can be utilized to assess the stability/robustness properties of MIMO nominal plant models without control are developed and identified as the first step in the compensator (or controller) design process. The previously developed Model Based Compensator/Linear Quadratic Gaussian/Loop Transfer Recovery (MBC/LQG/LTR) method is adopted as the general framework for MIMO compensator design and numerous computer programs generated to implement it. These programs are written for use with the engineering software package PC-MATLAB. Original methods are formulated, to be used in conjunction with the MBC/LQG/LTR methodology, which provide the control engineer with the means to design for some measure of stability/robustness in the controlled system. Here, also, computer programs were written to implement theoretical developments. Finally, a standard design process was created using the above methodology for the design and testing of a MIMO control system based upon stability/robustness considerations.

9 MC
100 20

Accession For	
NTIS GPA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability	
Dist	
A-1	



S N 0102- LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

U.S.N.A. - Trident Scholar project report; no.150 (1988)

DEVELOPMENT OF STABILITY/ROBUSTNESS CONSIDERATIONS
FOR CONTROL SYSTEM DESIGN WITH MULTIPLE
INPUT/MULTIPLE OUTPUT PLANTS

A Trident Scholar Project Report

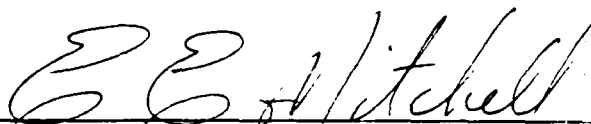
by

Midshipman First Class Daniel J. Hurdle, USN

Class of 1988

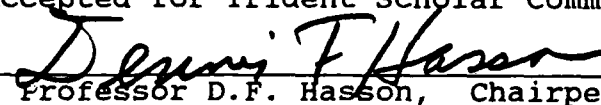
U. S. Naval Academy

Annapolis, Maryland



Prof. E.E. Mitchell, Chairman, WSE Dept.

Accepted for Trident Scholar Committee



Professor D.F. Hasson, Chairperson

10 June 1988

Date

USNA-1531-2

ABSTRACT

An investigation into stable, robust control system design with multiple input/multiple output (MIMO) plants was conducted. Stability/robustness is identified as the first and primary source of concern in MIMO control system design and thus is the focus of research. Performance/robustness requirements and the meeting of additional performance specifications are largely left for future research. A design example is presented, however, which incorporates the meeting of certain performance criteria into the overall framework of achieving MIMO stability/robustness.

Methods which can be utilized to assess the stability/robustness properties of MIMO nominal plant models without control are developed and identified as the first step in the compensator (or controller) design process. The previously developed Model Based Compensator/Linear Quadratic Gaussian/Loop Transfer Recovery (MBC/LQG/LTR) method is adopted as the general framework for MIMO compensator design and numerous computer programs generated to implement it. These programs are written for use with the engineering software package PC-MATLAB. Original methods are formulated, to be used in conjunction with the MBC/LQG/LTR methodology, which provide the control engineer with the means to design for some measure of stability/robustness in the controlled system. Here, also,

computer programs were written to implement theoretical developments. Finally, a standard design process was created using the above methodology for the design and testing of a MIMO control system based upon stability/robustness considerations.

TABLE OF CONTENTS

<u>Subject</u>	<u>Page</u>
Abstract.....	1
List of Figures.....	4
Notation.....	5
Introduction.....	6
Objectives.....	16
Research and Implementation.....	17
Part I: Methods to Assess Stability/Robustness of a MIMO Nominal Plant Model.....	18
Part II: A Review of the MBC/LQG/LTR MIMO Design Methodology.....	37
Part III: Compensator Design Based on Stability/Robustness Considerations.....	49
Part IV: Standard Design Process.....	61
Part V: A Design Example.....	70
Conclusions.....	82
Suggestions for Future Research.....	84
Acknowledgements.....	85
References.....	86
Appendix A.....	87
Appendix B.....	90

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
(1) Standard Control System Diagram.....	8
(2) SISO and MIMO Plant Examples.....	9
(3) Open Loop Plant Diagram.....	19
(4) Closed Loop Plant Diagram.....	19
(5) s Plane Diagram.....	23
(6) Nominal Plant OL Singular Value Plot.....	25
(7) Singular Value Inequality Test Example Plot.....	29
(8) Deviation of Parameters Method Example Plot.....	35
(9) TFL Diagram.....	42
(10) Nominal Plant CL Singular Value Plot.....	42
(11) TFL CL Singular Value Plot.....	42
(12) MBC Diagram.....	44
(13) MBC with Plant Diagram.....	44
(14) TFL and MBC with Plant CL Singular Value Plots.....	46
(15) CL Singular Value Plot Matched at Low w	53
(16) $(sI - A_{cl})$ Singular Value Plot.....	53
(17) TFL Singular Value Inequality Example Plot 1.....	54
(18) TFL Singular Value Inequality Example Plot 2.....	54
(19) Nominal Plant Diagram.....	63
(20) TFL Diagram.....	63
(21) Plant with Control Diagram.....	63
(22) Generic Turbine Example Description.....	71
(23) Nominal Plant Singular Value Inequality Test Plot..	73
(24) Region of Stability Plot- 2 Parameters.....	73
(25) Nominal Plant Rotor Speed Step Response.....	74
(26) TFL Singular Value Inequality Test Plot.....	76
(27) TFL and MBC with Plant CL Singular Value Plots.....	77
(28) TFL Rotor Speed Step Response.....	78
(29) MBC with Plant Rotor Speed Step Response- Stable...	78
(30) MBC with Plant Rotor Speed Step Response- Unstable.	80

NOTATION

MIMO	Multiple Input/Multiple Output
SISO	Single Input/Single Output
A, B, C	Nominal Plant Matrices
A_{cl}	Nominal Plant Closed Loop A Matrix
A_{clh}	Target Feedback Loop Closed Loop A Matrix
(nom)	Nominal Plant Value
(act)	Actual Plant Value
G_p	Nominal Plant Open Loop Transfer Function Matrix
G_{cl}	Nominal Plant Closed Loop Transfer Function Matrix
G_{clh}	Target Feedback Loop Closed Loop Transfer Function Matrix
u	Plant Input
y	Plant Output
x	Plant State Variable
K	Compensator
σ	Singular Value
σ_{max}	Maximum Singular Value
σ_{min}	Minimum Singular Value
Eacl	Error Between the Nominal Plant A_{cl} and the Actual Plant A_{cl}
Ew	Worst Case Error
MBC/LQG/LTR	Model Based Compensator/Linear Quadratic Gaussian/Loop Transfer Recovery
DPM	Design Plant Model
TFL	Target Feedback Loop
H	Target Feedback Loop Design Parameter Matrix
G	Model Based Compensator Design Parameter Matrix
w	Frequency
s	Laplace Operator
I	Identity Matrix
	Determinant
() ⁻¹	Matrix Inverse
	Spectral or 2-Norm

INTRODUCTION

Most systems require one form of control or another. The system alone, which will be referred to here as the plant, may be of almost any type. As one might expect, anything from an airplane to a home heating unit to an automobile constitutes a plant. Not so expectedly, one may consider economies, large businesses, or even societies as plants despite the fact that they are essentially non-technical in nature.

Of importance to this investigation is the realization that not only do plants differ widely in type but that they also differ widely in complexity. The plants of the past with a single input and single output (SISO) are more and more giving way to plants with multiple inputs and multiple outputs (MIMO) as technology races ahead. Extremely advanced systems such as the Navy's Phalanx or Aegis Combat System are possible only in the more advanced MIMO framework. The result of this transformation is a dramatic increase in complexity and therefore greater difficulty in both analysis and design.

Regardless of composition, the control engineer recognizes the fact that, standing alone, plants will more often than not exhibit operating characteristics that are

undesirable for a specific application or a range of applications. This necessitates the concept of control and the field of control engineering. Essentially, a control system in some way alters the basic plant operating characteristics. This enables the designer to meet stability and performance specifications in the controlled system (control plus plant). Figure (1) illustrates the standard control system configuration.

As described above the plant G_p may be either relatively simple or quite complex. Figure (2) demonstrates this disparity. Note that in the SISO example only one transfer function G relates the plant input to the plant output. In the MIMO example, however, numerous transfer functions relate the two inputs to the two outputs and all transfer functions are inter-connected and simultaneously affect one another. Some simplification of this MIMO diagram may be possible but not to the level of simplicity seen in the SISO diagram.

Before further examining the difficulties the MIMO case presents, it is first necessary to introduce the concept of stability/robustness. Simply having the plant of interest and being able to observe its operation under certain conditions is by no means sufficient information for the process of analysis and design which will yield a controller that, when used in conjunction with the plant,

STANDARD CONTROL CONFIGURATION:

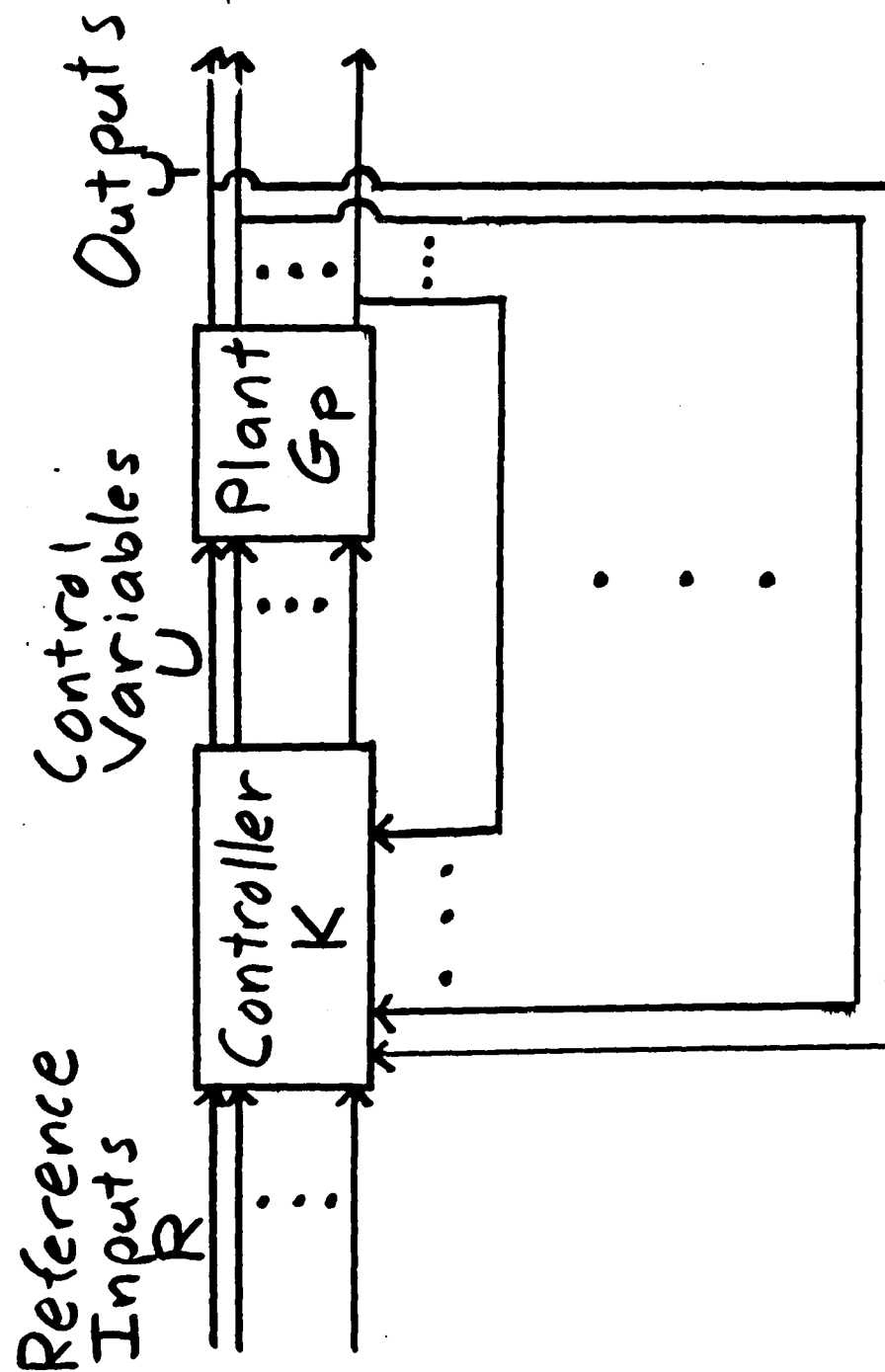
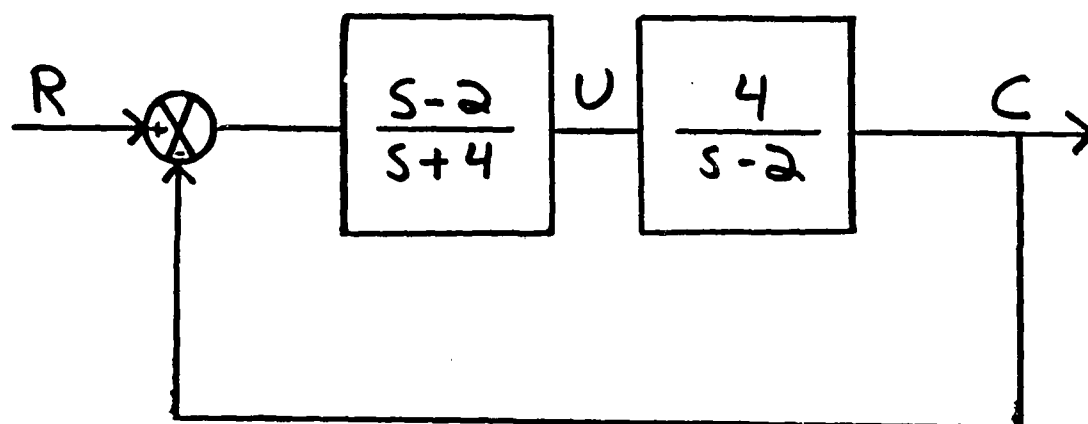


FIGURE (1)

SISO PLANT EXAMPLE

9



MIMO PLANT EXAMPLE

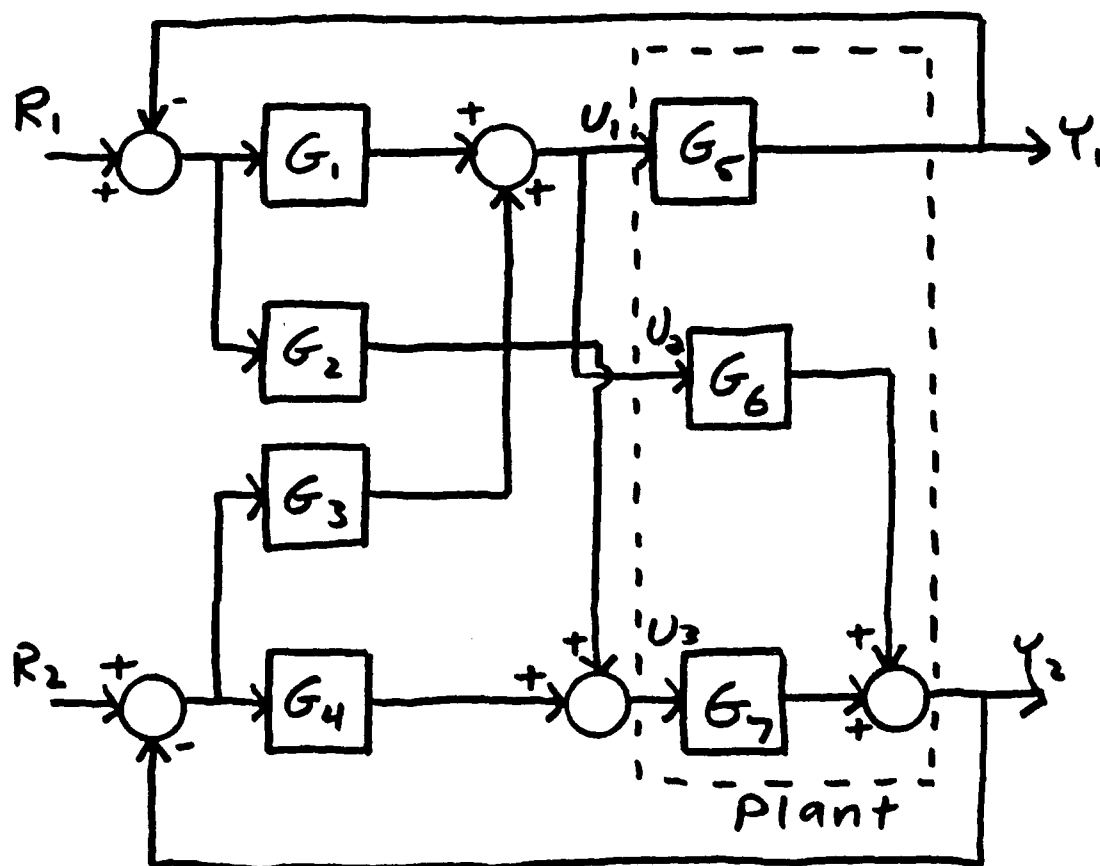


FIGURE (2)

will produce whatever operating specifications are desired. Rather it is necessary to possess or formulate some nominal plant model which, to some degree of accuracy, reproduces plant operation. Using that nominal plant model expressed in whatever format is relevant for the methods employed, the control engineer will then proceed to design the control system. The particular model format needed for the methods developed and used within this investigation will be presented in later sections.

What is of paramount importance here is the understanding that the nominal plant model will never be a true depiction of actual plant operation. For a variety of reasons and to varying degrees, an error between the actual and nominal model plants will always exist. The reasons for this disparity are three-fold:

- (1) Values of the parameters within the nominal plant model will exhibit variations due to changes in environmental conditions, maintenance-induced and calibration errors, and aging and wear.
- (2) Nonlinearities which cannot be expressed in the linear nominal plant models generally used by control engineers are present in real plants.
- (3) Certain dynamics which will not be present in the often rather crude, low order time invariant model approximations used in control system design are present in real plants.

This recognition of an inherent error between the nominal plant model and the actual plant raises a

fundamental question. If a control system is designed using the nominal plant model, how will the controlled system behave when operating with an actual plant? Because this paper is primarily concerned with stability, this question may be rephrased: If a controller is designed using the nominal plant model to preserve the stability of the system, will the system still be stable when it is an actual plant that is being controlled?

The concept of stability/robustness is the direct result of this question and is formally defined by:

Robustness [stability/robustness] refers to the delineation of finite regions of allowable variations in system parameters that preserve stability.[6, p.81]

Simply stated, stability/robustness is a measure of how much the parameters that make up our nominal plant model can vary before the controlled system becomes unstable. The task before the control engineer is to design a control system in which the critical property of stability holds in actual operation despite the presence of the errors described above.

Not surprisingly, considerable attention has been given to the issue of stability/robustness. For SISO systems, a significant body of theory exists that adequately addresses the problem by providing the designer with the means to

place built in guarantees or tolerances in the controller. Such effective SISO methodology includes Nyquist diagrams[4], Nichols charts, and Evans root-locus techniques[9]. For the MIMO case, it is quite another matter. Clearly, classical SISO methods cannot be applied to the MIMO stability/robustness problem:

...because these approaches [SISO methods] are incompatible with MIMO systems or with systems having nonlinear or time-varying plant ignorance, there is a need for alternative approaches...[9]

The reason is straightforward. In the SISO case the number of varying parameters is generally low and, if more than one, their relation usually is not complex. Using the methods identified above, the effects of variations can be analyzed and the controller designed to handle certain ranges of errors. In the MIMO case, many parameters will often vary and the relationships between them are of a highly complex nature beyond any hope of precise and useful mathematical expression. Primarily for these reasons, the above SISO stability/robustness methodology simply does not work with MIMO systems.

Having realized the ineffectual nature of SISO methods for the MIMO (or often referred to as multivariable) case, those in the control engineering field have firmly concluded that new methodology is needed. One approach which has been made is to look at each of the individual loops in the

multivariable system one at a time and then apply SISO methods while all other loops are held constant. The obvious deficiency in this approach lies in the complete failure to take into account simultaneous changes in these loops[6, p.76]. Also, an attempt has been made by MacFarlane, Rosenbrock, et al, to extend SISO Nyquist design methodology to handle multivariable systems. These methods too have run into the difficulty of failing to take into account possibilities inherent in the complex MIMO system[4]. Finally, singular values (denoted by σ) have been introduced which, roughly stated, are used to represent all the possibilities in the multivariable case. Using singular values, stability/robustness inequality tests have been developed which depend on both knowledge of the nominal plant model and knowledge of the error between that model and the actual plant[3].

It is at this point that this Trident Scholar Project enters the picture. The singular value inequality test method for stability/robustness presents the greatest opportunity for resolving the MIMO dilemma, but it too has a dramatic flaw. The error it employs in the inequality tests is based on the difference between the G_p of the nominal plant and the G_p of the actual plant. As will be shown later, this type of error is quite difficult to define.

This project, however, has developed alternative

singular value inequality tests based on error information which is much more readily determined by the engineer and therefore of greater practical value. The limitation of the project methodology lies in the fact that it applies mainly to systems operating at low to mid frequencies, although even for higher frequency systems, this is often the chief area of concern due to performance considerations.

An additional and quite different method of assessing the stability/robustness of a MIMO system was also developed in the project. Essentially, it involves the identification of the parameters which are most important to the stability of the system followed by the determination of regions of stability for these identified parameters. The method side-steps the problem of finding and using mathematical relationships governing the parameters of the system. Its effectiveness wanes as the number of parameters with significant effects on stability increases but is still shown to be a valuable design tool for many applications.

The remainder of the project is devoted to the design of a controller (or compensator) based upon stability/robustness considerations. The Model Based Compensator/Linear Quadratic Gaussian/Loop Transfer Method (MBC/LQG/LTR) [1],[3],[10] serves as the foundation for compensator design but only when augmented by the project findings described above does it become a useful tool which

the engineer can use to design for good stability/robustness in the controlled system. The latter portion of the paper, including a description of a standard design process and an example, encompasses all earlier portions of the paper and serves both as a summary and as a valuable tool which the practical control engineer can utilize when confronted with the MIMO control system design problem from start to finish.

OBJECTIVES

The objectives to be met in this Trident Scholar Research Project were:

- (1) Develop the means to assess the stability/robustness of a nominal multiple input/multiple output (MIMO) plant.
- (2) Complete an analysis of the MBC/LQG/LTR MIMO compensator design methodology and write the necessary computer programs to implement it on the engineering software package PC-MATLAB.
- (3) Develop guidelines, to be used within the framework of the MBC/LQG/LTR methodology, for the design of a compensator that will provide good stability/robustness for the controlled MIMO system and write the necessary PC-MATLAB programs to realize the theoretical findings.
- (4) Formulate a standard design process which enables the practical control engineer to use the results from objectives (1-3) and meet requirements for a stable, robust system.

RESEARCH AND IMPLEMENTATION

PART I
Methods to Assess Stability/Robustness of
a MIMO Nominal Plant Model

Description

Any nominal plant model will always differ by some error from the actual plant it represents. Stability/robustness is a measure of how sensitive a nominal model of a system is in terms of stability to this error. If a plant is nominally stable, but will remain stable only when faced with a slight error, it can be said to possess poor stability/robustness. Conversely, if a plant is nominally stable, and will remain stable even when subjected to a significant error, it can be said to possess good stability/robustness. Obviously, it is the latter and not the former condition that the control engineer desires.

The first step in any control system design process must be to assess the operating characteristics of the plant alone. This includes an analysis of both its open loop and closed loop properties. Figure (3) illustrates the standard open loop configuration and Figure (4) illustrates the standard closed loop configuration for the plant alone (no controller). The closed loop properties are of primary concern as it is the closed configuration which generally

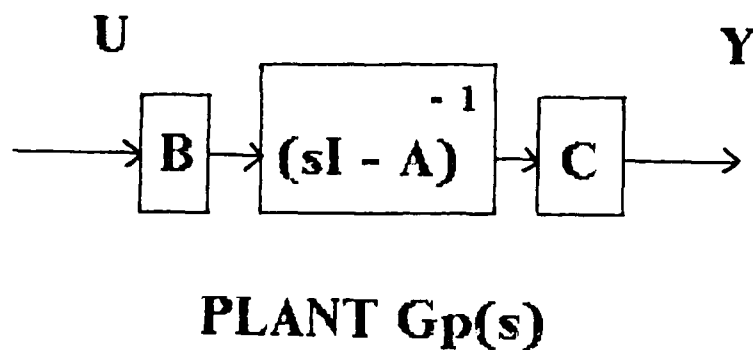


FIGURE (3)

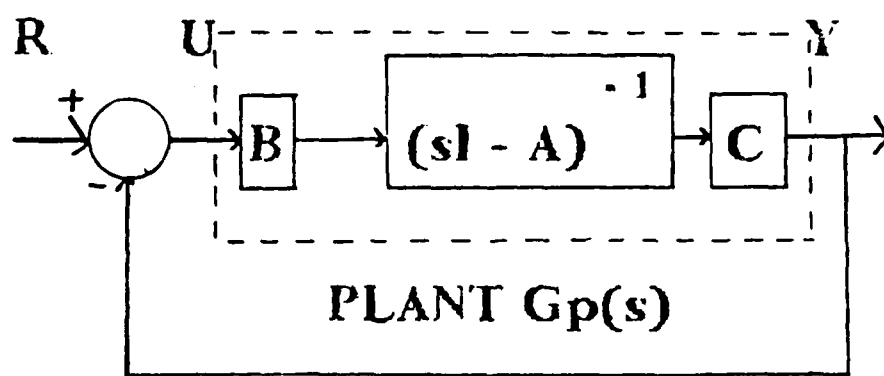


FIGURE (4)

affords the more desirable operating characteristics including, but not limited to, steady state error, command following, and disturbance rejection. Therefore, unless a statement is made to the contrary, it is the closed loop properties and more specifically closed loop stability that is discussed.

Reasons for assessing the stability/robustness of the plant alone (no controller) include (1) if the plant alone already meets design requirements, then the design of a controller (referred to later as a compensator) is not necessary and (2) this assessment will provide the designer with some feel as to what degree the plant operating characteristics must be modified by the control system.

This portion of the paper describes ways in which the stability/robustness of a nominal MIMO plant model can be determined. Two methods unique to this project, one in terms of format and the other in terms of the concept, are presented:

- (1) Use of singular value inequality tests method
- (2) Use of deviation of parameters method

It is important to note that while this methodology is being developed in terms of the plant alone in this portion of the paper, the findings, particularly the singular value inequality tests, will be equally applicable later when the

development of methodology for the design of a compensator for good stability/robustness is discussed.

Development and Discussion

A. Theoretical Background

1. Figures (3) and (4) show the format which will be used to describe the MIMO nominal plant model in this project.

The complete open loop mathematical formulation is given by:

$$\dot{x} = Ax + Bu \quad (1.1)$$

$$y = Cx \quad (1.2)$$

$$G_p(s) = C (sI - A)^{-1}B \quad (1.3)$$

While the complete closed loop formulation is given by:

$$\dot{x} = (A - BC)x + Br \quad (1.4)$$

$$y = Cx \quad (1.5)$$

$$G_{cl}(s) = \frac{G_p(s)}{I + G_p(s)} \quad (1.6)$$

$$\text{and} \quad I + G_p(s)$$

$$G_{cl}(s) = C (sI - A + BC)^{-1}B \quad (1.7)$$

$$A_{cl} = A - BC \quad (1.8)$$

Of importance for closed loop stability is the A_{cl} matrix.

By evaluating the characteristic equation:

$$|sI - A_{cl}| = 0 \quad (1.9)$$

one is able to solve for the values of s that satisfy it and thus obtain the eigenvalues of the closed loop nominal

plant. Figure (5) demonstrates some of the possible solutions in the s-plane. The key is that if any of these solutions, or eigenvalues, lie in the right half plane of this diagram then the plant will be unstable. In Figure (5), the system would be unstable because an eigenvalue is in the right half plane.

Computer-Aided Design (CAD) software such as PC-MATLAB have built-in functions which can calculate the eigenvalues of any matrix, and thus of A_{C1} , with ease. If the A, B, and C matrices are known at any time, then the eigenvalues of A_{C1} can be calculated and the stability of the system determined. This will be of use later in this part of the paper.

2. It is essential to gain an understanding of what is meant by singular values (denoted by σ) before utilizing them in the assessment of MIMO stability/robustness. Singular values are used to overcome a fundamental difficulty in multivariable analysis; namely, because of the complexity inherent in these systems there are often numerous possibilities as to how the system will operate for any given condition. For example, in the SISO case it can be determined what the gain will be between the plant input and output at a certain frequency. In the MIMO case, however, there may be an entire range of possible gains that could exist at that frequency.

S PLANE DIAGRAM: X INDICATES AN EIGENVALUE

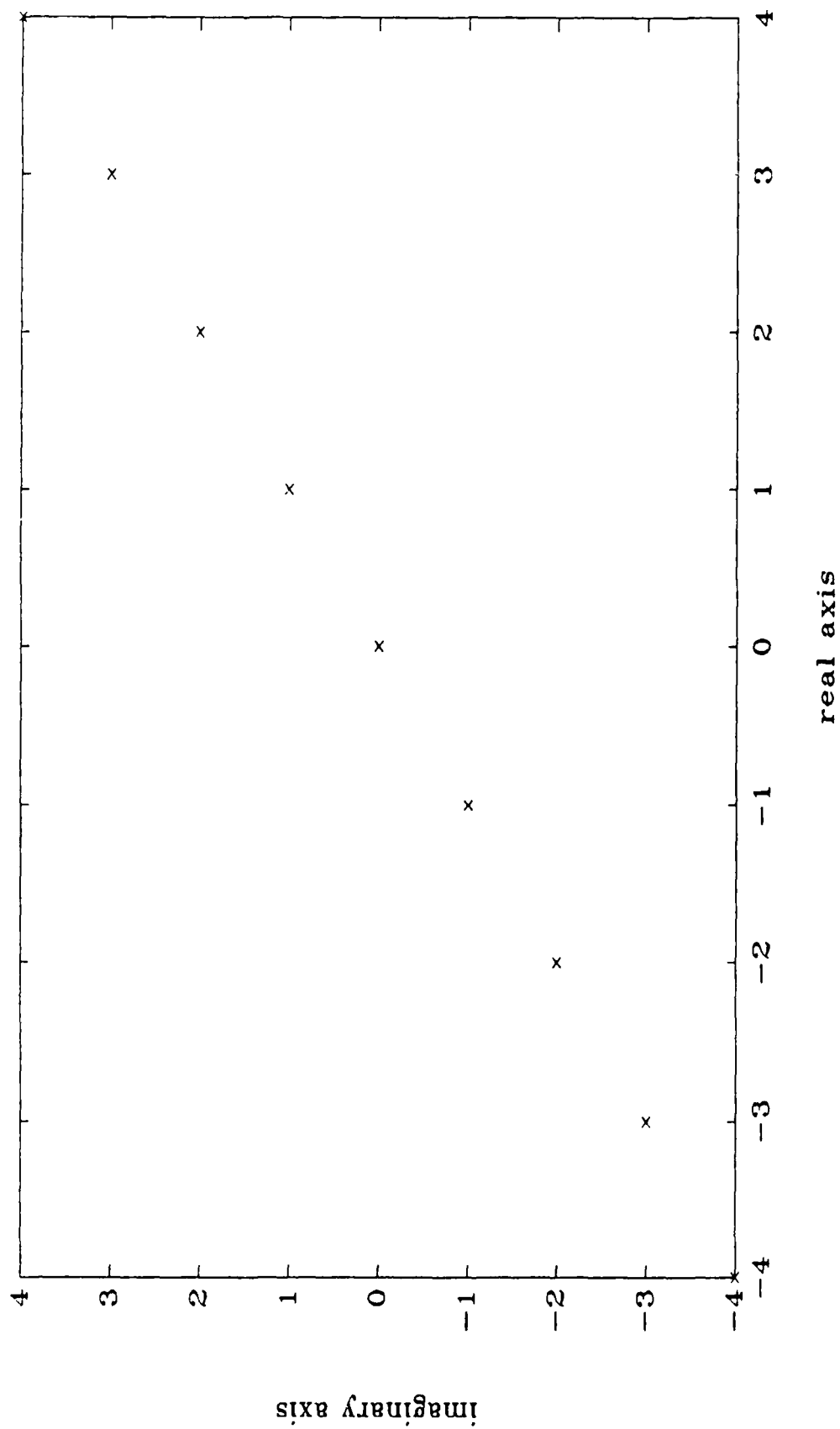


FIGURE (5)

Singular values are used to describe this range. Most importantly, the maximum and minimum singular values ($\sigma_{\max}$ and $\sigma_{\min}$) can be thought of as the upper and lower bounds, respectively, on this range. Figure (6) is a singular value plot of an open loop nominal plant. The y axis value is the gain in db between the plant input u and output y , which corresponds to the magnitude of $G_p(s)$. The upper line represents $\sigma_{\max}$ and the lower line $\sigma_{\min}$, and the gap between them shows the range of possible operating conditions. These maximum and minimum singular values will be heavily exploited in the singular value inequality tests that are developed. See Appendix A.1 for the definition of and fundamental relations governing singular values.

3. Mention will be made in the portions of the paper to follow of a theoretical worst error for stability. Solution of equation (1.9) gives the eigenvalues of the A_{C1} matrix which were shown to be direct indicators of stability according to which half plane of Figure (5) in which they lie. If the real part of any eigenvalue (x axis value) is positive, then the system is unstable. If the real part of one eigenvalue is zero and all the rest have negative real parts, then the system is on the very border of stability. In addition, it is stated in Appendix A.1 that the closest eigenvalue to zero is related to $\sigma_{\min}$ in the sense that if one goes to zero, the other must as well.

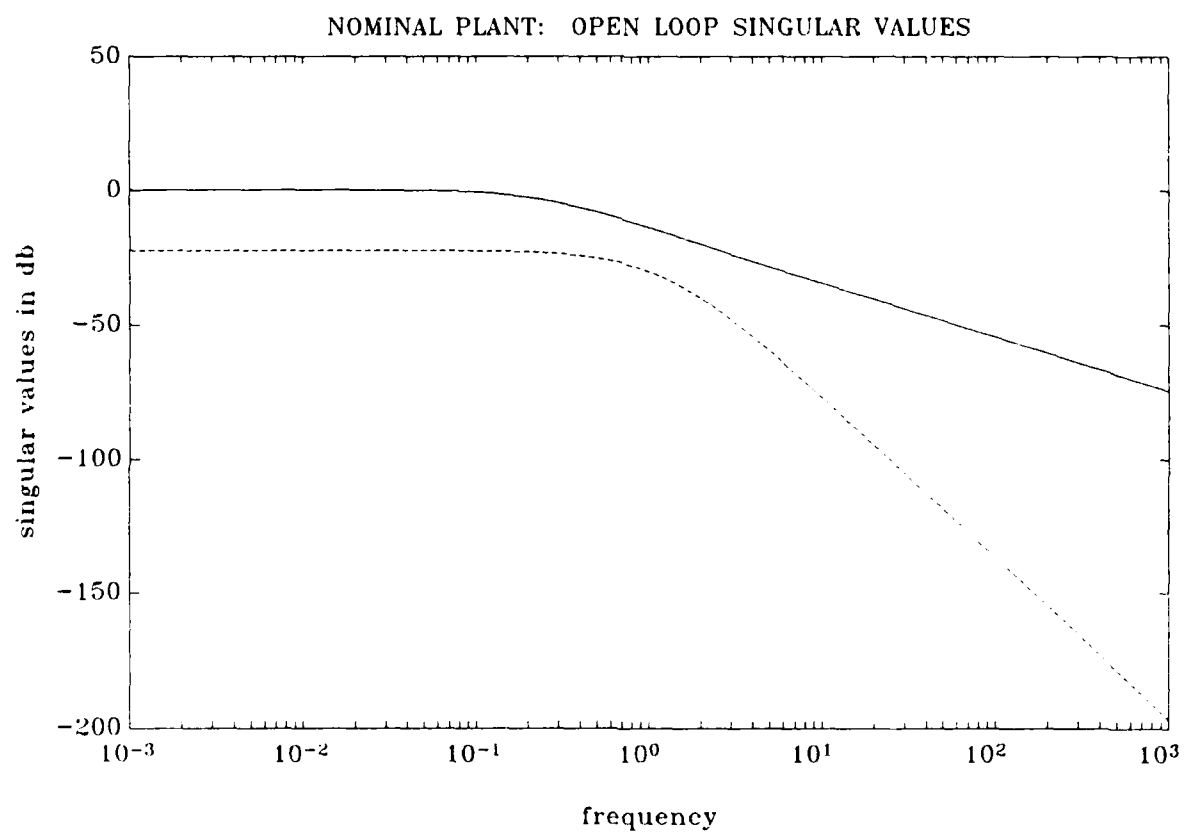


FIGURE (6)

This theoretical worst error for stability (notated as E_w) is defined as the minimum error in the least likely direction added to $A_{C1}(\text{nom})$ which will just make $\sigma_{\min}$ go to zero (and the closest eigenvalue as well) and place $A_{C1}(\text{act})$ on the very border of stability. Note that worst does not imply largest.

$$A_{C1}(\text{act}) = A_{C1}(\text{nom}) + E_{C1} \quad (1.10)$$

$$E_w = E_{C1} \text{ in worst case for stability} \quad (1.11)$$

See Appendix A.2 for the explicit calculation of E_w .

A couple of additional notes on E_w are in order. First, the E_w described here exists only at low frequency ($\omega \rightarrow 0$). As frequency increases, this E_w will no longer accurately describe the difference between $A_{C1}(\text{act})$ and $A_{C1}(\text{nom})$ for worst case error. Second, the true value of E_w lies in its application to the methodology to follow. In the deviation of parameters method, it is used as a basis for choosing which parameters have the greatest impact on the stability of the nominal plant model. In the singular value inequality method, it can be used as a check on the tests established. At low frequency, $\sigma(E_w)$ should exactly coincide with $\sigma_{\min}(SI - A_{C1}(\text{nom}))$.

B. Singular Value Inequality Test Method

1. The concept of using singular value inequality tests to gauge stability/robustness for MIMO systems is not a new one. Michael Athens, Gunter Stein, John Doyle, et al, have established such tests already that are entirely valid[3]. However, these tests include a large obstacle that even the developers have been unable to overcome. Essentially, these tests are based on both singular value plots of the nominal plant model and on an error between the nominal and actual $G_p(s)$, with $G_p(s)$ as defined in equation (1.3). The crux of the problem lies in the fact that it is very difficult, if not impossible, to physically define this error. $G_p(s)$ has little physical meaning.

The result is that however valid the tests may be, they remain generally ineffective for solving the MIMO stability/robustness problem. The answer given when confronted with this situation is basically that only extensive experience with the MIMO plant in question will yield any idea as to what this error might be.

2. Project methodology adopts the concept of the singular value inequality test but differs in the composition of the inequalities and definition of the error between actual and nominal systems. Instead of focusing on $G_p(s)$, A_{c1} as defined in equation (1.8) was adopted as the focus of interest. The necessary error for the inequalities used

here becomes E_{cl} as defined in equation (1.10).

Because E_{cl} is entirely dependent on differences between the nominal and actual A, B , and C matrices which are mathematical descriptions representing physical entities, it becomes a much more physically meaningful error than one involving $G_p(s)$. For instance, the nominal values of A, B , and C are always known at any time, and the control engineer has some hope of assessing what they are in actuality. Such is not the case when using an error based on $G_p(s)$.

The singular value inequality tests established were:

(1) IF : $\sigma_{\min}(SI - A_{cl}(\text{nom})) > \sigma_{\max}(E_{cl})$ (1.12)
 THEN: The actual system is guaranteed stable at low w

(2) IF : $\sigma_{\max}(SI - A_{cl}(\text{nom})) < \sigma_{\min}(E_{cl})$ (1.13)
 THEN: The actual system is guaranteed unstable at low w

(3) IF : Neither condition holds (1.14)
 THEN: The actual system may be either stable or unstable at low w

See Appendix A.3 for a proof of singular value inequality test (1), the most important and most used of the three, particularly for control system design. Although substantiated by considerable experimentation, a mathematical proof for test (2) has yet to be found. Test (3) follows from the first two.

Figure (7) illustrates graphically the possibilities

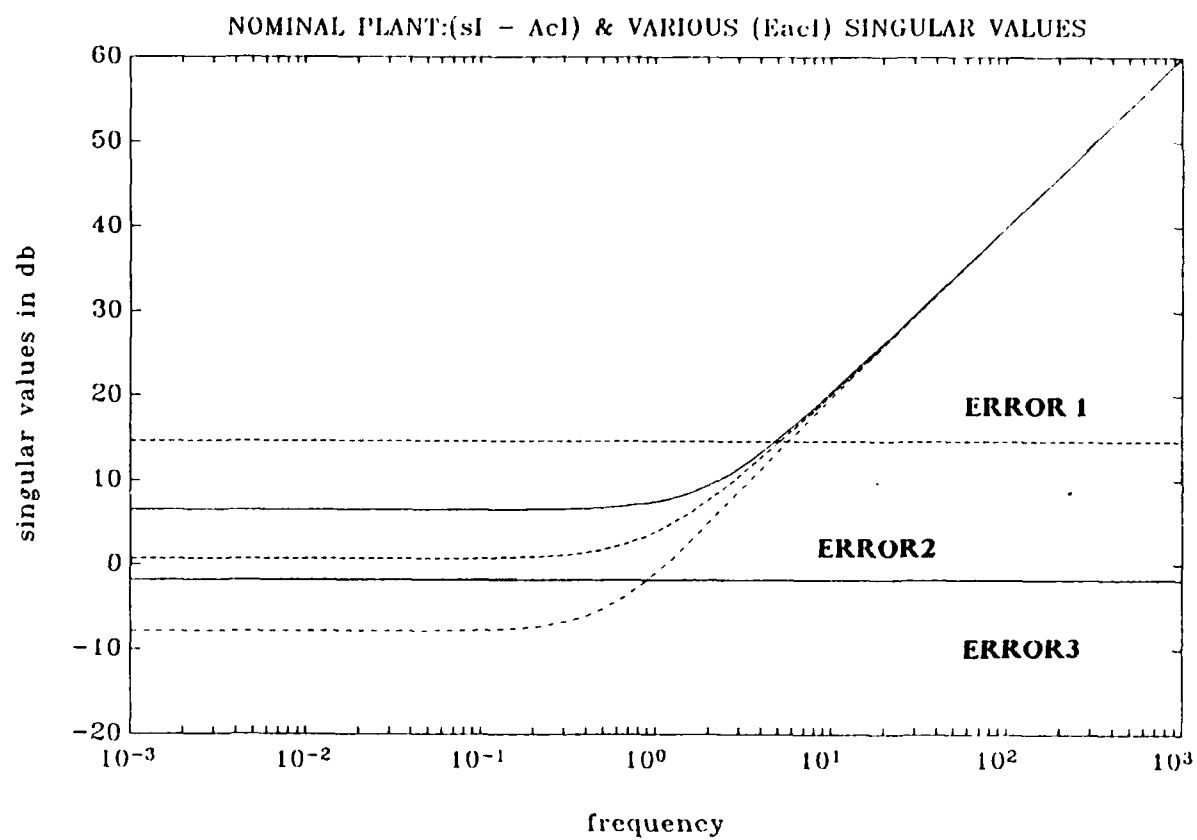


FIGURE (7)

inherent in these three tests. "Error1" represents $\sigma_{\min}(E_{ac11})$, "Error2" represents $\sigma_{\max}(E_{ac12})$, and "Error3" represents $\sigma_{\max}(E_{ac13})$. The first parallel and then joining lines represent $\sigma(sI - A_{c1}(\text{nom}))$ with the top being the maximum and the bottom the minimum. Applying the inequalities, one discovers that E_{ac11} will guarantee instability, E_{ac12} may result in stability or instability, and E_{ac13} will guarantee stability in the actual system.

3. Here then is a means which can be used to assess the stability/robustness of the MIMO nominal plant model. First $\sigma(sI - A_{c1})$ is plotted. This plot was chosen for use in the test because (1) A_{c1} is the plant term of direct interest for closed loop stability and (2) $\sigma(sI - A_{c1})$ is an expression of system behavior that does not demonstrate significant frequency effects until the mid frequencies as evidenced by Figure (7). The engineer then uses information he obtains concerning the difference between the nominal and actual plants at low frequency to calculate E_{ac1} . Finally, $\sigma(E_{ac1})$ is plotted and the conclusion regarding stability/robustness made using the three singular value inequality tests.

Obviously a limitation of this methodology lies in the imposition of a frequency range on the validity of the tests. Yet, while a constant E_{ac1} might be calculated at low frequencies ($\omega \rightarrow 0$), its relevance extends at least to

mid-range frequencies. The reason is that, as was mentioned above, frequency effects on $\sigma(sI - A_{C1})$ do not really come into play until the mid-range frequencies (the flat portions of the plots) and thus the relevance of the tests is maintained until that point.

It will also be shown in Part III of the paper that extra measures of stability/robustness are provided in the test's weakest region of applicability, the high frequencies, in the compensator design process. The end result is that the engineer can evaluate E_{ac1} directly from $A_{C1}(\text{nom})$ and $A_{C1}(\text{act})$ at low frequency and yet have valid stability/robustness results through higher frequencies, thereby encompassing the chief area in which the designer will desire the system to operate anyway for performance considerations. While the project method is not the overall, general solution to the problem, it does present the engineer with somewhat limited but usable techniques instead of the unlimited but unusable techniques previously in existence.

C. Deviation of Parameters Method

1. The deviation of a plant parameter refers to a change in value for one of the plant parameters between the nominal plant model and the actual plant. Reasons for such

parameter variations are offered in the Introduction and the fact that deviations will always be present is without question. Within a stability/robustness framework, however, the crucial issue centers not on the existence of deviations but on their effect on the stability of the plant.

Intuitively, one may suppose that certain deviations would have little effect on stability while others could have dramatic impacts.

The problems the MIMO case presents for analyzing the effects of parameter deviations on stability include (1) the fact that a MIMO plant is quite simply likely to be composed of many more parameters than in the SISO case and (2) as the numbers of parameters increase, the likelihood of any mathematical representation of the effects the deviations have on one another becomes extremely remote, if not quite hopeless. In fact, this project did not uncover any attempts to address the MIMO stability/robustness problem from this direction.

2. Project methodology requires, first, that a worst case error for stability be calculated. See Appendix A.2 for the calculation of this error E_w and section A.3 above for a description, which includes equations (1.10) and (1.11) as mathematical definitions. E_w reflects the change $A_{C1}(\text{nom})$ is most sensitive to with regard to stability, and will be used as the basis for determining the significance of

individual parameters on the stability/robustness of the plant. The only change in E_w here is that it is created to be physically meaningful in relation to $A_{C1}(\text{nom})$ for whatever parameters, including all, that are of interest initially. For example, if a certain element of the $A_{C1}(\text{nom})$ matrix does not change at all when all of the parameters of interest are deviated, then the corresponding element in the E_w matrix will be zero. Computer routines then search for a constant with which to modify the E_w matrix, which is now missing elements, so that it remains a worst case error.

The next step is to test each element of the matrix E_w for effects on the stability of the plant. This is accomplished by use of computer routines which deviate each element of E_w and see if the resulting change in the eigenvalue closest to zero falls within a specified tolerance or not. If the change does fall within the tolerance, that element is set to zero. All elements are tested in this manner and a reduced E_w is the result.

Following this the effect of each parameter on stability is evaluated by computer routines which (1) deviate the parameter, (2) calculate a new worst E_w using that deviation, (3) check the effect of the parameter on E_w by comparing the new to old E_w and (4) check the effect of the parameter on stability by seeing if the change in the

eigenvalues of the new and old E_w fall within a specified tolerance. The result is the elimination of parameters for stability/robustness concern which demonstrates negligible effects in (3) and (4), as well as a prioritization of those that remain in terms of their stability significance. The prioritization of the parameters is accomplished by gradually making the eigenvalue comparison tolerance in (4) more coarse until, one by one, the parameters drop out. The parameter remaining at the end is most important to stability, the last to be removed is the second most important and so on.

The final step is to construct region of stability plots for the parameters that are found by the above techniques to have significant effects on stability. These plots graphically show allowable percent deviations in each of the parameters plotted. The plot is constructed in the appropriate number of dimensions for the number of these parameters of concern for stability. Figure (8) is a region of stability plot for a MIMO plant that began with 11 nominal parameters and was reduced to the three parameters shown. The allowable percent deviations in each parameter are clearly delineated.

3. The methods developed here present a reliable, effective means of analyzing the stability/robustness of many MIMO plants. Obvious problems result when the number

STABILITY REGION PLOT: 2 RESISTORS, 1 CAPACITOR

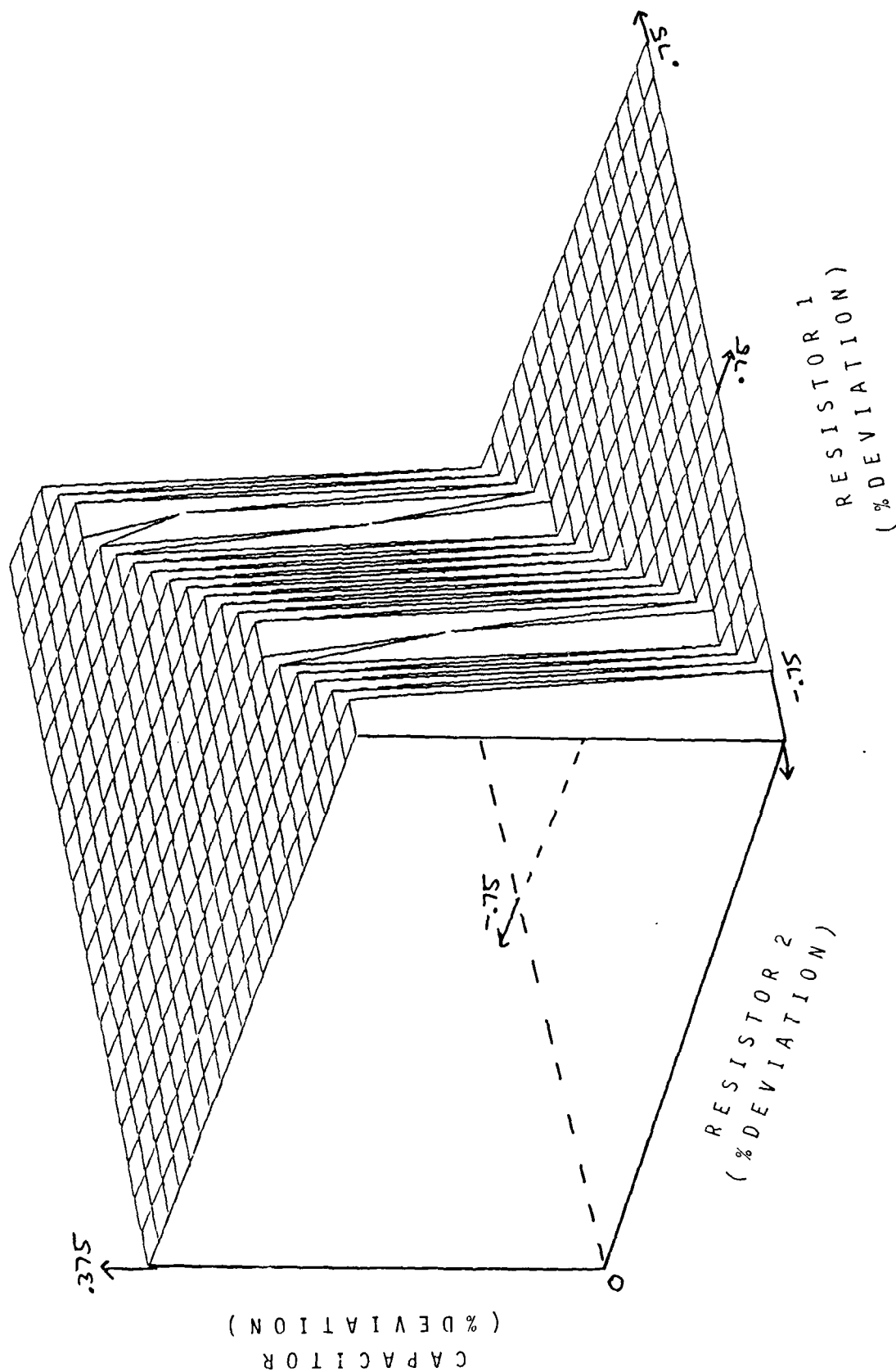


FIGURE (8)

of significant parameters cannot be reduced sufficiently for a single region of stability plot (more than three parameters to plot) and the reduction process itself becomes more course as the total number of parameters is increased. Nonetheless, considering the lack of any other progress in this approach to MIMO stability/robustness, the benefits of the method developed are very apparent.

PART II
A Review of the MBC/LQG/LTR MIMO
Design Methodology

Description

The design of a control system will be necessary if the control engineer determines that the operating characteristics of the nominal MIMO plant model are not satisfactory. Usually the engineer will be presented with certain specifications that are to be met. A conclusion that the plant must be modified is reached when the plant operating characteristics fail to meet these specifications. The nature of such specifications vary widely from any number of performance requirements, including but not limited to good command following and zero steady state error, to stability requirements. The specification that the system must remain stable under most conditions is almost always present and certainly the most important. If the system is not stable, performance is a moot point.

Having made the determination that the plant must be modified, attention then turns to the means with which to accomplish it. The control system may take many forms but the standard configuration, and the one used here, was seen

in figure (1). Essentially, it involves the addition of a compensator $K(s)$ in the feedforward path which includes the plant $G_p(s)$ and the use of a feedback path from the plant output to a reference input. The design of $K(s)$ then emerges as the basic problem confronting the control engineer.

Just as was the case with the assessment of MIMO stability/robustness, the design of a MIMO compensator presents significant problems in the context of conventional SISO methodology. The inherent complexity and possibilities of operation in the MIMO case once again render such methods virtually useless. It is only the recent emergence of new MIMO compensator design theory that has made the problem tractable. The Model Based Compensator/Linear Quadratic Regulator/Loop Transfer Recovery (MBC/LQG/LTR) [1],[3],[10] method developed by Athens, Stein, et al, provides the means to obtain a compensator $K(s)$ to meet stipulated specifications in the MIMO case. All material methodology presented in this part of the paper is attributed solely to them.

The purpose here is to present a brief outline of this method and its use. The intent is not to offer a complete explanation or justification, but rather to provide an understanding of its implementation for later portions of this paper which will utilize the method. Basically, this

methodology will serve as the framework upon which the tests developed in Part I will be used to design a compensator that will guarantee stability/robustness for the control system (compensator + plant). For those desiring a more complete understanding of the MBC/LQG/LTR method, attention should be directed to the References where some excellent sources are listed.

Finally, while a theory has been developed and is accessible, a significant obstacle remains for anyone desiring to implement it. The great majority of the theory depends entirely on the availability and use of Computer-Aided Design (CAD) software. The package PC-MATLAB was chosen and used for this project. Such CAD software might be thought of as the algebra, however, while the equations remain to be written. A significant portion of this Trident Project was spent developing the computer programs which would use the mathematical capabilities of the software to implement the MBC/LQG/LTR methodology. Descriptions and instructions for these programs are listed in Appendix B. The programs themselves will be kept in the U.S. Naval Academy Weapons and Systems Engineering Department where they are available for general use.

Development and Discussion

A. The Design Plant Model (DPM)

The first step in the MBC/LQG/LTR methodology is to specify a DPM. The DPM is similar to the nominal MIMO plant model but differs in two important aspects:

- (1) All variables used in the nominal plant model may not be of the same type, meaning that the units used to describe them may not be compatible. For example, an output in volts cannot be directly compared to an input in degrees. The units are different. Therefore, a scaling of the variables is reflected in the DPM.
- (2) The designer may want to include certain augmentation dynamics (such as integrators) to meet performance requirements (such as zero steady state error). These dynamics are included in the DPM.

The mathematical formulation of the DPM is identical to that seen earlier in equations (1.1) to (1.8) for the nominal plant model. The only difference is that the matrices A , B , and C will include those changes made for reasons (1) and (2) above. If no such changes are made, the DPM will be identical to the nominal plant model. Therefore, if a subsequent reference is made to the nominal plant model and

not the DPM, it means that neither scaling nor augmentation dynamics changes were made.

B. The Target Feedback Loop (TFL)

The TFL represents the feedback loop the designer desires when his compensator (described later as the MBC) is added to the plant, thus the name "Target". To some degree of accuracy, the way this loop performs will be the way the entire control system will perform. The meeting of all performance, and most importantly for this paper, stability requirements is done in this stage of the compensator design process.

Figure (9) is a diagram of the TFL. It is identical in structure and similar in content to that seen for the DPM. The open loop mathematical formulation is given by:

$$\dot{x} = Ax + Hu \quad (2.1)$$

$$y = Cx \quad (2.2)$$

$$G_h(s) = C (sI - A)^{-1}H \quad (2.3)$$

While the closed loop formulation is given by:

$$\dot{x} = (A - HC)x + Hr \quad (2.4)$$

$$y = Cx \quad (2.5)$$

$$G_{clh}(s) = \frac{G_h(s)}{I + G_h(s)} \quad (2.6)$$

$$\text{and} \quad I + G_h(s)$$

$$G_{clh}(s) = C (sI - A + HC)^{-1}H \quad (2.7)$$

$$A_{clh} = A - HC \quad (2.8)$$

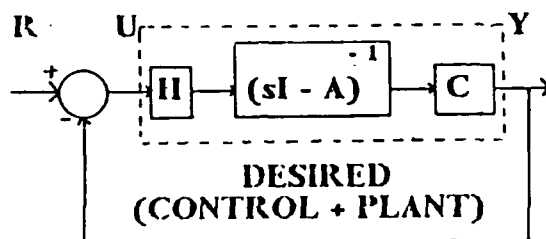


FIGURE (9)

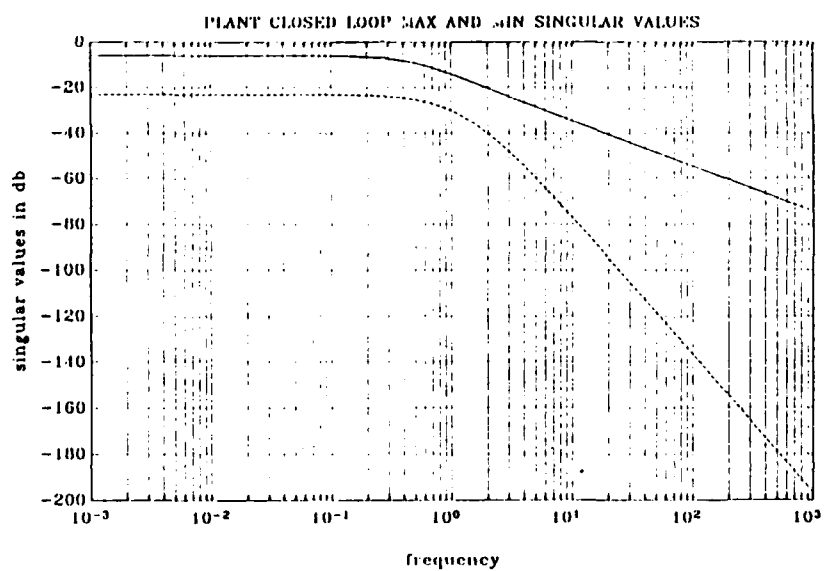


FIGURE (10)

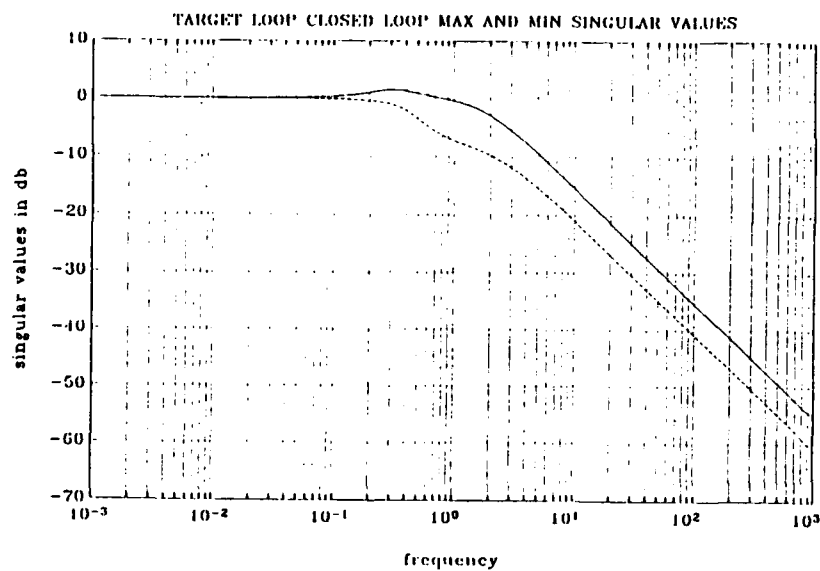
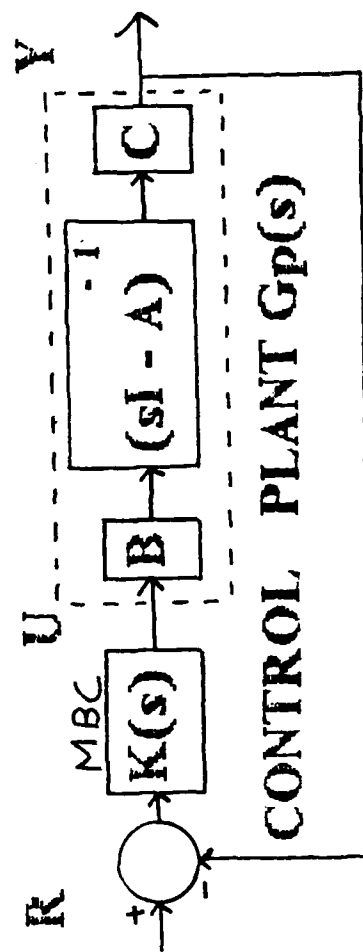
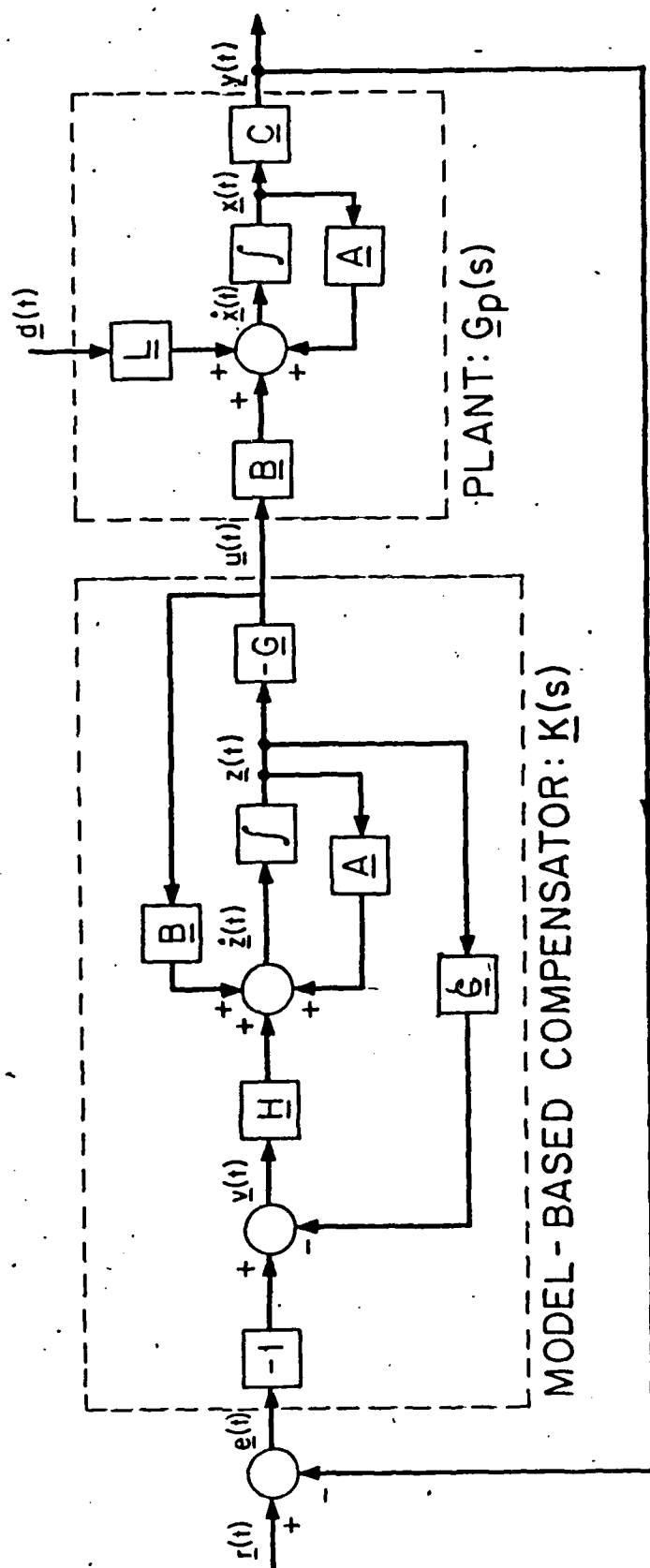


FIGURE (11)

Note that the only difference between this set of equations and those used for the nominal plant model (and the DPM) is the matrix H in place of the matrix B . This matrix H is the design parameter available to the engineer with which he can shape the TFL operating characteristics to meet specifications for the controlled system.

Figure (10) is a singular value closed loop plot of G_{cl} , the nominal plant. By a judicious selection of the matrix H , the designer wishes to match all singular values (essentially reduce to one line) and achieve a value of 0 db at low frequencies for good command following. Figure (11) is a singular value closed loop plot G_{clh} of a TFL that achieves these objectives. Methods currently exist, for both $G_h(s)$ and $G_{clh}(s)$, to match TFL singular values at low frequencies, high frequencies, or both, and to move the entire plot up or down in value.

For an explanation of how this is accomplished see the literature in the References. It essentially involves the use of a fictitious continuous time Kalman Filter to find H . The literature should also be consulted concerning the inherent, built in properties of the TFL. The computer programs in Appendix B provide the designer with a choice of where to match the singular values and also with a choice for the value of the scalar that will move the plot up or down in value. Based on these choices, the design parameter



matrix H is calculated and a TFL realized. The range of possible values of this scalar will be discussed in Part III with regards to stability/robustness.

C. The Model Based Compensator (MBC) and TFL Recovery

Now that the TFL has been designed to perform like the controlled system should perform, it is necessary to produce the compensator (referred to in this method as a MBC) that when added to the plant will recover this TFL. In other words, by adding a MBC to the plant, the system should now behave just as the TFL does, to some degree of accuracy. Figure (12) is a diagram of the MBC itself and Figure (13) is a diagram of the entire control system with a compensator.

While the structure of the MBC as seen in Figure (12) is new, all the terms in it are not save one. The matrices A , B , and C all come from the nominal plant model (or DPM) and the matrix H from the TFL. What is new is the matrix G . This is the design parameter that enables the MBC to essentially cancel the plant and insert in place of it the TFL operating characteristics. Figure (14) includes plots of the TFL closed loop singular values and the control system with the MBC added closed loop singular values. Ideally, they should be the same, but one can immediately

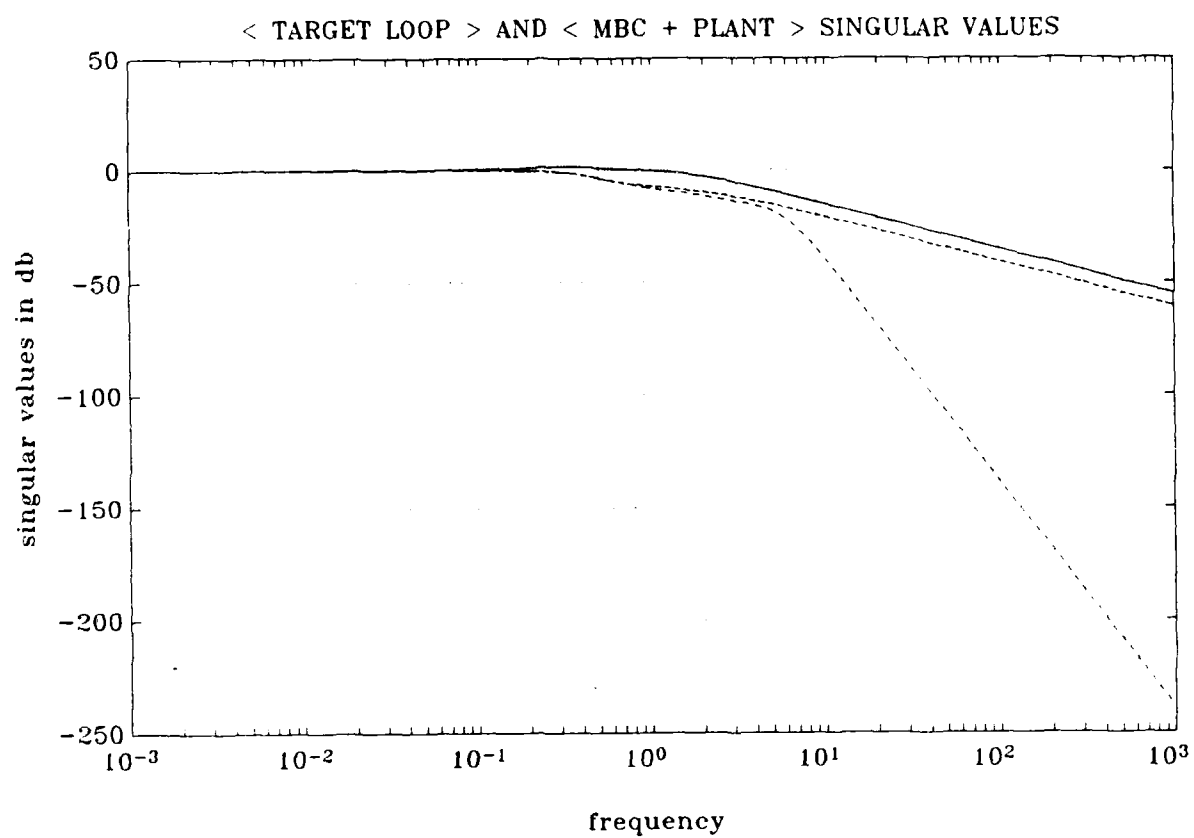


FIGURE (14)

see that at high frequencies they begin to differ.

Again, for those interested in exploring how the TFL is recovered by the MBC, attention is directed to the literature in the References where complete explanations are provided. Basically, G is calculated via a solution to the cheap-control Linear Quadratic Regulator (LQR) problem. The computer programs in Appendix B provide the designer with a choice for a scalar to be used in calculating G . This scalar determines the accuracy with which the control system with the MBC will recover the TFL operating characteristics and also what magnitudes of controls will be used. Not surprisingly, the better the recovery, the larger the control magnitudes.

D. Summary of MBC/LQG/LTR Method

- (1) Define the DPM. If neither scaling or augmentation dynamics changes are desired, then the DPM will be identical to the MIMO nominal plant model.
- (2) Calculate the TFL design parameter matrix H to make the TFL operating characteristics meet all required specifications. This is, ideally, what the plant augmented by the MBC will look like.

- (3) Calculate the MBC design parameter matrix G to recover the TFL operating characteristics. Compare MBC plus plant loop behavior to that designed in the TFL. If agreement is close, and therefore specifications are met, then the MBC design process is complete.

PART III
Compensator Design Based on Stability/Robustness
Considerations

Description

Part I provided the means with which to assess the stability/robustness of a MIMO nominal plant model. Quite often the control engineer will not find the results acceptable. The plant operating characteristics must then be modified by the addition of control to the plant to correct whatever deficiencies are found.

Control will be added to a plant not only to overcome poor stability/robustness but also to meet performance requirements that the plant alone does not attain. Even if the engineer is adding a compensator only to improve performance, however, he still must be concerned about stability/robustness. The reason is that when changing the plant to improve performance, the stability/robustness properties could deteriorate. Therefore, whenever a compensator is added to the plant, the designer must possess some allowable bounds on his compensator design parameters that will insure good stability/robustness.

Part II summarized the MBC/LQG/LTR design methodology.

A designer utilizing this methodology is able to modify the plant by use of the MBC to achieve whatever operating characteristics were defined in the TFL. In terms of stability/robustness, the problem then becomes how to shape the TFL so that an actual stable system is insured upon recovery of the TFL. The answer is to apply a stability/robustness singular value inequality test defined in Part I and then find which shapes of the TFL indicate that the actual system will remain stable. The result will bound some design parameter. These bounds are shown below to be on a scalar used in calculating the H matrix of the TFL, within which the designer can then seek to meet other requirements for system operation such as performance specifications.

Here too, computer programs were written for the software package PC-MATLAB. These programs are described in Appendix B. The specific uses for those most important to this part of the paper will be given below but Appendix B should be consulted to identify the applicable programs and to receive instructions on their use.

Development and Discussion

A. The MBC and TFL Shaping Using Singular Value Inequalities

1. The singular value tests given in equations (1.12) to (1.14) that were used to evaluate the stability/robustness of a MIMO nominal plant model have direct applicability in the TFL. As described above, the goal is to find the shapes of the TFL that will insure good stability/robustness. Equation (1.12) provided the test for guaranteed stability and therefore it is the test that will be used. However, because it is based on the nominal plant model and not the TFL, it must be modified slightly. The difference is in the definition of the closed loop A matrix A_{cl} . In the TFL interest is in A_{clh} from equation (2.8) and not A_{cl} , so the test equation becomes:

$$\begin{array}{ll} \text{IF} & : \sigma_{\min}(sI - A_{clh}) > \sigma_{\max}(E_{cl}) \\ \text{THEN} & : \text{The actual system is guaranteed stable at low } w \end{array} \quad (3.1)$$

Recall that E_{cl} is the difference between the actual and nominal closed loop A matrices that will occur as defined in equation (1.10). Pounds on the design parameter of the TFL can then be found such that the plots of $\sigma_{\min}(sI - A_{clh})$ and $\sigma_{\max}(E_{cl})$ never cross, thereby insuring a system which can never go unstable (given that the value of E_{cl} is

correct).

2. It was shown in Part II that the design parameter in the TFL is the matrix H and that computer programs have been developed in this project which facilitate the calculation of H to shape the TFL in certain ways. Specifically, the programs allow the choice of matching the singular values of the TFL closed and open loop transfer functions $G_h(s)$ and $G_{clh}(s)$, defined in equations (2.3)(2.6) and (2.7), at low, high, or both low and high frequencies. Unfortunately, this shaping does not directly influence the appearance of $\sigma(sI - A_{clh})$ which is the expression of interest here. Figures (15) and (16) illustrate this fact. The same H used to generate Figure (15) was used for Figure (16).

Therefore, all that is left to the designer is the scalar used in the calculation of H which moves the singular plots up and down in value. The allowable bounds on the value of this scalar thus become the allowable bounds on the TFL design parameter, given whatever type of shaping is chosen. A computer program was developed which, over any specified range in this scalar, would indicate for what scalar values the above plots crossed (the inequality test failed) and for what values they did not cross (the inequality test held). The result is that the designer has a range of scalar values (and thus a range of H matrixes) for which he knows a system with good stability/robustness

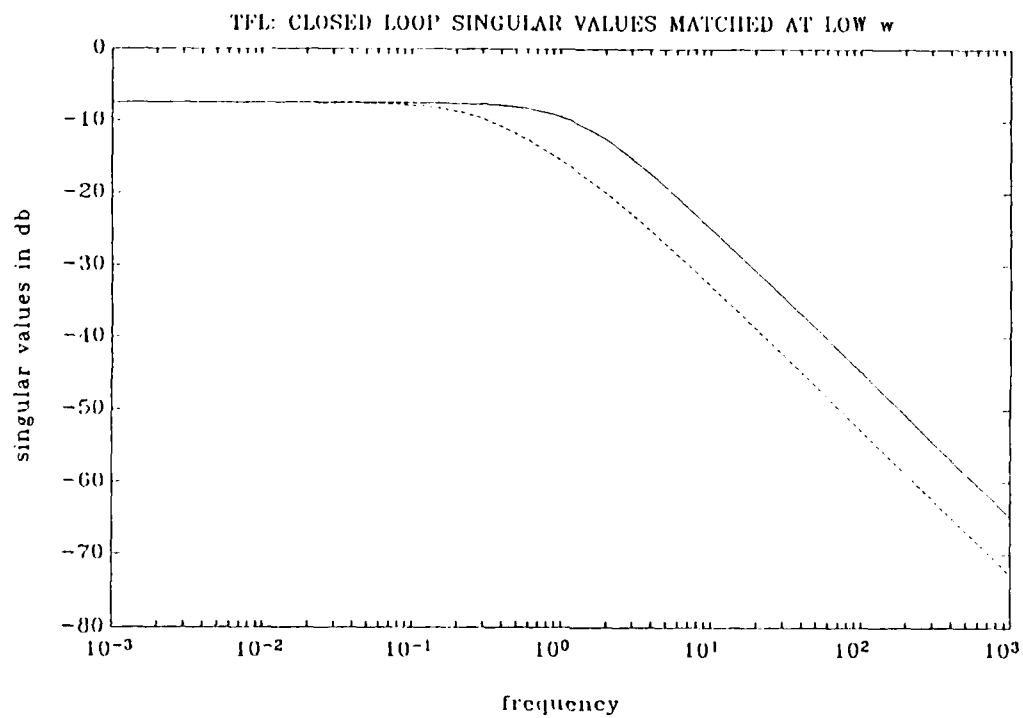


FIGURE (15)

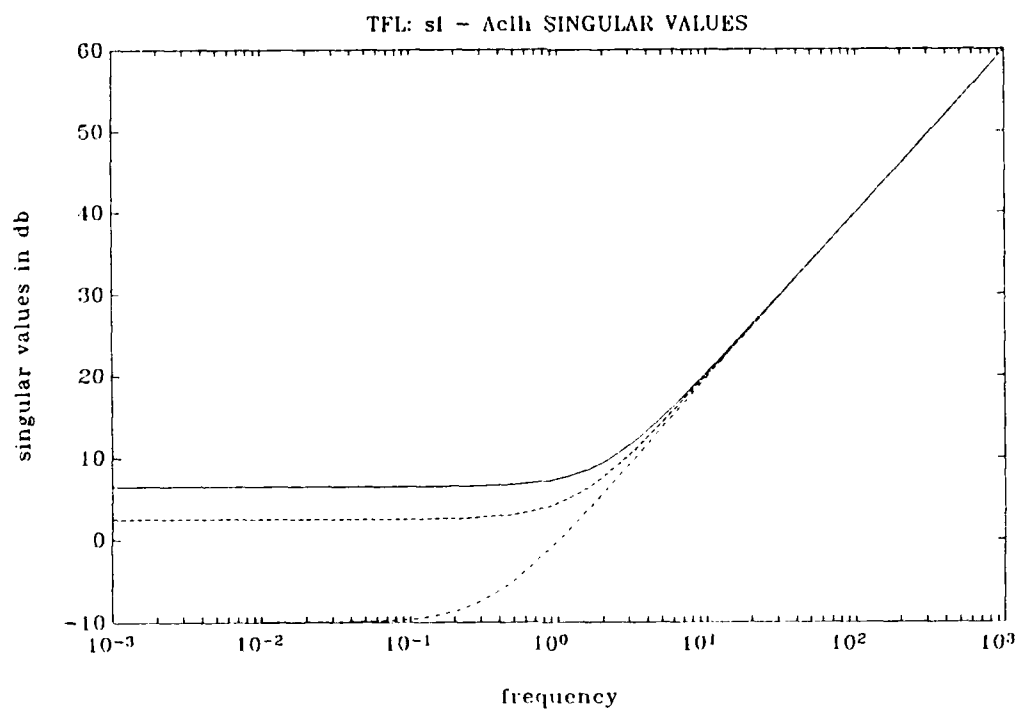


FIGURE (16)

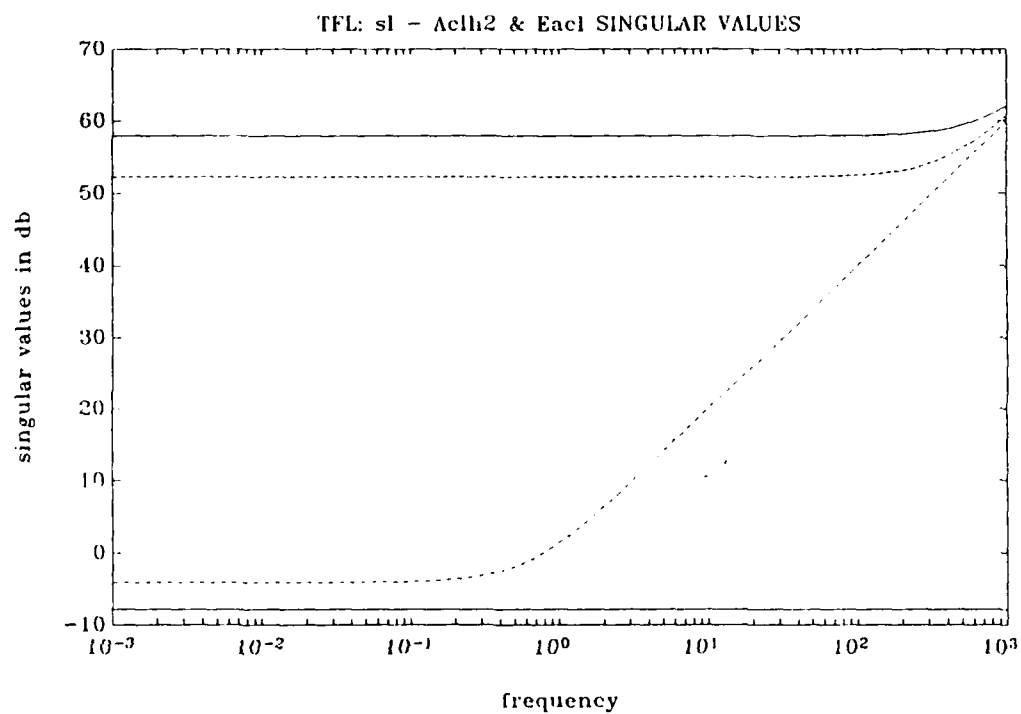


FIGURE (17)

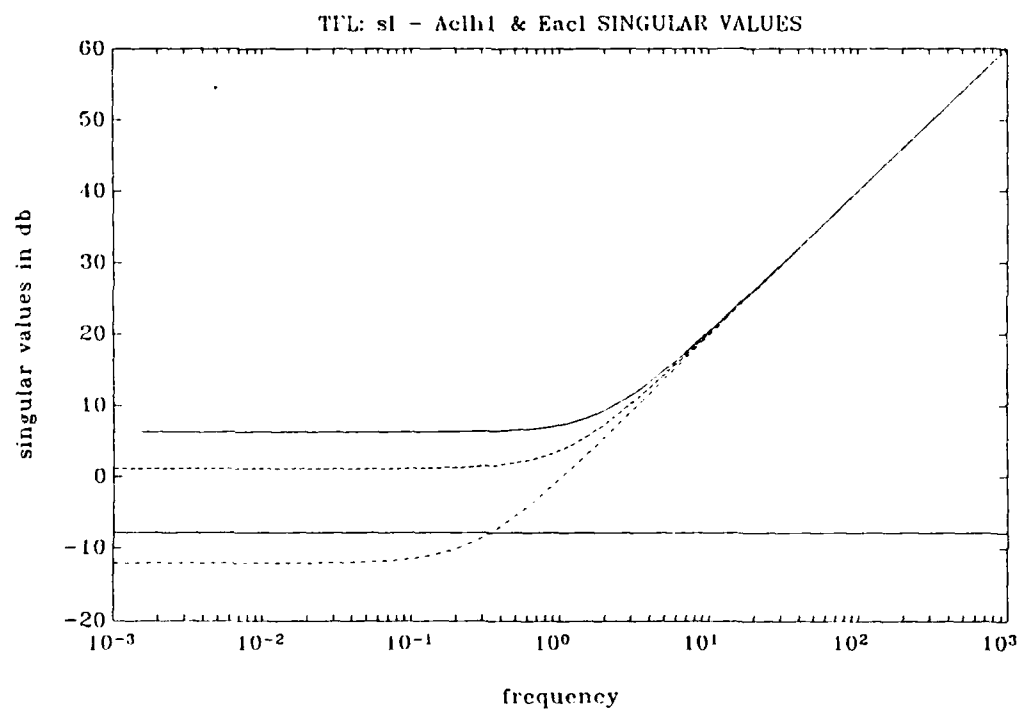


FIGURE (18)

will result. Figure (17) demonstrates a choice of a scalar that guaranteed a stable system while Figure (18) illustrates one that did not guarantee a stable system.

3. When actually implementing this method, the correct identification of E_{cl} is of obvious importance. As was discussed in Part I section B.3, the singular values of E_{cl} may vary with frequency due to the fact that the difference between the nominal and actual closed loop A matrices, $A_{cl}(\text{nom})$ and $A_{cl}(\text{act})$, varies with frequency. The singular value inequality test, which uses a constant E_{cl} calculated at low frequency only, is therefore not using the true error between the actual and nominal plants.

However, it was shown that one of the advantages of using the singular value inequality test method of this project versus other methods was that frequency effects on $(sI - A_{cl})$ can be seen graphically as in Figures (17) and (18) to be minimal (hence the flat plots) up to the mid-range frequencies. The point is that the designer would be able to directly calculate E_{cl} at low frequency according to equation (1.10) and yet have the stability/robustness properties given by the test remain relevant up through the mid-range frequencies. It is this low to mid-range frequency band that is usually where the system will operate due to performance considerations anyway (at high frequency performance such as command following rapidly deteriorates).

There is another justification for employing the Part I test inequality (as modified in equation (3.1)) which uses an error E_{a1} calculated at low frequency. When the TFL is recovered by the MBC, the closed loop singular values of the controlled system (MBC plus Plant) fall off at -40 db/decade instead of the -20 db/decade seen in the desired TFL. This is clearly evident in Figure (14). The result is that the recovery process inherently adds an extra margin of insulation against high frequency errors such as sensor noise. This high frequency region is the same range of frequencies over which the inequality test equation theoretically is no longer valid.

The ultimate solution is the formulation of a methodology which is applicable at all frequencies and which utilizes practical error information that the engineer can provide. However, it has been seen that for many applications the use of inequality (3.1) will provide meaningful if not absolutely guaranteed measures of stability/robustness through compensator design.

B. The MBC and TFL Shaping Using Eigenvalues

1. In Part I section A.1, it was shown that if the closed loop A matrix A_{c1} is known at any time, its eigenvalues can

be calculated and a direct stability determination made based on the presence of any eigenvalues with positive real parts. This serves as a foundation for another method that finds bounds on a design parameter in the TFL such that, when the MBC is recovered and added to the plant, a stable controlled system will be insured.

As was the case above, since design for the MBC is taking place in the TFL the closed loop A matrix A_{clh} from equation (2.8) becomes the term of interest. One of the guarantees inherent in the TFL is that closed loop stability which can be read directly from A_{clh} by means of its eigenvalues will always be present. This matrix, however, represents a nominal value in the sense that when the TFL operating characteristics are recovered by use of the MBC the resulting closed loop A matrix of the combined MBC plus nominal plant will approximate those of A_{clh} . Because stability/robustness hinges on how the controlled system will behave in actual operation (MBC plus actual plant) with errors present, it is the A_{clh} matrix subjected to the error between actual and nominal closed loop A matrices, or E_{acl} , that must be evaluated for stability.

2. Equation (1.10) provides the relationship between the closed loop actual and nominal A matrices and is also a definition of E_{acl} for the MIMO nominal plant model case. For design in the TFL using this method, the equation

becomes:

$$A_{clh}(act) = A_{clh}(nom) + Eacl \quad (3.2)$$

where $A_{clh}(nom)$ is identical to the A_{clh} described above and in Part II, and is equal to $A - HC$, with H being the design parameter of the TFL obtained from computer programs in Appendix B. As described in section 1 above, a scalar emerges as the design parameter available to the designer for the calculation of H .

The procedure for determining bounds on this scalar to insure stability of the actual system when the TFL is recovered by the MBC follows directly. If $A_{clh}(act)$ represents what the actual system (MBC plus actual plant) will behave like, $A_{clh}(act)$ can be calculated at any scalar value by first generating H from that scalar, calculating $A_{clh}(nom)$ directly from equation (2.8), and then applying equation (3.2). Once $A_{clh}(act)$ is obtained, its eigenvalues are found and a stability determination is made. A computer program was developed which would implement this procedure over a specified range of scalar values, thereby indicating which values of the scalar would insure a stable, actual controlled system.

3. The limitations of this method are much greater than those in the above method. First, while both methods allow

for only a constant E_{acl} determined at low frequency, it was shown both above and in Part I that the project singular value inequality technique has applicability at other frequencies. Without any factors to indicate that there might be applicability at higher frequencies in the eigenvalue method, however, it must be considered strictly limited to low frequency.

Second, the singular value test method insures stability by using $\sigma_{\max}(E_{acl})$, which by definition serves as a maximum upper boundary on the different errors that the plant may be subjected to. The test therefore takes all error possibilities into account in assuring actual stability. In the eigenvalue method, there is no such upper boundary and therefore while the designer may insure that the actual system will not be unstable with this particular error, he has little idea of what will happen when a different error is present (although an error of smaller size, as defined by the 2-norm, is less likely to cause instability).

For these reasons, this method is offered more as a source of additional information and as a check when utilizing the singular value test method for insuring stability/robustness. For instance, if for a particular design parameter constant value the eigenvalue method indicates an unstable system, then clearly the plots of

$\sigma_{\min}(sI - A_{clh})$ and $\sigma_{\max}(Eacl)$ should cross for that constant value as test equation (3.1) stipulates they must.

Part IV

Standard Design Process

Description

The primary objective of this Trident Project is to provide the practical engineer with certain methods which can be used to address the stability/robustness problem in MIMO control system design. It has been described how heretofore this basically was not a tractable problem and as a result certain guidelines were offered in Parts I and III (Part II describing methodology on which later results would be based) that can be used in the MIMO stability/robustness case. Whatever the merits of these guidelines, they have little value for the engineer unless they are integrated into a coherent process that achieves the design objective described above.

This part of the paper serves to realize such an integration and yield a standard design process which the engineer can follow when faced with a MIMO design problem. No effort will be made here to replicate the development and discussion of the methods in Parts I through III. Rather, it will be shown how these techniques fit together and it is up to the designer to go back in the paper and arrive at the full understanding necessary for their use. Figures

(19), (20), and (21) review the basic structure of the nominal plant alone, the target feedback loop (TFL) used in the MBC/LQG/LTR method, and the final controlled system (MBC plus plant) discussed earlier and referred to below.

Formation and Discussion

A. Examine MIMO Nominal Plant Model and Specify Error

1. It is necessary to have a full understanding of the nominal plant model as seen in Figure (19) before proceeding to the design of a control system. This information tells the designer (1) whether control is necessary at all, and (2) if plant operation is not satisfactory, what type and amount of control needs to be employed.

The first step is to assess the stability/robustness of the MIMO nominal plant model. Both the open loop and closed loop stability of the model are immediately determined by calculating the eigenvalues of the A and A_{C1} matrices. If any of the eigenvalues have positive real parts, then instability is present. All methodology presented in this paper requires that the plant have nominal closed loop stability. Failing that, either a new or revised plant should be obtained or new methodology sought by the designer. The project did not uncover any such workable

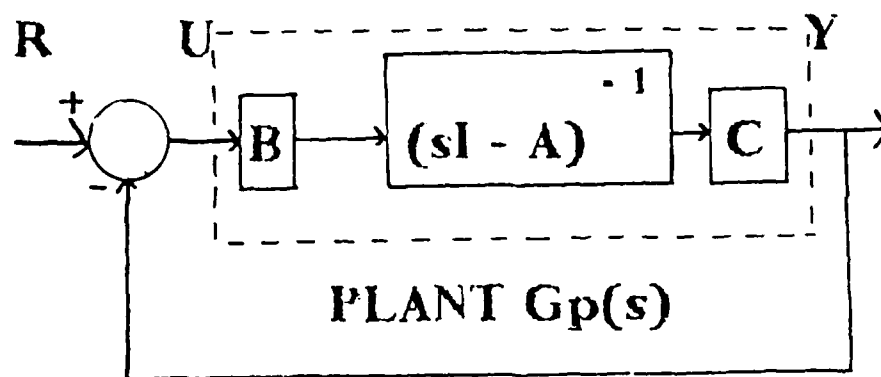


FIGURE (19)

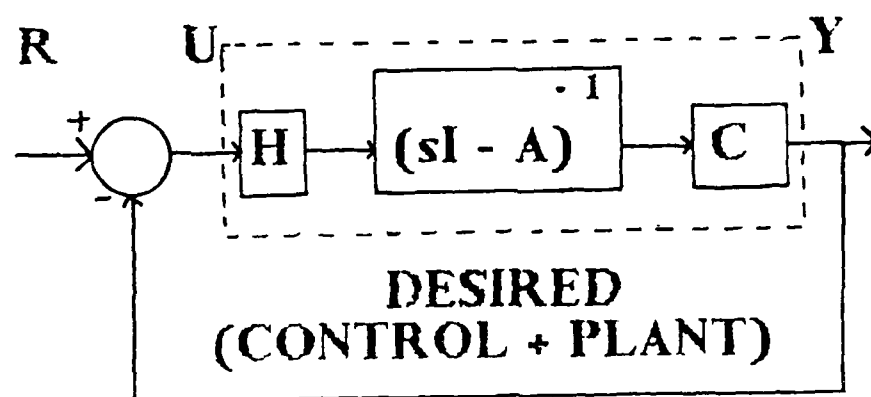


FIGURE (20)

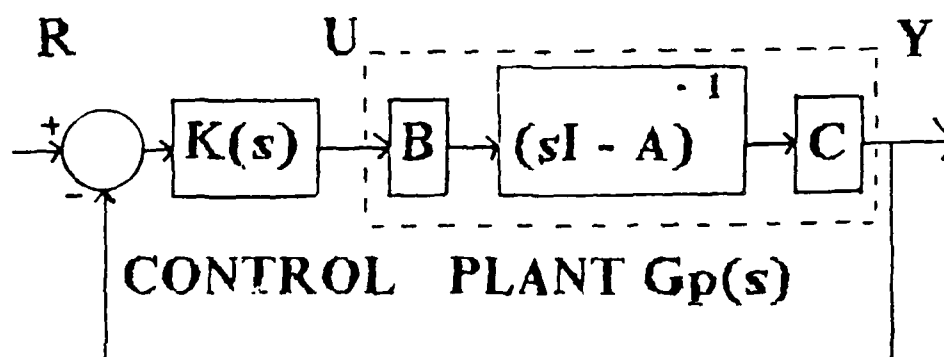


FIGURE (21)

methods in existence, making this an important area for future research.

Each of the methods presented in Part I to gauge the stability/robustness of the MIMO nominal plant model should now be employed. The use of the deviation of parameters method is straightforward and will give the designer a direct feel for how close the plant is to instability. For example, if only a five percent deviation between actual and nominal values can be tolerated in each of the significant plant parameters, then the designer knows that control will have to be added to make plant stability more resistant to such changes. The plant's stability/robustness must be enhanced. A precise determination of how much and in what way the plant must be modified to insure better stability/robustness is left to other methods outlined below.

The singular value inequality test method is used next. This method will also provide the designer with information about the nominal plant's stability/robustness but of a different nature than the deviation of parameters technique. If the test reveals either that actual plant stability is unsure or that it is guaranteed unstable, then control will be a necessity.

2. The other portion of the nominal plant analysis

involves a review of its performance characteristics. Open and closed loop singular value plots, linear simulations to different classes of inputs, and other measures of performance should be obtained. Because this paper is primarily concerned with stability, little more will be offered in the way of description in this area. The objective is to demonstrate where performance fits into the overall design process that has been developed.

3. The definition of the error E_{acl} between the actual and nominal closed loop A matrices (A and A_{cl}) used in the inequality test is all-important both here and below in the design of the TFL. The designer needs to determine the constant E_{acl} between the actual and nominal A_{cl} matrices at low frequency which is then extended across all frequencies. Justification for and limitation of this extension is offered in Parts I and III. As a last resort, the designer should calculate a worst error E_w as discussed in Part I and Appendix A.2 which represents the E_{acl} error the nominal plant is theoretically most sensitive to in terms of stability. The reason for using this worst error as a default lies in the fact that if when the compensator is designed below the worst case condition is taken into account, then all other operating conditions will be assured stable.

There is another factor in error determination.

Finding complete error knowledge which encompasses all possibilities at all operating conditions should be the ideal, but it is not likely in practice. Rather, emphasis is placed on determining the upper and lower bounds on this error which are represented by $\sigma_{\max}$ and $\sigma_{\min}$, respectively. This is the information used in the singular value inequality tests.

B. Design the TFL to meet Specifications

The analysis of the MIMO nominal plant tells the designer what changes will be necessary in order to meet the specifications for system operation. Part II describes the TFL illustrated in Figure (20) and explains that it is used to represent the desired behavior the system will exhibit when control in the form of a MBC is added. For this reason it is in the TFL that stability/robustness and performance requirements are met.

1. The first step is to design for good stability/robustness (it is highly unlikely that poor stability/robustness would be a system requirement) by determining a bound on the TFL design parameter. This process was described in Part III and involves use of the singular value inequality test as modified for the TFL. The error E_{acl} that is required by the inequality test was

described in section A above.

Particular attention should be paid to the situation where, in the nominal plant model assessment, the deviation of parameters method indicates the presence of poor stability/robustness and the inequality test method does not. Generally, this situation will mean the presence of poor high frequency stability/robustness which may go undetected in the latter method. As a result the designer will want to add extra degrees of resistance to high frequency errors. This is accomplished both by moving the closed and open loop TFL singular value plots (G_{clh} and G_h) down in value and by restricting the bandwidth of the closed loop plot.

2. The outcome of this stability/robustness assessment for the TFL is a boundary on the design parameter available to change TFL operating characteristics. This design parameter was shown in Part II to be a scalar that the designer specifies when utilizing the software in Appendix B to realize the TFL. The process now becomes:

- a. Pick a scalar value from within the allowable bounds for good stability/robustness.
- b. Use Appendix B computer programs to calculate the design matrix H and produce the TFL singular value plots, linear simulation plots, or whatever else is desired to analyze performance.
- c. Is this performance satisfactory?

- d. If so, the process is complete.
- e. If not, the designer should pick a new scalar within the specified bounds that will achieve performance specifications. Return to step b.

C. TFL Recovery with the MBC

1. Having realized the TFL and decided that its operating characteristics are those that the engineer desires the controlled system (compensator plus plant) to possess, the compensator must be designed. Within the MBC/LQG/LTR methodology that is used in this project, this compensator is referred to as a MBC and its formation is accomplished as was described in Part II. Essentially, the engineer must use Appendix B software to calculate a matrix G that, within the MBC structure shown in Figure (21), will to some degree of accuracy recover TFL behavior once the MBC is appended to the plant.

2. The process that yields a desirable G and therefore a MBC is:

- a. Pick a value for the scalar used to create G .
- b. Use Appendix B computer programs to explicitly calculate G and form the MBC.
- c. Analyze the resulting controlled system in terms of both stability/robustness and performance to see if the TFL was effectively recovered.

- d. Check the magnitude of the controls in the MBC during system operation.
- e. Is this MBC satisfactory upon the analysis provided in steps c and d?
- f. If so, the process is complete.
- g. If not, either pick a new scalar value for a better recovery (lower in value) or pick a new scalar for a poorer recovery that will reduce control magnitudes (higher in value).

D. Testing

Little explanation is required here. Simply test the controlled system in all types of operation, particularly with regard to the presence of errors between the nominal plant model and the actual plant. When errors that were described in section A above and used in section B for TFL design are applied, the controlled system should always remain stable. When errors that were not provided for are applied to the controlled system, the behavior that is observed should be either stable or unstable depending on the nature of the error applied.

PART V

A Design Example

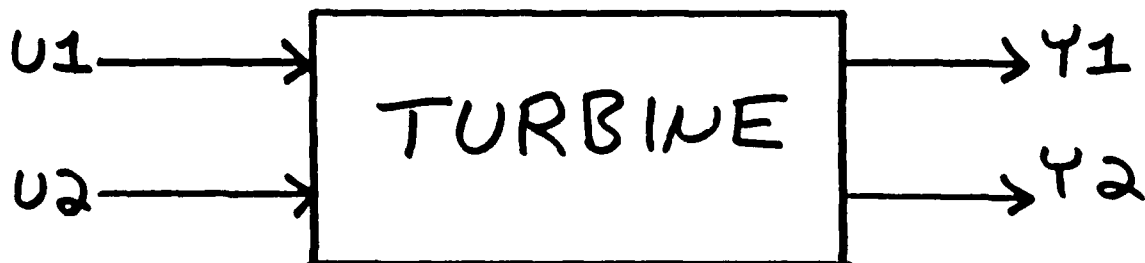
Description

An example is offered to elucidate both the concepts of Parts I to III and their integration in the standard design process of Part IV. Hopefully, the example will facilitate the understanding necessary for the practical control engineer to employ the methods presented in this paper in real-world applications. The example does not illuminate every facet of the methodology that has been offered in this Trident Project but it does provide a basic summary of the process. Each application is unique and the literature, incorporating both this paper and those in the References, should be consulted extensively to insure methods are being applied correctly.

It is stressed that this example is entirely fictitious. No attempt should be made to compare it to an actual turbine in any way. All the variables, parameters, and relationships have been arbitrarily chosen and named. In this sense the example could best be termed a "generic" MIMO design problem. The use of the turbine plant name and all other names is done purely for the benefit of the reader. The intended result is the elimination of any

"GENERIC" TURBINE EXAMPLE

DIAGRAM



DESCRIPTION

INPUTS : U1 = DESIRED SPEED

U2 = LOAD TORQUE

OUTPUTS: Y1 = PER UNIT TURBINE ROTOR SPEED

Y2 = PER UNIT FUEL FLOW

PARAMETERS: (1) VALVE POSITIONER GAIN
 (2) SPEED GOVERNOR GAIN
 (3) FUEL SYSTEM FEEDBACK GAIN
 (4) FUEL SYSTEM TIME CONSTANT

EQUATIONS: $\dot{X} = AX + BU$, $Y = CX$
 A,B,C = NOMINAL SYSTEM MATRICES

FIGURE (22)

potential confusion caused by numerous and indistinguishable inputs, outputs, and parameters.

Presentation and Discussion

A. Nominal Plant Analysis

Figure (22) summarizes the MIMO turbine nominal plant model. The plant is comprised of two inputs, two outputs, and four parameters and the mathematical formulation is identical to that used in the previous parts of the paper. The plant possesses nominal closed loop stability.

Figures (23) and (24) summarize the nominal plant stability/robustness properties. The singular value test inequalities (1.12) to (1.14) applied to Figure (23), where the horizontal plot represents $\sigma_{\max}(E_{cl})$, indicate that the actual plant alone (no control) could be either stable or unstable. Recall that E_{cl} is the error between the actual and nominal closed loop A matrices at low frequency. Figure (24) is the result of applying the deviation of parameters method. The original four parameters have been reduced to two, the speed governor gain and fuel system time constant, with significant effects on stability. Here especially poor stability/robustness is apparent. The combination of deviations that, simultaneously, limits the

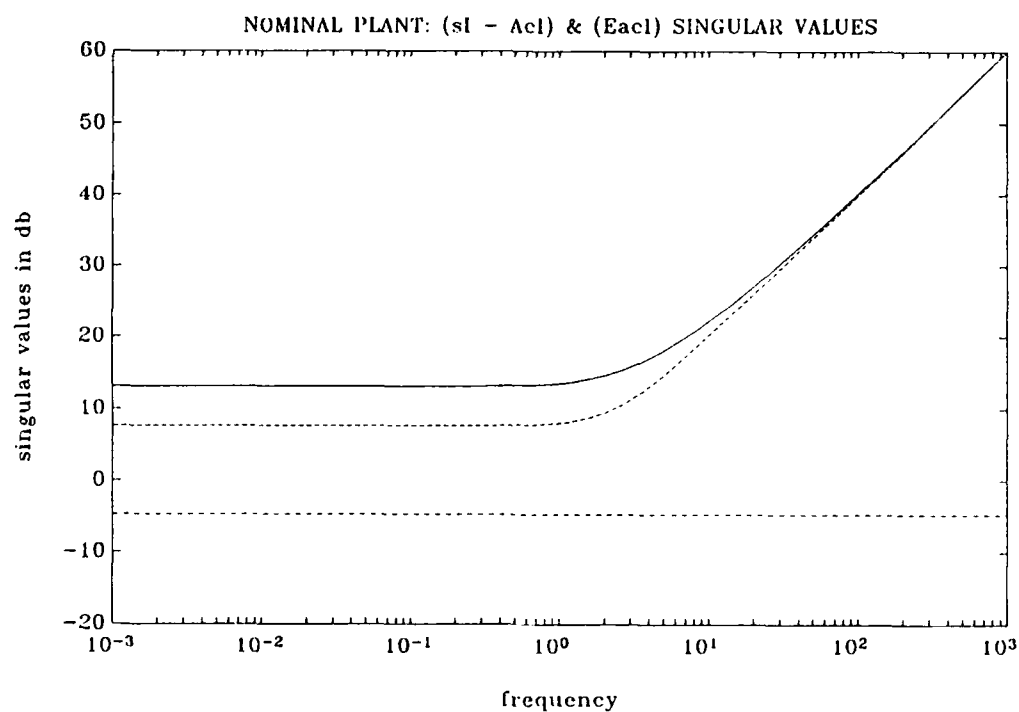


FIGURE (23)

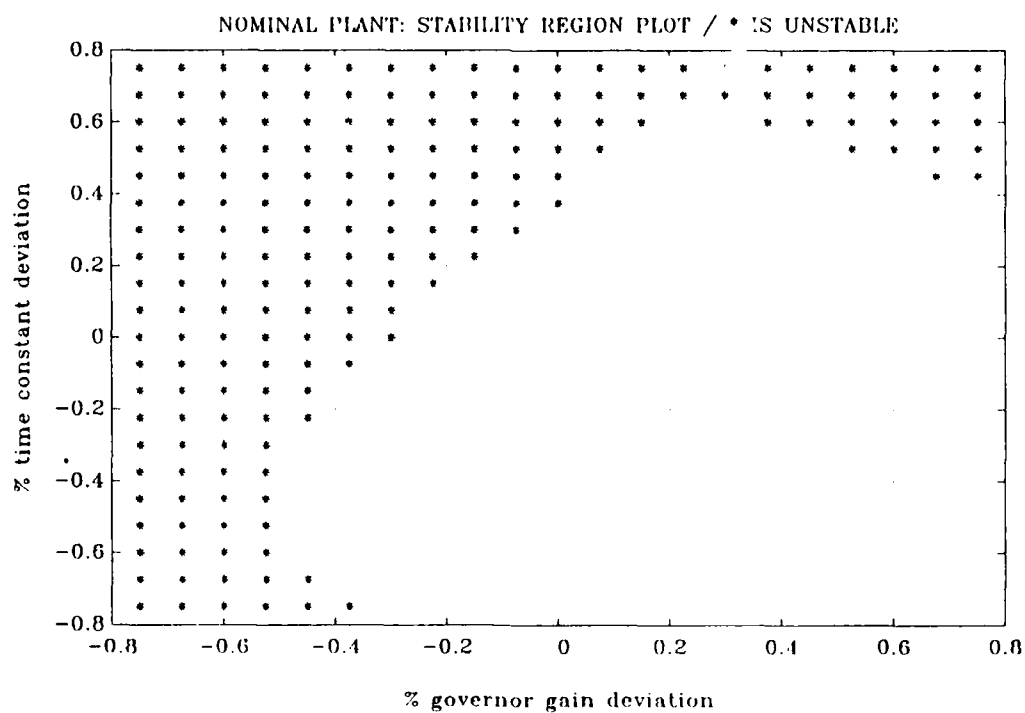


FIGURE (24)

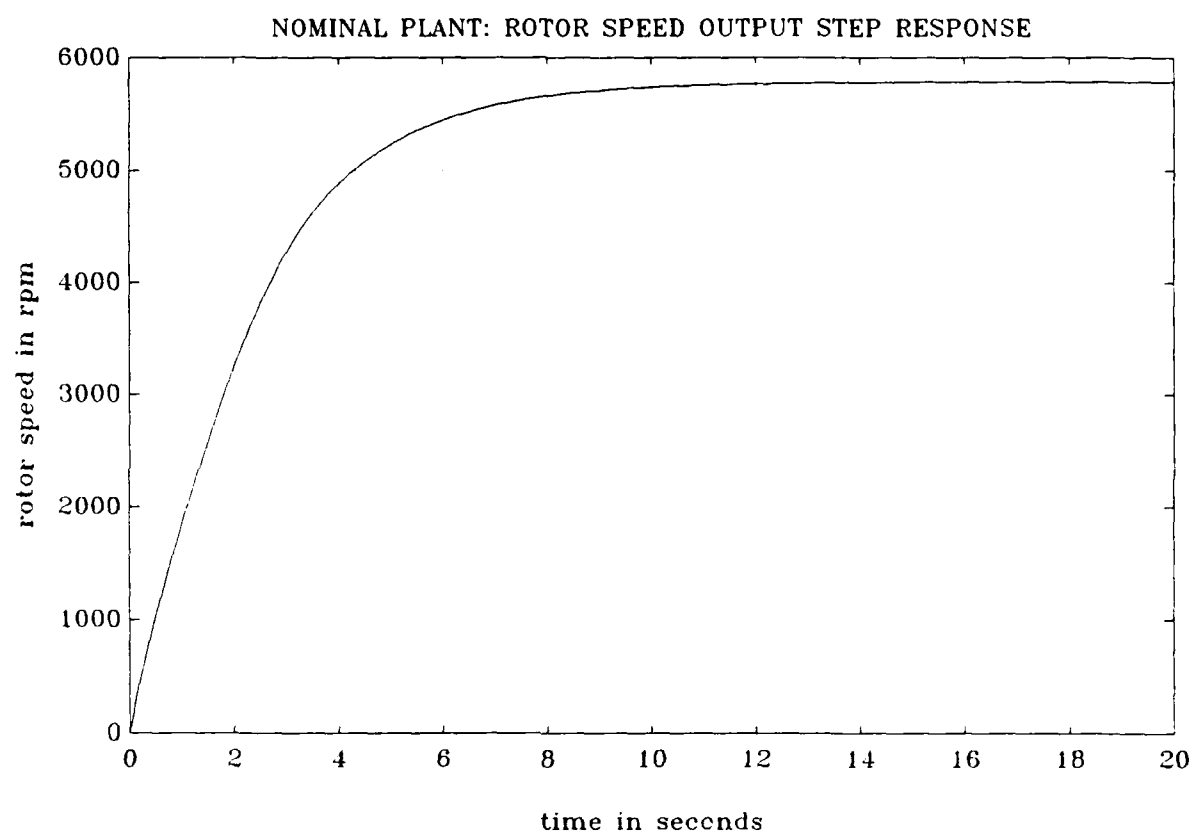


FIGURE (25)

governor gain to 10% deviation and the time constant to 30% deviation barely preserves stability.

Figure (25) illustrates the analysis of a nominal plant performance condition, the rotor speed output when a step is applied at the desired speed input. One notes that the 10 second time to peak represents a rather sluggish system response.

A summary can now be made of the turbine MIMO nominal plant model. Clearly, its stability/robustness properties need to be improved through control. The nominal plant, in conjunction with the error expected, in no way guarantees stable operation for the turbine. A properly designed compensator in the standard control feedback loop needs to be appended to the plant to insure the good stability/robustness that must exist in a plant such as a turbine.

It was also observed that in at least one key performance characteristic, the rotor speed step response, nominal plant performance was also deficient. The specification that this rotor step response must be improved will be added to the obvious one of good stability/robustness. Specifically, the requirements of a 10 rad/sec bandwidth to insure a fast time to peak (or response time) and 0db closed loop unity gain between

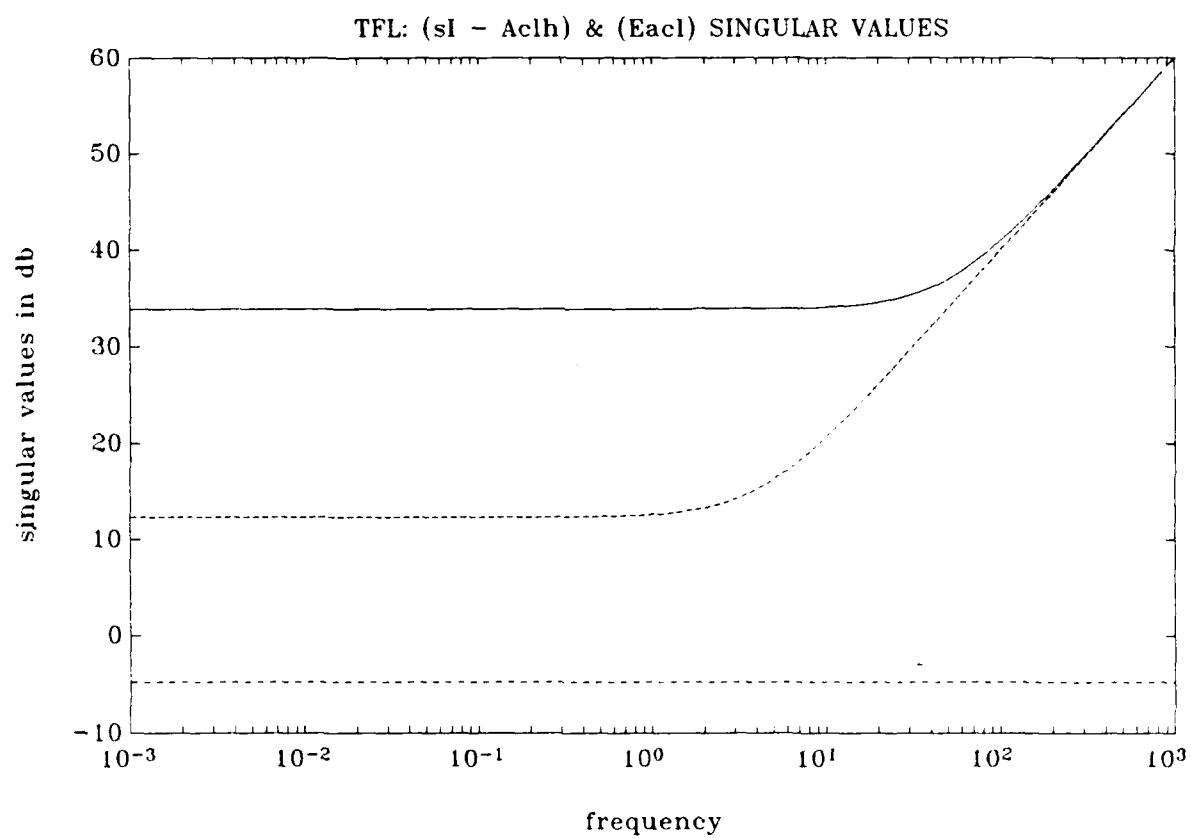


FIGURE (26)

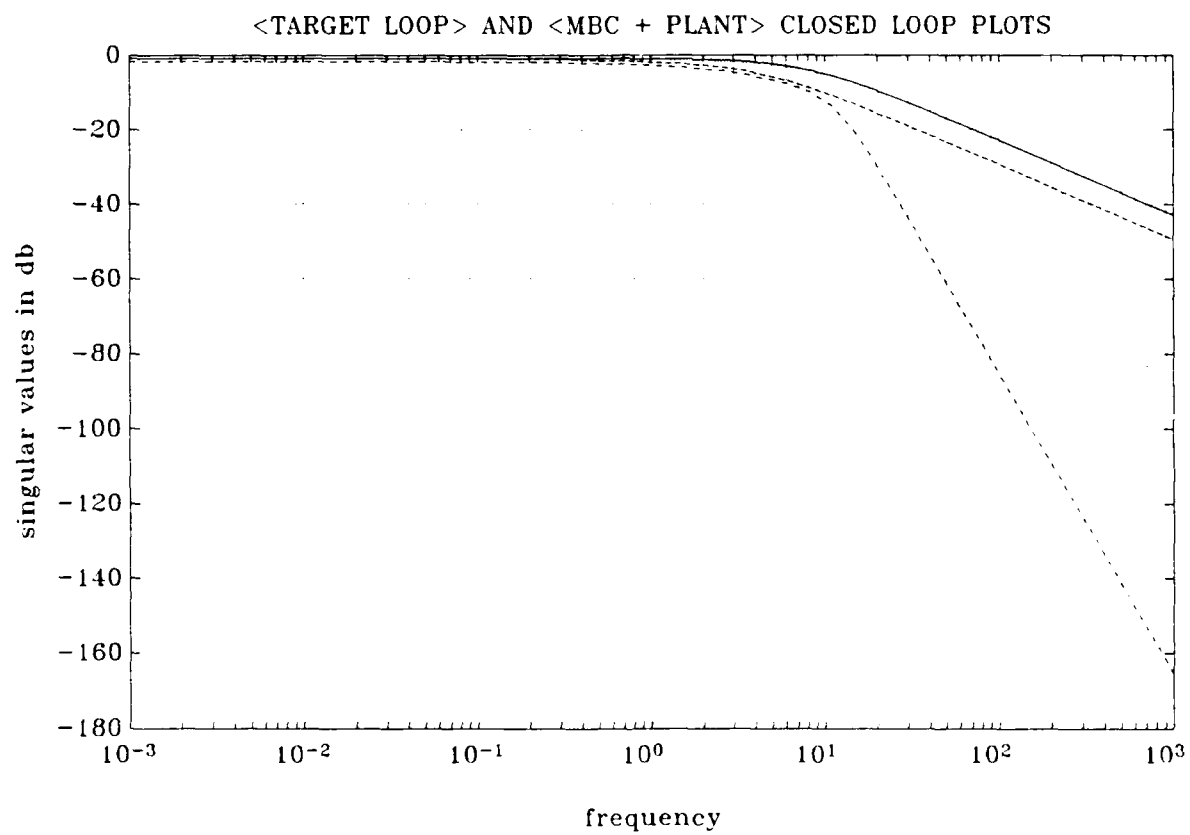


FIGURE (27)

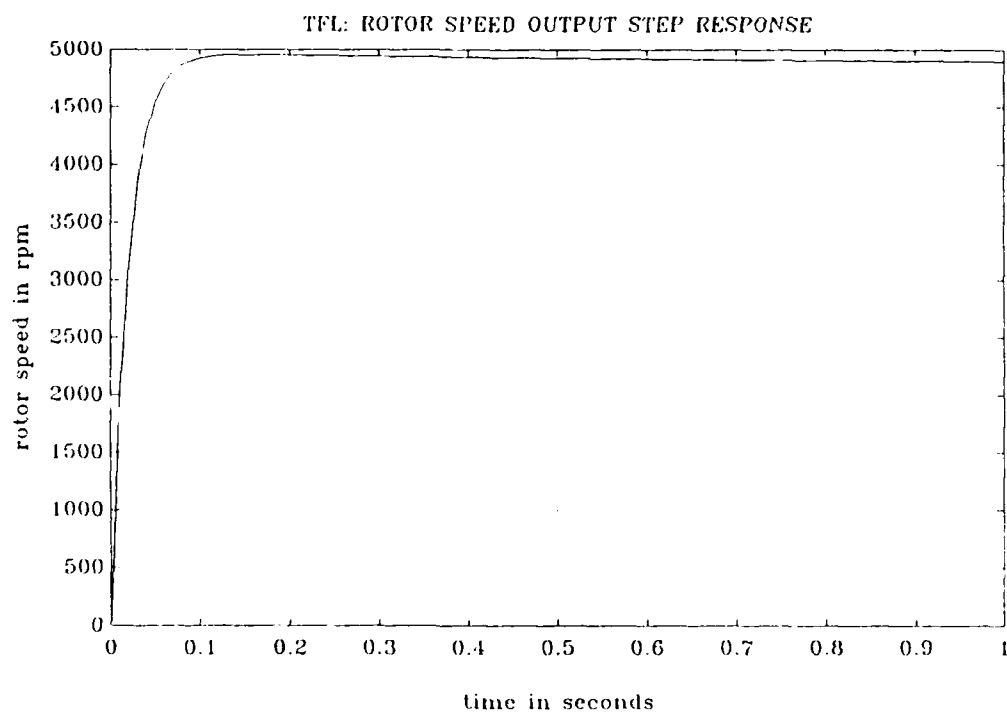


FIGURE (28)

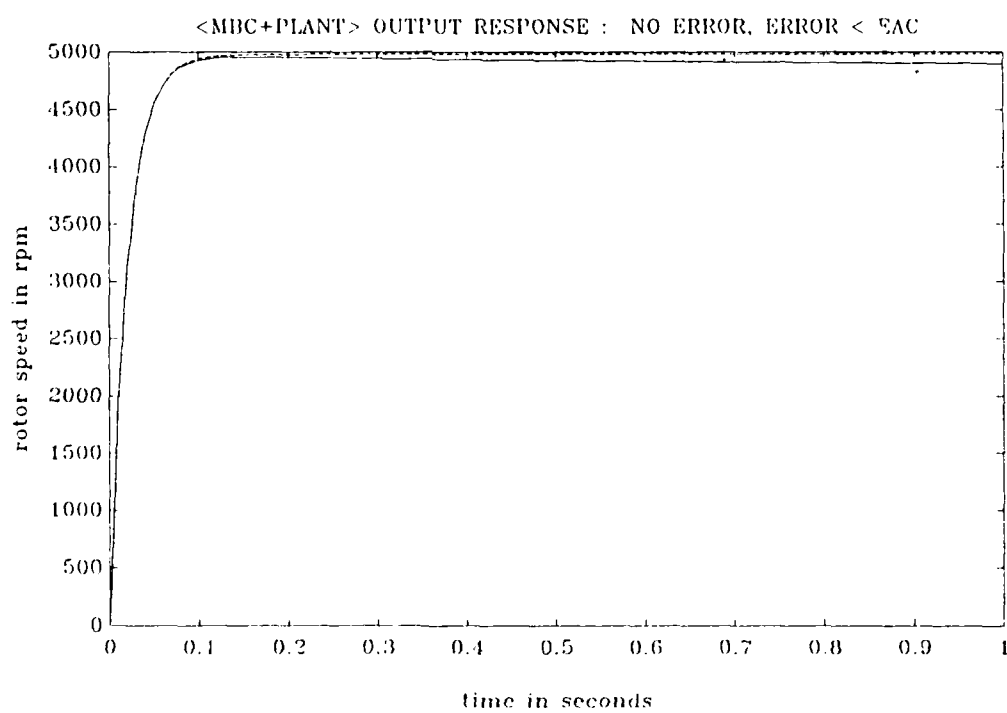


FIGURE (29)

reference input and rotor output at low to mid frequencies are made.

Figure (26) shows the design in the Target Feedback Loop (TFL) to meet singular value test inequality (3.1) and thus insure good stability/robustness in the controlled system upon recovery of the MBC. The scalar chosen to obtain the TFL design parameter (H matrix) that gave Figure (26) was one of a range a possible values provided by Appendix B programs that would insure good stability/robustness. It specifically was chosen and used from that range of values because it was the scalar that would give Figure (26) and result in the TFL closed loop singular value plot shown in Figure (27). The controlled system closed loop singular value plot, also in Figure (27), demonstrates the near recovery of the TFL when the MBC is added to the plant. Note that the closed loop plot meets the design requirements of 0db unity gain at low to mid frequencies and 10 rad/sec bandwidth.

Figure (28) shows the TFL rotor step response which represents an improvement over the nominal plant behavior due to the meeting of the closed loop performance requirements in Figure (27). Figure (29) illustrates two points. First, the near recovery of the TFL in the controlled system in terms of output response is readily apparent when Figures (28) and (29) are compared.

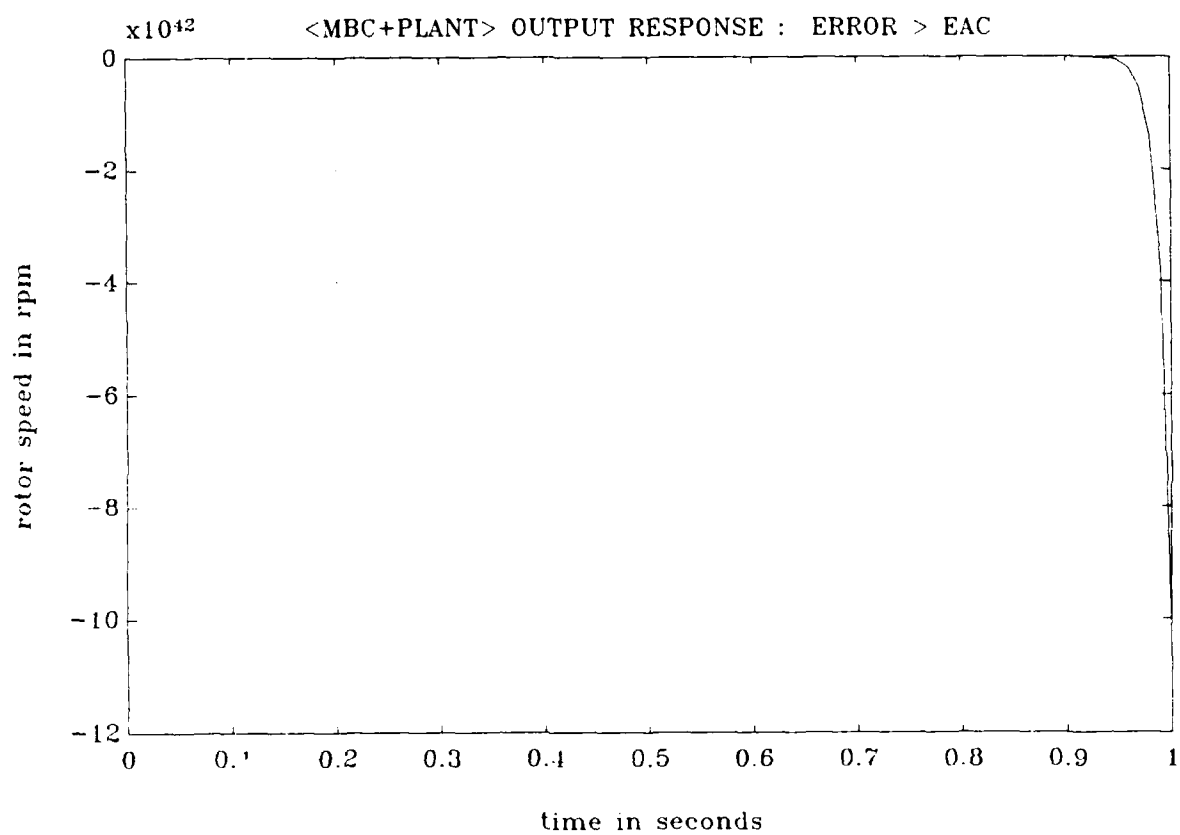


FIGURE (30)

Second, Figure (29) includes both an output response with no error and one with error present. The controlled system's output response when subjected to an error that was included in the Eacl used for the stability/robustness test is not unstable as was expected. Figure (30), however, is an indication of what can happen when Eacl is not properly specified. In this case an error not included in the Eacl used for the stability/robustness test was applied. A highly unstable response is evident.

CONCLUSIONS

From the investigation outlined in this paper, the following conclusions can be drawn about control system design with desirable stability/robustness properties for MIMO plants:

1. Singular values are an effective tool with which to describe MIMO system behavior by bounding all possibilities of operation inherent in the MIMO case.
2. Error information for stability/robustness singular value inequality tests developed prior to this investigation was seen to be prohibitively hard for the engineer to obtain.
3. The singular value inequality test method detailed in this paper provides good stability/robustness information about MIMO systems at low to mid frequencies and may be relevant at higher frequencies in some cases.
4. The error information required by the project inequality test is more readily obtained by engineer than that required by previous techniques.
5. The deviation of parameters method developed in this project provides valuable information about

the nearness to instability of a MIMO nominal plant model in terms of allowable parameter variations; however, the method is limited to plants with relatively few parameters that impact significantly on stability.

6. The MBC/LQG/LTR design methodology is an effective tool for MIMO control system formulation but its ability to insure good stability/robustness hinges on the availability of usable Target Feedback Loop (TFL) singular value inequality tests.
7. The project procedures for stability/robustness evaluation can be effectively used within the MBC/LQG/LTR design framework by providing the TFL tests that are needed.
8. The formulation of a standard design process that combines all elements of both project and MBC/LQG/LTR methodology enables the engineer to tackle the MIMO control system design problem from start to finish.
9. Performance requirements need not be disregarded to insure good stability/robustness and are provided for in the project standard design process.

SUGGESTIONS FOR FUTURE RESEARCH

Subsequent research into multivariable control system design methods that insure good stability/robustness should consider:

1. Singular value inequality tests that will preserve stability/robustness at all frequencies and are based on practical error information obtainable by the engineer.
2. Ways to determine allowable parameter variations which maintain stability when numerous parameters with significant impacts on stability are present.
3. The capability to handle plants which do not possess nominal closed loop stability, as both project and MBC/LQG/LTR methodology require that the plant possess nominal closed loop stability.
4. Use of nonlinear chaotic systems theory to assess actual plant operation and thus generate the error between actual and nominal plants that stability/robustness tests require.

ACKNOWLEDGEMENTS

The author wishes to thank those people without whom this Trident Research Project would never have been able to either begin or end:

- * Professor E.E. Mitchell, Chairman of the USNA Weapons and Systems Engineering Department, who served as advisor for this Trident Project and provided invaluable assistance from start to finish.
- * Dr. R. DeMoyer of the USNA Weapons and Systems Engineering Department who was always ready and able to lend whatever assistance was required.
- * All members of the USNA Weapons and Systems Engineering Technical Support Staff who never said no.
- * And, lastly, to my roommates who were willing to sleep through almost anything and who somehow managed not to throw me out the window.

REFERENCES

- [1] M. Athans, "A Tutorial on the LQG/LTR Method," Proc. American Control Conference, Seattle, WA, June 1986, pp. 1-8.
- [2] M. Athans, P. Kapassouris, E. Kappos, and H.A. Spang III, "Linear-Quadratic-Gaussian with Loop-Transfer-Recovery Methodology for the F-100 Engine," AIAA Journal of Guidance, Control and Dynamics, Vol. 9, January 1986, pp.45-52.
- [3] J.C. Doyle and G. Stein, "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis," IEEE Trans. on Automatic Control, Vol. AC-26, No. 1, February 1981, pp. 4-16.
- [4] J.C. Doyle and G. Stein, "Singular Values and Feedback: Design Examples," 15th Annual Allerton Circuits and Systems Conference, October 1978, p.461.
- [5] N.A. Lehtomaki, "Practical Robustness Measures in Multivariable Control System Analysis," Ph.D. Thesis, LIDS-TH-1093, MIT, Dept. of Electrical Engineering and Computer Science, Cambridge, Mass., May 1981.
- [6] N.A. Lehtomaki, N.R. Sandell, Jr., and M. Athans, "Robustness Results in Linear-Quadratic-Gaussian Based Multivariable Control Designs," IEEE Trans. on Automatic Control, Vol. AC-26, February 1981, pp. 75-92.
- [7] R. Martin, L. Valavani, and M. Athans, "Multivariable Control of a Submersible Using the LQG/LTR Design Methodology," Proc. American Control Conference, Seattle, WA, June 1986.
- [8] W.H. Pfeil, M. Athans, and H.A. Spang III, "Multivariable Control of the GET700 Engine Using the LQG/LTR Design Methodology," Proc. American Control Conference, Seattle, WA, June 1986.
- [9] M. Safanov, "Tight Bounds on the Response of Multivariable Systems with Component Uncertainty," 15th Annual Allerton Circuits and Systems Conference, October 1978, p. 451.
- [10] G. Stein and M. Athans, "The LQG/LTR Procedure for Multivariable Feedback Control Design," Report LIDS-P-1384, MIT Lab. for Information and Decision Systems, Cambridge, Mass., May 1984.

APPENDIX A

A-1. Singular Values (σ) [6]

The singular values of a square $n \times n$ complex matrix A , denoted by $\sigma_i(A)$, are defined as

$$\sigma_i(A) = \text{eig}_i^{\frac{1}{2}}(A^H A) = \text{eig}_i^{\frac{1}{2}}(A A^H) \quad (\text{A.1})$$

where A^H denotes the complex conjugate transpose of A and $\text{eig}_i(A^H A)$ the i th largest eigenvalue of $A^H A$. A way of representing the matrix A , known by the singular value decomposition (SVD) is given by

$$A = U S V^H = \sum_{i=1}^n \sigma_i(A) u_i v_i^H \quad (\text{A.2})$$

where

$$U = [u_1, u_2, \dots, u_n]; \quad U^H U = I \quad (\text{A.3})$$

$$V = [v_1, v_2, \dots, v_n]; \quad V^H V = I \quad (\text{A.4})$$

$$S = \text{diagonal}[\sigma_1, \sigma_2, \dots, \sigma_n] \quad (\text{A.5})$$

and the columns of V and U are eigenvectors of $A^H A$ and $A A^H$, respectively. The minimum and maximum singular values denoted $\sigma_{\min}$ and $\sigma_{\max}$, respectively, are sometimes equivalently defined in terms of the spectral matrix norm $\|\cdot\|$ as

$$\sigma_{\max}(A) = \max_{\|x\|_2=0} \frac{\|Ax\|_2}{\|x\|_2} = \|A\|_2 \quad (\text{A.6})$$

and

$$\begin{aligned} \sigma_{\min}(A) &= \min_{\|x\|_2=0} \frac{\|Ax\|_2}{\|x\|_2} \quad (\text{A.7}) \\ &= \|A^{-1}\|_2^{-1}, \quad \text{if } \det A \neq 0 \\ &= 0, \quad \text{if } \det A = 0 \end{aligned}$$

The minimum singular value $\sigma_{\min}(A)$ provides a measure of the nearness to singularity of the matrix A in the following sense. If $A+E$ is singular then

$$\|E\|_2 = \sigma_{\max}(E) \geq \sigma_{\min}(A) \quad (\text{A.8})$$

Some other useful singular values facts are

$$\sigma_{\max}(A + B) \leq \sigma_{\max}(A) + \sigma_{\max}(B) \quad (\text{A.9})$$

$$\sigma_{\min}(A) = \frac{1}{\sigma_{\max}(A^{-1})} \quad (\text{A.10})$$

$$\text{if } \sigma_{\min}(A) \rightarrow 0, \text{ then } \text{eig}_{\min}(A) \rightarrow 0 \quad (\text{A.11})$$

A-2. Calculation of Ew

Ew represents the error that the matrix $A_{C1}(\text{nom})$ is most sensitive to in terms of stability as defined by

$$A_{C1}(\text{act}) = A_{C1}(\text{nom}) + Ew$$

The minimum eigenvalue of any matrix A (absolute value, assuming all have negative real parts) is an indicator of how close A is to instability (an eigenvalue with a positive real part). Also, (A.11) indicates the relationship between $\text{eig}_{\min}(A)$ and $\sigma_{\min}(A)$. Therefore

If : $\sigma_{\min}(A_{C1}) \rightarrow 0$
Then : The matrix A_{C1} will be on the border of stability

This leads to the conclusion that there must be some matrix Ew, that when added to $A_{C1}(\text{nom})$, will place $A_{C1}(\text{act})$ just on the border of stability by making $\sigma_{\min}(A_{C1}(\text{act}))$ go to zero. Such an Ew is calculated by

$$Ew = -\sigma_{\min}(A_{C1}(\text{nom})) * u_{\min} * v_{\min}^H \quad (\text{A.12})$$

where $u_{\min}$ corresponds to u_1 (A.3) and $v_{\min}$ corresponds to v_1 (A.4) which are obtained from the SVD of $A_{C1}(\text{nom})$.

The deviation of parameters method in Part I section C.2 uses Ew to identify parameters which impact significantly on stability. However, the method requires an Ew that is physically meaningful. By physically meaningful, it is meant that Ew should only have elements that correspond to elements in the A_{C1} matrix that will change when parameters are deviated.

The effect of eliminating elements of Ew that are not physically meaningful is that, when added to $A_{C1}(\text{nom})$, $A_{C1}(\text{act})$ will generally no longer be on the border of instability. To rectify the problem, Ew(old) is first calculated according to (A.12), then elements that are not physically meaningful are eliminated, and finally a constant x seen by

$$Ew(\text{new}) = -\sigma_{\min}(A_{C1}(\text{nom})) * u_{\min} * v_{\min}^H / x \quad (\text{A.13})$$

is generated by computer routines that will make Ew(new) take $A_{C1}(\text{act})$ to the border of instability just as Ew(old) did.

A-3. Proof of Singular Value Inequality Test (1.12)

The inequality test is:

IF : $\sigma_{\min}(SI - A_{C1}(\text{nom})) > \sigma_{\max}(E_{aC1})$
 THEN : The actual system is guaranteed stable at low w

As $w \rightarrow 0$, $(SI - A_{C1}(\text{nom})) \rightarrow -A_{C1}(\text{nom})$

Proof:

(A.8) states that if $(A + E)$ is singular then

$$\sigma_{\max}(E) \geq \sigma_{\min}(A)$$

This implies that if $(A + E)$ is nonsingular the following inequality must hold:

$$\sigma_{\min}(A) > \sigma_{\max}(E)$$

Equation (1.10) defines the relationship between E_{aC1} and $A_{C1}(\text{nom})$:

$$A_{C1}(\text{act}) = A_{C1}(\text{nom}) + E_{aC1}$$

For stability, $A_{C1}(\text{act})$ must be nonsingular which gives:

$$\sigma_{\min}(A_{C1}(\text{nom})) > \sigma_{\max}(E_{aC1})$$

Furthermore it can be shown that

$$\sigma_{\min}(A_{C1}(\text{nom})) = \sigma_{\min}(-A_{C1}(\text{nom}))$$

Combining the above:

$$\sigma_{\min}(-A_{C1}(\text{nom})) > \sigma_{\max}(E_{aC1})$$

which is a proof of inequality test (1.12) as $w \rightarrow 0$

APPENDIX B

MBC/LOG/LTR Method Function Files

AUGINT	NOTSEMID	RETSS
CLMBCMAT	QCOMPNG	SVCLINV
CLYFMATR	QMBCDES	SVPLOTS
COMPNG	QMBCDESA	SVRETDIF
MBCDES	QNTSEMID	TARGHI
MBCDESA	QTARG	TARGHLA
MBCMTRIX	QTARGA	TARGLOW
MBCSTEPS		

Deviation of Parameter Method Function Files

DEVPAR1	DIRECT	FINDX
DEVPAR2	DRXTOEW	REDUCEA
DEVPAR3	EWTDVP	RETSS

Singular Value Inequality Method Function Files

EATOMU1	SVEATOMU	SVPHI
RETSS		

General Purpose Function Files

AUGINT	NOTSEMID	SVCLINV
CLYFMATR	QNTSEMID	SVPHI
DIRECT	REDUCEA	SVPLOTS
DRXTOEW	RETSS	SVRETDIF
FINDX		

Example State Space Representation Function Files

APROB	BPROB	NETRET
-------	-------	--------

APROB

Command Line

```
[a,b,c,d] = aprob(v)
```

Description

this function provides an example set of a,b,c, and d matrices which can be used to learn and test other functions provided in this package

this example is an RC network with 3 resistors (r1...r3) and 3 capacitors (c1...c3)

calculates a,b,c & d matrices for the resistor and capacitor values entered as input arguments; contains 1 input, 3 states and 1 output

v = vector containing parameters:

```
    r1=v(1),,,r3=v(3),c1=v(4),,,c(3)=v(6)
```

```

function [a,b,c,d] = aprob(v)
%APROB [a,b,c,d] = aprob(v)

r1=v(1);r2=v(2);r3=v(3);c1=v(4);c2=v(5);c3=v(6);

a = [-1/(r1*c1)-1/(r2*c1)  1/(r2*c1)  0
      1/(r2*c2)           -1/(r2*c2)  0
      0                   0           -1/(r3*c3)];

b = [1/(r1*c1); 0 ; 1/(r3*c3)];
c = [0 1 0];
d = [0];

```

AUGINT

Command Line

[aa,ba,ca,da] = augint(a,b,c,d)
or [aa,ba,ca] = augint(a,b,c)

Description

augments plant matrices by placing an integrator
at each plant input channel

the result is a system which will generate a zero
steady-state error when subjected to constant
command or disturbance inputs

a,b,c,d = original plant matrices
aa,ba,ca,da = augmented plant matrices

a (n,n) ---> aa (n+m,n+m)
b (n,m) ---> ba (n+m,m)
c (m,n) ---> ca (m,n+m)
d (m,m) ---> da (m,m)

```
function [aa,ba,ca,da] = augint(a,b,c,d)
%AUGINT  [aa,ba,ca,da] = augint(a,b,c,d)
%        or [aa,ba,ca] = augint(a,b,c)
```

```
[n,m] = size(b);
aa = [zeros(m,m) zeros(m,n); b a];
ba = [eye(m); zeros(n,m)];
ca = [zeros(m,m) c];
narg = nargin;
if ( narg == 4 )
    da = d;
end
```

BPROB

Command Line

```
[a,b,c,d] = bprob(v)
```

Description

this function provides an example set of a,b,c, and d matrices which can be used to learn and test other functions provided in this package

this example is an RC network with 3 resistors (r1...r3) and 3 capacitors (c1...c3)

calculates a,b,c & d matrices for the resistor and capacitor values entered as input arguments; contains 1 input, 3 states and 1 output

v = vector containing parameters:

```
r1=v(1) , , , r3=v(3) , c1=v(4) , , , c(3)=v(6)
```

```

function [a,b,c,d] = bprob(v)
%BPROB    [a,b,c,d] = bprob(v)

r1=v(1);r2=v(2);r3=v(3);c1=v(4);c2=v(5);c3=v(6);

a = [(-r1+r2)/(r1+r2)    1/r2    1/r1+1/r2
      -1/(r3*c1)    r3/(r2+r3)    0
      0    1    -(r3*c2)/(c3-c2)];

b = [1; 10; 1];
c = [1 0 0];
d = [0];

```

CLMBCMAT

Command Line

[ac,bc,cc,ccu,dc] = clmbcmat(a,b,c,d,g,h)
 or [ac,bc,cc,ccu] = clmbcmat(a,b,c,g,h)

Description

creates the closed loop matrices for a plant which has a MBC compensator added to the forward path

the system is describes by the following equations where z represents the states of the MBC and x the states of the plant:

$$\begin{aligned} \dot{x} &= ax - bgz, \quad \dot{z} = hc x + a-bg-hc-hdgz - hr \\ y &= cx - dgz, \quad u = -gz \end{aligned}$$

a,b,c,d = open loop plant matrices

ac,bc,cc,dc = closed loop < MBC + Plant > matrices

ac,bc,ccu,dc = closed loop < MBC + Plant > matrices
 (which have the control efforts as outputs)

g = MBC gain matrix

h = target loop, kalman filter design gain matrix

```

function [ac,bc,cc,ccu,dc] = clmbcmat(a,b,c,d,g,h)
%CLMBCMAT [ac,bc,cc,ccu,dc] = clmbcmat(a,b,c,d,g,h)
%          or [ac,bc,cc,ccu] = clmbcmat(a,b,c,g,h)

[n,m] = size(b);

ac = [a -b*g;h*c a-b*g-h*c-h*d*g];
zm1 = zeros(n,m);
bc = [zm1;-h];
zm2 = zeros(m,n);
cc = [c -d*g];
ccu = [zm2 -g];
narg = nargin;
if ( narg == 6 )
    dc = d;
end

```

CLYFMATR

Command Line

```
[ac,bc,cc,dc] = clyfmatr(a,b,c,d)
or  [ac,bc,cc] = clyfmatr(a,b,c)
```

Description

creates closed loop matrices from open loop matrices

utilizes a feedback loop which feeds back the outputs
(y or unity feedback) to be compared with the reference
inputs r as seen by:

$$u = r - y = r - Cx$$

a,b,c,d = open loop plant matrices
ac,bc,cc,dc = closed loop matrices

```
function [ac,bc,cc,dc] = clyfmatr(a,b,c,d)
%CLYFMATR [ac,bc,cc,dc] = clyfmatr(a,b,c,d)
%      or [ac,bc,cc] = clyfmatr(a,b,c)

[n,m] = size(b);

ac = a - b*c;
bc = b;
cc = c;
narg = nargin;
if ( narg == 4 )
    dc = zeros(m);
end
```

COMPGN

Command Line

```

      [g,svolk,svsek,svclk] = compgn(a,b,c,d,h,w,pl)
or   [g,svolk,svsek,svclk] = compgn(a,b,c,d,h,w)

```

Description

computes the model-based compensator (MBC) gain, G , using the lqr algorithm and the expression:

$$q = c' * c$$

g = MBC gain

h = target loop, kalman filter gain

$svolk$ = singular values for $K(s) = G * \text{inv}(sI - A + BG + HC) * H$

$svsek$ = singular values for $S(s) = \text{inv}(I + K)$

$svclk$ = singular values for $C(s) = \text{inv}(I + K) * K$

w = frequency range, p = scalar row (entered at run time)

$pl = 1 ==>$ plots sent to screen (default if pl not included in command line)

$pl = 0 ==>$ plots not sent to screen

```

function [g,svolk,svsek,svclk] = compgn(a,b,c,d,h,w,pl)
%COMPGN    [g,svolk,svsek,svclk] = compgn(a,b,c,d,h,w,pl)
%          or [g,svolk,svsek,svclk] = compgn(a,b,c,d,h,w)

narg = nargin;
if (narg == 6 )
    pl = 1;
end

q = c'*c;

if ( any(eig(q) < 0) )
    disp('q < MBC > is not a positive semi-definite
matrix')
    disp(' a revised q must be calculated to make it so'),
    newq = notsemid(q);
    q = newq;
end

r = eye(min(size(c)));

p = input(' Enter a value for row, p = '),
if ( p < 0 )
    error(' Row, p, must be a positive scalar ')
end

disp('*****')
disp(' BE PATIENT-- IT WILL TAKE A FEW SECONDS TO COMPUTE ')

disp(' G AND THE COMPENSATOR SINGULAR VALUE PLOTS ')
disp('*****')

g = lqr(a,b,q,p*r);

ak = a-b*g-h*c;
bk = h;
ck = g;
dk = zeros(min(size(c)));

[svolk,svsek,svclk] = svplots(ak,bk,ck,dk,w,pl);
^Z

```

DEVPAR1

Command Line

```
[maxeig,dev] = devpar1(tol,par,v,slsys,slocl,np)
```

Description

this function plots the maximum system eigenvalues versus the percent deviation of the parameter specified

tol = tolerance of the parameter deviation

par = # of desired parameter to be deviated, v(par)

v = vector of nominal parameter values

ie. = [r1;r2;...;c2;c3]

slsys = number of system to be called in RETSS

slocl = 0 ==> select open loop option

 = 1 ==> select closed loop option

np = # of desired deviations

```

function [maxeig,dev] = devpar1(tol,par,v,slsys,sloc1,np)
%DEVPAR1 [maxeig,dev] = devpar1(tol,par,v,slsys,sloc1,np)

lv = length(v);
vcof = zeros(lv,1);

if ( tol > .99 )
    toll = -.99;
else
    toll = -tol;
end

for i=1:np+1
    dev(i) = toll + (i-1)*(tol - toll)/np;
    vcof(par) = dev(i);
    [ax,bx,cx] = errvcof(vcof,v,slsys);
    if ( sloc1 == 1 )
        eigval = eig(ax-bx*cx);
    elseif ( sloc1 == 0 )
        eigval = eig(ax);
    else
        error('select <1> closed or <0> open!')
    end
    maxeig(i) = max(real(eigval));
end

plot(dev,maxeig,'*')
xlabel('% deviation');ylabel('max eigenvalue');
title('1 PARAMETER DEV:  MAX EIGENVALUE VS. % DEVIATION')

```

DEVPAR2

Command Line

```
[phplane,dev1] = devpar2(tol,pars,v,slsys,slocl,np)
```

Description

this function plots a stability region plot for deviations in two specified parameters

after execution of the program a plot is obtained by:

```
axis([-tol(1) tol(1) -tol(2) tol(2)])
plot(dev1,phplane,'*')
axis    note: this done to auto-range axis again
```

```
tol = vector of tolerances in parameter deviations
      = [tol(1);tol(2)]
```

```
pars = vector of selected parameters
      = [pars(1);pars(2)]
```

```
v = vector of nominal parameter values
    ie. = [r1;r2;...;c2;c3]
```

```
slsys = number of system to be called in RETSS
```

```
slocl = 0 ==> select open loop option
```

```
        = 1 ==> select closed loop option
```

```
np = vector of # of deviations for each parameter
     = [np(1);np(2)]
```

```

function [phplane,dev1] = devpar2(tol,pars,v,slsys,slocl,np)
%DEVPAR2 [phplane,dev1] = devpar2(tol,pars,v,slsys,slocl,np)

lv = length(v);
vcof = zeros(lv,1);

if ( tol(1) > .99 )
    t111 = -.99;
else
    t111 = -tol(1);
end
if ( tol(2) > .99 )
    t121 = -.99;
else
    t121 = -tol(2);
end
clg
axis([t111 tol(1) t121 tol(2)])
hold on
np = np - 1;
for i=1:np(1)+1
    dev1(i) = t111 + (tol(1) - t111)*(i-1)/np(1);
    vcof(pars(1)) = dev1(i);
    for k=1:np(2)+1
        dev2(k) = t121 + (tol(2) - t121)*(k-1)/np(2);
        vcof(pars(2)) = dev2(k);
        [ax,bx,cx] = errvcof(vcof,v,slsys);
        if ( slocl == 1 )
            eigval = eig(ax-bx*cx);
        elseif ( slocl == 0 )
            eigval = eig(ax);
        else
            error('select <1> closed or <0> open!')
        end
        meigval = max(real(eigval));
        if ( meigval >= 0.0 )
            plot(dev1(i),dev2(k),'*')
            phplane(i,k) = dev2(k);
        else
            phplane(i,k) = 2*t121;
        end
    end
end
end
hold off
axis('normal')
grid

```

DEVPAR3

Command Line

```
[zu,zl] = devpar3(tol,pars,v,slsys,slocl,np)
```

Description

this function plots an upper and lower stability region plot for deviations in three parameters

pars(1) and pars(2) are represented on the x and y axis while pars(3) is on the z axis; therefore the command, after execution of the function:

```
mesh(zu)
```

will give the allowable deviations in pars(1), pars(2) and positive deviations of pars(3), while the command:

```
mesh(zl)
```

will give the allowable deviations in pars(1), pars(2) and negative deviations of pars(3)

```
tol = vector of tolerances in parameter deviations
      = [tol(1);tol(2);tol(3)]
```

```
pars = vector of selected parameters
      = [pars(1);pars(2);pars(3)]
```

```
v = vector of nominal parameter values
    ie. = [r1;r2;...;c2;c3]
```

```
slsys = number of system to be used in RETSS
```

```
slocl = 0 ==> select open loop option
```

```
        = 1 ==> select closed loop option
```

```
np = vector of # of deviations for each parameter
     = [np(1);np(2);np(3)]
```

```

function [zu,zl] = devpar3(tol,pars,v,slsys,sloc1,np)
%DEVPAR3 [zu,zl] = devpar3(tol,pars,v,slsys,sloc1,np)

lv = length(v);
vcof = zeros(lv,1);

if ( tol(1) > .99 )
    tl1l = -.99;
else
    tl1l = -tol(1);
end
if ( tol(2) > .99 )
    tl2l = -.99;
else
    tl2l = -tol(2);
end
if ( tol(3) > .99 )
    tl3l = -.99;
else
    tl3l = -tol(3);
end
i1 = (tol(1) - tl1l)/np(1);
i2 = (tol(2) - tl2l)/np(2);
[dev1,dev2] = meshdom(tl1l:i1:tol(1), tl2l:i2:tol(2));
for i=1:(np(3)/2+1)
    dev3u(i) = (i-1)*tol(3)/(np(3)/2+1);
    dev3l(i) = (i-1)*tl3l/(np(3)/2+1);
end
ct1 = 0;
ct2 = 0;
ct3 = 0;
[n,m] = size(dev1);
for i=1:n
    ct1 = ct1+1
    for k=1:m
        ct2 = ct2+1
        vcof(pars(1)) = dev1(i,k);
        vcof(pars(2)) = dev2(i,k);
        zu(i,k) = 0.0;
        zl(i,k) = 0.0;
        for q=1:(np(3)/2+1)
            ct3 = ct3+1
            vcof(pars(3)) = dev3u(q);
            [ax,bx,cx] = errvcof(vcof,v,slsys);
            if ( sloc1 == 1 )
                eigval = eig(ax-bx*cx);
            elseif ( sloc1 == 0 )
                eigval = eig(ax);
            else
                error('select <1> closed or <0> open!')
            end
            meigval = max(real(eigval));
            if ( meigval >= 0.0 )

```

```

                break
            else
                zu(i,k) = dev3u(q);
            end
        end
    end
    for q=1:(np(3)/2+1)
        ct3 = ct3+1
        vcof(pars(3)) = dev3l(q);
        [ax,bx,cx] = errvcof(vcof,v,slsys);
        if ( slocl == 1 )
            eigval = eig(ax-bx*cx);
        elseif ( slocl == 0 )
            eigval = eig(ax);
        else
            error('select <1> closed or <0> open!')
        end
        meigval = max(real(eigval));
        if ( meigval >= 0.0 )
            break
        else
            zl(i,k) = dev3l(q);
        end
    end
end
end

hold on
mesh(zu)
mesh(zl)
hold off

```

DIRECT

Command Line

```
[dr] = direct(a,nsv)
```

Description

this function computes the singular value decomposition (SVD) of the matrix a and then calculates:

$$dr = u * v^H$$

where u and v^H are vectors from the SVD(a) and correspond to whatever singular value(σ) was specified by nsv

dr is used to find an error $E = -\sigma_{\min}(a)*dr$ such that when E is added to a (a+E), whatever singular value was specified in the calculation of dr will go to zero in (a+E)

nsv = which singular value you want to go to zero

ex: nsv = 1, choose minimum singular value

nsv = 2, choose next largest singular value

.....

nsv = n, choose maximum singular value

```
function [dr] = direct(a,nsv)
%DIRECT    [dr] = direct(a,nsv)

narg = nargin;
if ( narg == 1 )
    nsv = 1;
end

[u,s,v] = svd(a);

[n,m] = size(a);
if ( nsv > n )
    error(' nsv is too large, exceeds size of matrix!! ')
end
r = (n+1) - nsv;

u1 = u(1:n,r);
v1 = v(1:n,r);
dr = u1*v1';
```

DRXTOEW

Command Line

```
[ew,xlow] = drxtoew(pars,v,slsys,sloc1,npx)
```

Description

this function generates a worst case error for stability E_w that is physically meaningful

by worst case error for stability, it is meant that E_w represents the minimum error in the least likely direction that will make the minimum singular value of $(a+E_w)$ go to zero (a for open loop case; a_{cl} for closed loop case)

by physically meaningful, it is meant that E_w contains only elements that correspond to elements in the matrix a which will vary when the parameters that make up the matrix a are varied

the function DIRECT is used to find an error E_w that is not physically meaningful; elements that do not have physical meaning are then eliminated; the function FINDX is then used to find a constant x_{low} which is used to modify the E_w with elements missing to again be a worst case error for stability for the matrix a (this is the E_w outputted by the function)

npx = number of points used to determine x_{low} in the function FINDX

$npx = 500$ if npx not included in input arguments

$sloc1 = 0 \implies$ select open loop option

$sloc1 = 1 \implies$ select closed loop option

$slsys$ = number of system to be evaluated as required by the function RETSS

v = vector containing nominal parameter values required by the function RETSS

$pars$ = vector of parameters that will change in system

```

function [ew,xlow] = drxtoew(pars,v,slsys,sloc1,npx)
%DRXTOEW [ew,xlow] = drxtoew(pars,v,slsys,sloc1,npx)

lpars = length(pars);
lv = length(v);
[a,b,c] = retss(v,slsys);
for i=1:lpars
    v(pars(i)) = v(pars(i)) + .50*v(pars(i));
end
[ax,bx,cx] = retss(v,slsys);

if ( sloc1 == 0 )
    svmina = min(svd(a));
    dr = direct(a);
    etem = ax - a;
elseif ( sloc1 == 1 )
    svmina = min(svd(a-b*c));
    dr = direct(a-b*c);
    etem = (ax-bx*cx) - (a-b*c);
else
    error('open <0> or closed <1> loop are the choices!')
end

[n,m] = size(dr);
drx = zeros(n,m);
for i=1:n
    for j=1:m
        if ( etem(i,j) ~= 0.0000 )
            drx(i,j) = dr(i,j);
        end
    end
end

if ( drx == zeros(n,m) )
    disp('An EA which will affect only the parameters')
    disp('in pars and will result in a minimum eigenvalue')

    disp('of zero cannot be calculated -- drx is a zeros')
    disp('matrix!!!')
    error('reenter with different pars')
end

narg = nargin;
if ( narg == 4 )
    npx = 500;
end

if ( sloc1 == 0 )
    xlow = findx(a,drx,npx,0);
else
    xlow = findx(a-b*c,drx,npx,0);
end
ea = -svmina*drx/xlow;
ew = ea;

```

EATOMU1

Command Line

```
[mulow,mu,pl] = eatomul(ea,a,b,c,lomu,himu,npmu)
```

Description

this function produces a plot of the eigenvalues of:
(a - hc + ea)
versus the scalar (mu) used to calculate h

where:

ea = error between nominal and actual system
closed loop a matrices

a run-time choice is given as to whether the scalar
variations are to occur when singular values are shaped
at low or high frequencies

after execution of the function, typing the command:
plot(mu,pl,'*')
will produce the plot

mulow = value of the scalar that gave the minimum
eigenvalue with the smallest absolute value

lowmu = smallest value of the scalar to be tested
himu = largest value of the scalar to be tested
npmu = number of scalar values to be tested

```

function [mulow,mu,pl] = eatomul(ea,a,b,c,lomu,himu,npmu)
%EATOMUL [mulow,mu,pl] = eatomul(ea,a,b,c,lomu,himu,npmu)

disp(' Do you wish to match the singular values of the ')
disp(' target loop at low or high frequencies? '),
flag = input(' Type 0 for low or 1 for high: ');

eiglow = 1.e10;
mulow = 1.e10;

for i=1:npmu+1
    mu(i) = lomus + (i-1)*(himu-lomu)/npmu;
    if ( flag == 0 )
        l = -c'*inv(c*inv(a)*c');
    elseif ( flag == 1 )
        l = c'/(c*c');
    else
        error('low <0> or high <1> are the only choices!')
    end

    q = l*l';
    if ( any(eig(q) < 0) )
        newq = qntsemd(q);
        q = newq;
    end
    r = mu(i)*eye(min(size(c)));
    h = lqe(a,eye(a),c,q,r);

    eigval = eig((a-h*c) + ea);
    pl(i) = max(real(eigval));
    if ( abs(pl(i)) < abs(eiglow) )
        mulow = mu(i);
        eiglow = pl(i);
    end
end

plot(mu,pl,'*')
xlabel('mu value');ylabel('max cl eigenvalue')
title('MAX CL LOOP EIGENVALUE VS. MU VALUE - USING EA')

```

EWTODVP

Command Line

```
[ewr,parsr] = ewtodvp(ew,xew,pars,v,sloc1,slsys)
```

Description

this function produces a vector of system parameters (reduced from the original vector pars) that have significant impacts on system stability given the values of the tolerances entered at run-time

the tolerances entered at run-time are:

- * tlear (see tol in REDUCEA)
- * tlpars which is a tolerance used to compare the change in the maximum real eigenvalue of the system when a parameter is deviated

see the function DRXTOEW for an explanation of all the command line input arguments (xew is xlow)

the basic procedure used in this function is:

1. enter with worst error from DRXTOEW
2. enter with a list of nominal system parameters of interest- this is pars
3. reduce that error using REDUCEA (get error1)
4. deviate a system parameter
5. compute new worst, reduced error using deviated parameter (get error2)
6. compare maximum real eigenvalues of error1 and error2
7. if difference > tlea then parsr = pars, quit
8. if difference < tlea then parsr = pars minus the deviated parameter; using parsr instead of pars, return to step (1)

ewr = last worst, reduced error used to test parameters

```

function [ewr,parsr] = ewtodvp(ew,xew,pars,v,sloc1,slsys)
%EWTODVP [ewr,parsr] = ewtodvp(ew,xew,pars,v,sloc1,slsys)

ea = ew;xea = xew;
[a,b,c] = retss(v,slsys);
ac = a - b*c;

disp('The maximum real eigenvalue of a + ea = '),
if ( sloc1 == 1 )
    eigea = max(real(eig(ac+ea)))
elseif ( sloc1 == 0 )
    eigea = max(real(eig(a+ea)))
else
    error('choices are <1> closed or <0> open loop!')
end

tlear = input('Enter the tolerance used to reduce ea, tlear
= ');
tlpar = input('Enter the tolerance used to reduce pars,
tlpar = ');

while 1
    if ( sloc1 == 1 )
        ear = reducea(ea,ac,tlear);
        eigea = max(real(eig(ac+ear)));
    else
        ear = reducea(ea,a,tlear);
        eigea = max(real(eig(a+ear)));
    end

    lpars = length(pars);
    lv = length(v);
    [e,f] = size(ear);
    parst = [];
    count1 = 0;
    for w=1:lpars
        vx1 = v;vx2 = v;
        vx1(pars(w)) = v(pars(w)) + .50*v(pars(w));
        for z=1:lpars
            vx2(pars(z)) = v(pars(z)) + .50*v(pars(z));
        end
        [ax1,bx1,cx1] = retss(vx1,slsys);
        [ax2,bx2,cx2] = retss(vx2,slsys);

        if ( sloc1 == 0 )
            svminax = min(svd(ax1));
            dr = direct(ax1);
            etem = ax2 - a;
        else
            svminax = min(svd(ax1-bx1*cx1));
            dr = direct(ax1-bx1*cx1);
            etem = (ax2-bx2*cx2) - (a-b*c);
        end
    end
end

```

```

[n,m] = size(dr);
drx = zeros(n,m);

for i=1:n
    for j=1:m
        if ( etem(i,j) ~= 0.0000 )
            drx(i,j) = dr(i,j);
        end
    end
end

if ( drx == zeros(n,m) )
    disp('An EA which will affect only the
parameters')
    disp('in pars and will result in a minimum
eigenvalue')
    disp('of zero cannot be calculated -- drx is
a zeros')
    disp('matrix!!!')
    error('reenter with different pars')
end
eax = -svminax*drx/xea;
earx = reducea(eax,ac,tlear);
if ( slocl == 1 )
    eigearx = max(real(eig(ac + earx)));
else
    eigearx = max(real(eig(a + earx)));
end
test = abs(eigearx - eigear);
flag = 0;
for k=1:e
    for p=1:f
        if(ear(k,p)~=0.0 & test>tlpar)
            flag = 1;
            count1 = count1+1;
            parst(count1) = pars(w);
            break
        end
    end
    if ( flag == 1 )
        break
    end
end
end

lparst = length(parst);
if ( lparst == lpars )
    break
else
    pars = parst;
end
end
parsr = pars;
ewr = ear;

```

FINDX

Command Line

```

      [xlow,svminlow,x,svmin] = findx(a,drx,np,pl)
or   [xlow,svminlow,x,svmin] = findx(a,drx,np)

```

Description

provides plot of the minimum singular value ($\sigma_{\min}$) of:
 $a - \sigma_{\min}(a) * drx / x$
 versus x

where:

x is some constant, whose range is specified
 by $\|drx\|_2^2$ +or- .25

drx is $u_{\min} * v_{\min}^H$ (from SVD of a) with
 elements eliminated that do not have physical
 meaning (do not correspond to elements of a
 that vary when parameters are varied)

the plot can be produced with the command:

```

      plot(x,svmin,'*')
after execution of this function

```

from this plot the x value, x_{low} , that corresponds to
 the smallest $\sigma_{\min}$ of the expression above, $svmin_{\text{low}}$, is
 found

the function DRXTOEW uses this x_{low} to calculate an E_w
 which, when added to a , makes $(a + E_w)$ on the border of
 instability

$pl = 1 \Rightarrow$ plot sent to screen (default if pl not
 included in input arguments)

$pl = \text{other} \Rightarrow$ plot not sent to screen

np = number of points used to construct the plot

```

function [xlow,svminlow,x,svmin] = findx(a,drx,np,pl)
%FINDX [xlow,svminlow,x,svmin] = findx(a,drx,np,pl)

narg = nargin;
if ( narg == 3 )
    pl = 1;
end

svmina = min(svd(a));

nmdrx = norm(drx);
nmsqrd = nmdrx^2;
lo = nmsqrd - .25;
if ( lo < 0.0 )
    lo = 0.0001;
end
hi = nmsqrd + .25;

xlow = nmsqrd;
svminlow = min(svd(a - svmina*drx/xlow));

for i=1:np+1
    x(i) = lo + (i-1)*(hi-lo)/np;
    svmin(i) = min(svd(a-svmina*drx/x(i)));
    if ( svmin(i) < svminlow )
        xlow = x(i);
        svminlow = svmin(i);
    end
end

if ( pl == 1 )
    plot(x,svmin);xlabel('x value');ylabel('min sv');
    grid;title('MIN SV VS. X: SVD(A-SVMINA*DRX/X')
end

```

MBCDES

Command Line

```
[g,h,svolf,svsef,svclf,svolh,svseh,svclh] =
                                     mbcdes(a,b,c,d,w,pl)
or
[g,h,svolf,svsef,svclf,svolh,svseh,svclh] =
                                     mbcdes(a,b,c,d,w)
```

Description

design an MBC compensator for a plant with matrices A,B,C & D (plant not augmented with integrators at the input channels)

a target loop with transfer function:

$$G_kf(s) = C \cdot \text{inv}(sI - A) \cdot H$$

is designed first- this is the desired (MBC + Plant)

svolh, svseh, and svclh are the singular value plots for the target loop open, sensitivity, and closed loop plots (see TARGHI AND TARGLOW)

the MBC gain G is found next- this is used to 'recover' the target loop when the MBC is added to the plant; the MBC has the transfer function:

$$K(s) = G \cdot \text{inv}(sI - A + B \cdot G + H \cdot C) \cdot H$$

the plant has the transfer function:

$$G_p(s) = C \cdot \text{inv}(sI - A) \cdot B$$

you will be asked if singular values should be matched at low or high frequencies in the target loop-- this will affect the calculation of H

you will also be asked for values of the scalars:

- * mu, u; used in target loop design (H matrix)
- * row, p; used in target loop recovery (G matrix)

g = MBC gain matrix

h = target loop, kalman filter gain matrix

svolf = singular values for $G(s) = G_p(s) \cdot K(s)$

svsef = singular values for $S(s) = \text{inv}(I + G(s))$

svclf = singular values for $C(s)=\text{inv}(I+G(s))*G(s)$

a,b,c,d = plant matrices, w = frequency range

pl = 1 ==> plots sent to screen (default if
pl not included in command line)

pl = 0 ==> plots not sent to screen

```

function [g,h,svolf,svsef,svclf,svolh,svseh,svclh] =
mbcdes(a,b,c,d,w,pl)
%MBCDES [g,h,svolf,svsef,svclf,svolh,svseh,svclh] =
mbcdes(a,b,c,d,w,pl)
%      or [g,h,svolf,svsef,svclf,svolh,svseh,svclh] =
mbcdes(a,b,c,d,w)

narg = nargin;
if ( narg == 5 )
    pl = 1;
end

disp(' Do you wish to match the singular values of the ')
disp(' target loop at high or low frequencies? '),
flag = input(' Type 0 for low or 1 for high: ');

if ( flag == 0 )
    [h,svolh,svseh,svclh] = targlow(a,b,c,d,w,pl);
elseif ( flag == 1 )
    [h,svolh,svseh,svclh] = targhi(a,b,c,d,w,pl);
else
    error(' low <0> or high <1> are the only choices!! ')
end

g = compgn(a,b,c,d,h,w,pl);

ak = a - b*g - h*c;
bk = h;
ck = g;
dk = d;

[af,bf,cf,df] = series(ak,bk,ck,dk,a,b,c,d);

disp('*****')
disp(' BE PATIENT-- IT WILL TAKE A FEW SECONDS TO COMPUTE ')

disp(' THE < MBC + PLANT > SINGULAR VALUE PLOTS ')
disp('*****')

[svolf,svsef,svclf] = svplots(af,bf,cf,df,w,pl);

```

MBCDESA

Command Line

```
[ga,ha,svolf,svsef,svclf,svolh,svseh,svclh] =
                                mbcdesa(a,b,c,d,w,pl)
or
[ga,ha,svolf,svsef,svclf,svolh,svseh,svclh] =
                                mbcdesa(a,b,c,d,w)
```

Description

design an MBC compensator for a plant with matrices A,B,C & D -- plant is augmented with integrators at the input channels to produce a zero steady state error

a target loop with transfer function:

$$G_kf(s) = CA \cdot \text{inv}(sI - AA) \cdot HA$$

is designed first- this is the desired (MBC + Plant)

the MBC gain GA is found next- this is used to 'recover' the desired target loop when the MBC is added to the plant; the MBC transfer function is:

$$K(s) = GA \cdot \text{inv}(sI - AA + BA \cdot GA + HA \cdot CA) \cdot HA$$

the augmented plant has the transfer function:

$$G_p(s) = CA \cdot \text{inv}(sI - AA) \cdot BA$$

the singular values of the target loop will automatically be matched at both high and low frequencies due to the extra degrees of freedom provided by the integrators

you will be asked for values of the scalars:

- * mu, u; used in target loop design (H matrix)
- * row, p; used in target loop recovery (G matrix)

ha = target loop, kalman filter gain
for augmented plant

ga = MBC gain matrix for augmented plant

svolf = singular values for $G(s) = G_p(s) \cdot K(s)$
svsef = singular values for $S(s) = \text{inv}(I + G(s))$
svclf = singular values for $C(s) = \text{inv}(I + G(s)) \cdot G(s)$

```
a,b,c,d = plant matrices
aa,ba,ca,da = augmented plant matrices
              (integrators added at each input channel)
w = frequency range

pl = 1 ==> plots sent to screen (default if
              pl not included in command line)
pl = 0 ==> plots not sent to screen
```

```

function [ga,ha,svolf,svsef,svclf,svolh,svseh,svclh] =
mbcdesa(a,b,c,d,w,pl)
%MBCDESA [ga,ha,svolf,svsef,svclf,svolh,svseh,svclh] =
mbcdesa(a,b,c,d,w,pl)
%      or [ga,ha,svolf,svsef,svclf,svolh,svseh,svclh] =
mbcdesa(a,b,c,d,w)

narg = nargin;
if ( narg == 5 )
    pl = 1;
end

[ha,svolh,svseh,svclh] = targhla(a,b,c,d,w,pl);

[aa,ba,ca,da] = augint(a,b,c,d);

ga = compgn(aa,ba,ca,da,ha,w,pl);

aka = aa - ba*ga - ha*ca;
bka = ha;
cka = ga;
dka = da;

[afa,bfa,cfa,dfa] = series(aka,bka,cka,dka,aa,ba,ca,da);

disp('*****')
disp(' BE PATIENT-- IT WILL TAKE A FEW SECONDS TO COMPUTE ')

disp(' THE < MBC + PLANT > SINGULAR VALUE PLOTS ')
disp('*****')

[svolf,svsef,svclf] = svplots(afa,bfa,cfa,dfa,w,pl);

```

MBCMTRIX

Command Line

```
[ak,bk,ck,dk,af,bf,cf,df] = mbcatrix(a,b,c,d,g,h)
```

Description

returns the MBC compensator matrices and the
< MBC + Plant > matrices given:

g = MBC gain matrix used in target loop
recovery

h = target (or desired) loop, kalman filter
gain matrix

ak,bk,ck,dk = MBC compensator matrices

af,bf,cf,df = < MBC + Plant > matrices

```
function [ak,bk,ck,dk,af,bf,cf,df] = mbcatrix(a,b,c,d,g,h)
%MBCMTRIX [ak,bk,ck,dk,af,bf,cf,df] = mbcatrix(a,b,c,d,g,h)
```

```
ak = a - b*g - h*c;
```

```
bk = -h;
```

```
ck = -g;
```

```
dk = d;
```

```
[af,bf,cf,df] = series(ak,bk,ck,dk,a,b,c,d);
```

MBCSTEPS

Command Line

```

[yfc,yfuc,yhc] = mbcsteps(a,b,c,d,t,iu,h,g)
or  [yfc,yfuc,yhc] = mbcsteps(a,b,c,d,t,iu,h)
or  [yfc,yfuc,yhc] = mbcsteps(a,b,c,d,t,iu)

```

Description

gives the closed loop < MBC + Plant > output and control effort and closed loop < Target Loop > output step responses for a specified input

this function will either evaluate these step responses for a specified h and/or g (and thus a specified MBC to be added to the plant) or allow for the design and computation of an MBC to be added to the plant; once the MBC is found it is combined with the plant and step responses evaluated

in either case the step responses of the target loop, yhc, are included for the h either provided or found within the function

the functions QMBCDES, QMBCDESA, AUGINT, and QCOMPNG are used to form the target loop and/or MBC according to what information is included in the command line if both h and g are not provided

the function CLYFMATR is used to form the closed loop target loop matrices to which step responses are applied to get yhc

the function CLMBCMAT is used to form the closed loop MBC matrices, when both g and h have been either found or inputted, to which step responses are applied to get the <MBC + Plant> output response, yfc, and the MBC control effort response, yfuc.

```

function [yfc,yfuc,yhc] = mbcsteps(a,b,c,d,t,iu,h,g)
%MBCSTEPS [yfc,yfuc,yhc] = mbcsteps(a,b,c,d,t,iu,h,g)
%      or [yfc,yfuc,yhc] = mbcsteps(a,b,c,d,t,iu)

narg = nargin;
if ( narg < 8 )
    if ( narg == 6 )
        disp(' Do you want the plant to be augmented with
inte- ')
        disp(' grators at the plant input channels- choose
no if ')
        disp(' you are entering an h- ? '),
        flag = input(' Type 0 for no and 1 for yes: ');
        disp(' '),
        if ( flag == 0 )
            [g,h] = qmbcdes(a,b,c,d);
        elseif ( flag == 1 )
            [g,h] = qmbcdesa(a,b,c,d);
            [a,b,c,d] = augint(a,b,c,d);
        else
            error('no <0> and yes <1> are the only
choices!')
        end
        elseif( narg == 7 )
            g = qcompn(a,b,c,d,h);
        else
            error(' incorrect number of input arguments! ')
        end
    end

[n,m] = size(h);
k = eye(m);
gr = eye(m);
[ahc,bhc, chc, dhc] = clyfmatr(a,h,c,d,k,gr);

[afc,bfc,cfc,cfuc,dfc] = clmbcmat(a,b,c,d,g,h);

yfc = step(afc,bfc,cfc,dfc,iu,t);
yfuc = step(afc,bfc,cfuc,dfc,iu,t);
yhc = step(ahc,bhc, chc, dhc,iu,t);

```

NETRET

Command Line

```
[a,b,c,d] = netret(v)
```

Description

this function provides an example set of a,b,c, and d matrices which can be used to learn and test other functions provided in this package

this example is an RC network with 7 resistors (r1...r7) and 4 capacitors (c1...c4)

calculates a,b,c & d matrices for the resistor and capacitor values entered as input arguments; contains 2 inputs, 4 states and 2 outputs.

v = vector containing parameters:

```
    r1=v(1) , , , r7=v(7) , c1=v(8) , , , c(4)=v(11)
```

```

function [a,b,c,d] = netret(v)
%NETRET [a,b,c,d] = netret(v)

r1=v(1);r2=v(2);r3=v(3);r4=v(4);r5=v(5);r6=v(6);r7=v(7);
c1=v(8);c2=v(9);c3=v(10);c4=v(11);

d0 = r5*r6*r7 + r4*r6*r7 + r4*r5*r7 + r4*r5*r6;
d1 = r6*c1*d0;
d2 = r7*c2*d0;
d4 = r4*c4*d0;

% a (4x4) matrix

a(1,1) = (r4*r5*r7)/d1 - 1/c1*(1/r1 + 1/r6 + 1/r2);
a(1,2) = 1/(r2*c1) + (r4*r5*r6)/d1;
a(1,3) = 0;
a(1,4) = (r5*r6*r7)/d1;

a(2,1) = 1/(r2*c2) + (r4*r5*r7)/d2;
a(2,2) = (r4*r5*r6)/d2 - 1/c2*(1/r2 + 1/r7 + 1/r3);
a(2,3) = 1/(r3*c2);
a(2,4) = (r5*r6*r7)/d2;

a(3,1) = 0;
a(3,2) = 1/(r3*c3);
a(3,3) = -1/(r3*c3);
a(3,4) = 0;

a(4,1) = (r4*r5*r7)/d4;
a(4,2) = (r4*r5*r6)/d4;
a(4,3) = 0;
a(4,4) = (r5*r6*r7)/d4 - 1/(r4*c4);

% b (4x2) matrix

b(1,1) = 1/(r1*c1);
b(1,2) = (r4*r6*r7)/d1;

b(2,1) = 0;
b(2,2) = (r4*r6*r7)/d2;

b(3,1) = 0;
b(3,2) = 0;

b(4,1) = 0;
b(4,2) = (r4*r6*r7)/d4;

% c (2x4) matrix

c(1,1) = 0;
c(1,2) = 0;
c(1,3) = 1;
c(1,4) = 0;

```

```
c(2,1) = 0;  
c(2,2) = 0;  
c(2,3) = 0;  
c(2,4) = 1;
```

```
% d (2x2) matrix
```

```
d(1,1) = 0;  
d(1,2) = 0;
```

```
d(2,1) = 0;  
d(2,2) = 0;
```

NOTSEMID

Command Line

```
[newq] = notsemid(q)
```

Description

sometimes q will have eigenvalues that are quite small which the computer may interpret to be negative (ie. $-0.0e-15$)

if an algorithm requires a positive semi-definite matrix (like the lqe routine) then this condition will produce

an error message and an abort this algorithm adds small increments to the diagonal of q until these small, 'negative' eigenvalues are interpreted by the computer as being positive

the increments are not allowed to exceed $1.e-4$ so as not to have any real effect on the eigenvalues themselves

q = original matrix
newq = revised matrix
epps = increment added to q's diagonal

```

function [newq] = notsemid(q)
%NOTSEMID [newq] = notsemid(q)

flag = 1;

if( any(eig(q) < 0) )
    flag = 0;

    eplimit = 1.e-4;

    q,
    disp(' push any key to continue '),pause,
    disp(' the eigenvalues of q are ')
    eig(q),
    disp(' push any key to continue '),pause,

    epps = 1.e-20;

    while( any( eig( epps*eye(q) + q) < 0 ) )
        epps = epps*10;
        if ( epps > eplimit )
            error(' epps increment exceeded limit ')
        end
    end

    disp(' the increment, epps, necessary was ')
    epps,
end

if flag == 0
    newq = epps*eye(q) + q;
    newq,
end

if flag == 1
    disp(' no eigenvalues were less than zero ')
    disp(' the matrix is positive semi-definite ')
    newq = q;
end
^Z

```

QCOMPGN

Command Line

```
[g,ak,bk,ck,dk] = qcompgn(a,b,c,d,h)
```

Description

computes the model-based compensator (MBC) gain, G, using the lqr algorithm and the expression: $q = c' * c$

this "quick" design function differs from the COMPGN function in that no singular values are computed or plotted (hence less calculation time)

g = MBC gain

h = target loop, kalman filter gain

ak,bk,ck,dk = MBC matrices

a,b,c,d = plant matrices

p = scalar row (entered at run time)

```

function [g,ak,bk,ck,dk] = qcomp gn(a,b,c,d,h)
%QCOMP GN [g,ak,bk,ck,dk] = qcomp gn(a,b,c,d,h)

q = c'*c;

if ( any(eig(q) < 0) )
    disp(' q < MBC > is not a positive semi-definite matrix
')
    disp(' a revised q must be calculated to make it so'),
    newq = notsemid(q);
    q = newq;
end

r = eye(min(size(c)));

p = input(' Enter a value for row, p = '),
if ( p < 0 )
    error(' Row, p, must be a positive scalar ')
end

g = lqr(a,b,q,p*r);

ak = a-b*g-h*c;
bk = h;
ck = g;
dk = zeros(min(size(c)));

```

QMBCDES

Command Line

```
[g,h,af,bf,cf,df,ak,bk,ck,dk] = qmbcdes(a,b,c,d)
```

Description

design an mbc compensator for a plant with matrices A,B,C & D (plant not augmented with integrators at the input channels)

this "quick" design function differs from the MBCDES function in that no singular values are computed or plotted (hence less calculation time)

the plant has transfer function $G_p(s) = C \cdot \text{inv}(sI - A) \cdot B$

a target (or desired) loop with transfer function $G_kf(s) = C \cdot \text{inv}(sI - A) \cdot H$ will be designed first- this is the desired < MBC + Plant >

you will be asked if singular values should be matched at low or high frequencies in the target loop- this will affect the calculation of H

the MBC gain G is found next- this is used to "recover" the target loop when the MBC is added to the plant; the MBC has the transfer function:

$$K(s) = G \cdot \text{inv}(sI - A + B \cdot G + H \cdot C) \cdot H$$

you will be asked for values of the scalars:

- * mu, u; used in target loop design (H matrix)
- * row, p; used in target loop recovery (G matrix)

g = MBC gain matrix

h = target loop, kalman filter gain matrix

a,b,c,d = plant matrices

ak,bk,ck,dk = MBC matrices

af,bf,cf,df = < MBC + plant > matrices

```

function [g,h,af,bf,cf,df,ak,bk,ck,dk] = qmbcdes(a,b,c,d)
%QMBCDES [g,h,af,bf,cf,df,ak,bk,ck,dk] = qmbcdes(a,b,c,d)

h = qtarg(a,b,c,d);

p = input(' Enter a value for row, p = '),
if ( p < 0 )
    error(' row, p, must be a positive scalar!! ')
end

q = c'*c;

if ( any(eig(q) < 0) )
    disp(' q < MBC > is not a positive semi-definite matrix
')
    disp(' a revised q must be calculated to make it so'),
    newq = notsemid(q);
    q = newq;
end

r = p*eye(min(size(c)));
g = lqr(a,b,q,r);

[ak,bk,ck,dk,af,bf,cf,df] = mbcmtx(a,b,c,d,g,h);

```

QMBODESA

Command Line

```
[ga,ha,af,bf,cf,df,ak,bk,ck,dk]= qmbodesa(a,b,c,d)
```

Description

design an MBC compensator for a plant with matrices A,B,C & D -- plant is augmented with integrators at the input channels to produce zero steady state error

this "quick" design function differs from the MBCDESA function in that no singular values are computed or plotted (hence less calculation time)

the open loop matrices are first augmented with integrators at the input channels to create matrices AA,BA,CA & DA with the transfer function
 $G_p(s) = CA \cdot \text{inv}(sI - AA) \cdot BA$

a target (or desired) loop with transfer function
 $G_{kf}(s) = CA \cdot \text{inv}(sI - A) \cdot HA$ will be designed- this is the desired < MBC + Plant >

the MBC gain GA is found next- this is used to "recover" the target loop when the MBC is added to the plant; the MBC has the transfer function
 $K(s) = GA \cdot \text{inv}(sI - AA + BA \cdot GA + HA \cdot CA) \cdot HA$

the singular values of the target loop will automatically be matched at both low and high frequencies due to the extra degrees of freedom provided by the integrators

you will be asked for values of the scalars:
 *mu, u; used in target loop design (H matrix)
 *row, p; used in target loop recovery (G matrix)

ga = MBC gain matrix with augmented plant
 ha = target loop, kalman filter gain matrix with augmented plant

a,b,c,d = original plant matrices
 aa,ba,ca,da = augmented plant matrices

ak,bk,ck,dk = MBC matrices
 af,bf,cf,df = < MBC + Plant > matrices

```

function [ga,ha,af,bf,cf,df,ak,bk,ck,dk]= qmbcdesa(a,b,c,d)
%QMBCDESA [ga,ha,af,bf,cf,df,ak,bk,ck,dk]=qmbcdesa(a,b,c,d)

[aa,ba,ca,da] = augint(a,b,c,d);

u = input(' Enter a value for mu, u = '),
if ( u < 0 )
    error(' mu, u, must be a positive scalar!! ')
end

l1 = c*inv(a)*b;
    if ( abs(det(l1)) < eps )
        error(' c*inv(a)*b not invertable!! ')
    end
lh = c'*inv(c*c');
l = [-inv(l1);lh];

r = u*eye(min(size(ca)));
q = l*l';

if ( any(eig(q) < 0) )
    disp(' q <for MBC> not a positive semi-definite')
    disp(' matrix! A revised q must be found to make it
so. ')
    newq = notsemid(q);
    q = newq;
end

ha = lqe(aa,eye(aa),ca,q,r);

p = input(' Enter a value for row, p = '),
if ( p < 0 )
    error(' row, p, must be a positive scalar!! ')
end

q = ca'*ca;

if ( any(eig(q) < 0) )
    disp(' q <for target loop> not a positive
semi-definite')
    disp(' matrix! A revised q must be found to make it
so. ')
    newq = notsemid(q);
    q = newq;
end

r = p*eye(min(size(ca)));
ga = lqr(aa,ba,q,r);

[ak,bk,ck,dk,af,bf,cf,df] = mbcatrix(aa,ba,ca,da,ga,ha);

```

QNTSEMID

Command Line

```
[newq] = qntsemid(q)
```

Description

sometimes q will have eigenvalues that are quite small which the computer may interpret to be negative (ie. $-0.0e-15$)

if an algorithm requires a positive semi-definite matrix (like the lqe routine) then this condition will produce

an error message and an abort this algorithm adds small increments to the diagonal of q until these small, 'negative' eigenvalues are interpreted by the computer as being positive

the increments are not allowed to exceed $1.e-4$ so as not to have any real effect on the eigenvalues themselves

q = original matrix
 $newq$ = revised matrix
 $epps$ = increment added to q 's diagonal

this function differs from the NOTSEMID function in that no messages are displayed on the screen at run time

```

function [newq] = qntsemid(q)
%QNTSEMID [newq] = qntsemid(q)

flag = 1;

if( any(eig(q) < 0) )
    flag = 0;
    eplimit = 1.e-4;
    epps = 1.e-20;

    while( any( eig( epps*eye(q) + q) < 0 ) )
        epps = epps*10;
        if ( epps > eplimit )
            error(' epps increment exceeded limit ')
        end
    end
end

if flag == 0
    newq = epps*eye(q) + q;
end

if flag == 1
    disp(' no eigenvalues were less than zero ')
    disp(' the matrix is positive semi-definite ')
    newq = q;
end
^Z

```

QTARG

Command Line

[h] = qtarg(a,b,c,d)

Description

this function combines the functions TARGLOW and TARGHI in terms of calculating the h matrix for the target loop

singular values may be matched at either low or high frequency and the scalar (mu) is inputted when the function is executed

this function differs from TARGLOW or TARGHI in that no singular values are calculated and therefore it allows a much quicker calculation of h

```

function [h] = qtarg(a,b,c,d)
%QTARG [h] = qtarg(a,b,c,d)

disp(' Do you wish to match the singular values of the ')
disp(' target loop at low or high frequencies? '),
flag = input(' Type 0 for low or 1 for high: ');

if ( flag == 0 )
    l = -c'*inv(c*inv(a)*c');
elseif ( flag == 1 )
    l = c'/(c*c');
else
    error(' low <0> or high <1> are the only choices!! ')
end

q = l*l';
if ( any(eig(q) < 0) )
    disp(' q is not a positive semi-definite matrix ')
    disp(' a revised q must be calculated to make it so'),
    newq = notsemid(q);
    q = newq;
end

disp(' '),
u = input(' Enter a value for mu, u = '),
if ( u < 0 )
    error(' mu, u, must be a positive scalar!! ')
end

r = u*eye(min(size(c)));
h = lqe(a,eye(a),c,q,r);

```

QTARGA

Command Line

[ha] = qtarga(a,b,c,d)

Description

this function is similar to the function TARGHLA in terms of calculating the h matrix for the target loop

singular values may be matched at either low or high frequency and the scalar (μ) is inputted when the function is executed

this function differs from TARGHLA in that no singular values are calculated and therefore it allows a much quicker calculation of h

```

function [ha] = qtarga(a,b,c,d)
%QTARGA [ha] = qtarga(a,b,c,d)

u = input(' Enter a value for mu, u = ')
if ( u < 0 )
    error(' mu, u, must be a positive scalar ')
end

[aa,ba,ca,da] = augint(a,b,c,d);

ll = c*inv(a)*b;
lh = c'*inv(c*c');
l = [-inv(ll);lh];

q = l*l';
if ( any(eig(q) < 0) )
    disp(' q is not a positive semi-definite matrix ')
    disp(' a revised q must be calculated to make it so'),
    newq = notsemid(q);
    q = newq;
end

r = u*eye(min(size(ca)));
ha = lqe(aa,eye(aa),ca,q,r);

```

REDUCEA

Command Line

[ear] = reducea(ea,a,tol)

Description

this function is used to reduce a matrix ea that is added to a matrix a such that the reduced ea, ear, when added to a, (a+ear), will to some tolerance tol have the same maximum real eigenvalue as (a+ea)

the routine used is:

1. find the element of ea with minimum absolute value
2. eliminate that element; result is ear
3. compare maximum real eigenvalues of (a+ea) and (a+ear)
4. if difference is greater than tol, the process is complete; ear = ea
5. if difference is less than tol, ea = ear, go to step (1)

this function is called from the function EWTODVP which is used to find which parameters in a system have significant impact on stability

```

function [ear] = reducea(ea,a,tol)
%REDUCEA [ear] = reducea(ea,a,tol)

eigea = max(real(eig(a+ea)));
ear = ea;
narg = nargin;
if ( narg == 2 )
    if ( abs(eigea) > .0001 )
        tol = abs(.001 - eigea);
    else
        tol = .0005;
    end
end
[n,m] = size(ear);
limit = n*m;
count1 = 1;
while 1
    if ( count1 > limit )
        break
    end
    temp1 = ear(:);
    lt1 = length(temp1);
    count2 = 0;
    temp2 = [];
    for i=1:lt1
        if ( temp1(i) ~= 0.0 )
            count2 = count2 + 1;
            temp2(count2) = temp1(i);
        end
    end
    [minelem,index] = min(abs(temp2));
    for i=1:n
        for j=1:m
            if ( ear(i,j) == temp2(index) )
                row = i;
                col = j;
            end
        end
    end
    ear(row,col) = 0.0;
    eigea = max(real(eig(a+ear)));
    test = abs(eigea - eigea);
    if ( test > tol )
        ear(row,col) = minelem;
        break
    end
    count1 = count1+1;
end

```

RETSS

Command Line

```
[a,b,c,d] = retss(v,slsys)
```

Description

this function allows a particular state space representation of a system comprised of a,b,c and d matrices to be chosen from all the systems the user may have created

three example state space representations of RC networks that have been provided are currently referenced from this function

to add networks that the user creates, elseif statements identical to those currently seen in the function must be added

for example, if another network (now the 4th) was to be added then the statements:

```
elseif ( slsys == 4 )  
    [a,b,c,d] = filename(v);
```

would have to be added to the function file

a,b,c,d = matrices for state space representation
of the desired system

v = vector of nominal parameter values (ie. r1,r2,...)

```
slsys = 1 ==> select netret.m  
       = 2 ==> select aprob.m  
       = 3 ==> select bprob.m
```

```
function [a,b,c,d] = retss(v,slsys)
%RETSS    [a,b,c,d] = retss(v,slsys)

lv = length(v);

if ( slsys == 1 )
    [a,b,c,d] = netret(v);
elseif ( slsys == 2 )
    [a,b,c,d] = aprob(v);
elseif ( slsys == 3 )
    [a,b,c,d] = bprob(v);
elseif ( slsys == 4 )
    [a,b,c,d] = cprob(v);
else
    error('IMPROPER CHOICE OF SYSTEM!')
    disp('If you wish to add another choice the file')
    disp('retss.m much be edited to include it.')
end
```

SVCLINV

Command Line

[svcli] = svclinv(a,b,c,d,w)

Description

calculates and plots singular values for the inverse of the closed loop transfer function matrix given the entered matrices and frequency range, w

$$G(s) = C*INV(sI-A)*B+D, \quad INV(C) = (I+INV(G))$$

```
function [svcli] = svclinv(a,b,c,d,w)
%SVCLINV [svcli] = svclinv(a,b,c,d,w)

ii = eye(min(size(c)));
iii = eye(a);

j = sqrt(-1);

nw = length(w);

for i=1:nw

    g = c*inv(j*w(i)*iii-a)*b+d;
    cli = ii + inv(g);
    svwcli(:,i) = svd(cli);
    svcli(:,i) = 20*log10(svwcli(:,i));

end
```

SVEATOMU

Command Line

```
[pl,mu] = sveatomu(svea,a,b,c,w,lomu,himu,npmu)
```

Description

this function is used to determine what values of the scalar (μ) will cause the plots $\max(svea)$ and $\sigma_{\min}(sI - a + hc)$ to cross in the target loop; if they cross they violate an inequality which requires them not to cross to insure stability robustness for the closed loop system

$\sigma_{\min}(sI - a + hc)$ are the minimum singular values of sI minus the closed loop a matrix of the target loop and are found within this function using SVPHI with a, h , and c as input arguments

$svea$ are the inputted singular values of the error between the actual and nominal plant closed loop a matrices, determined at low frequency; therefore $\max(svea)$ are the maximum singular values of this error

a run-time choice is given as to whether the scalar variations are to occur when singular values are shaped at low or high frequencies

after executing the function, the command:

```
plot(mu,pl,'*')
```

will produce a plot showing for which values of the scalar the plots crossed and for which they did not

the design parameter in the target loop, the h matrix, can then be found; using a scalar value that did not cause the plots to cross, the function QTARG is entered and an h calculated which preserves good stability/robustness in the target loop

```
lowmu = smallest scalar to be tested
himu  = largest scalar to be tested
npmu  = number of points to be tested
```

```

function [pl,mu] = sveatomu(svea,a,b,c,w,lomu,himu,npmu)
%SVEATOMU [pl,mu] = sveatomu(svea,a,b,c,w,lomu,himu,npmu)

disp(' Do you wish to match the singular values of the ')
disp(' target loop at low or high frequencies? '),
flag = input(' Type 0 for low or 1 for high: ');

nw = length(w);
for i=1:npmu+1
    mu(i) = lommu + (i-1)*(himu-lomu)/npmu;
    if ( flag == 0 )
        l = -c'*inv(c*inv(a)*c');
    elseif ( flag == 1 )
        l = c'/(c*c');
    else
        error('low <0> or high <1> are the only choices!')
    end

    q = l*l';
    if ( any(eig(q) < 0) )
        newq = qntsemid(q);
        q = newq;
    end
    r = mu(i)*eye(min(size(c)));
    h = lqe(a,eye(a),c,q,r);

    svphic1 = svphi(a,h,c,l,w);

    fl = 0;
    for k = 1:nw
        minsvphic1 = min(svphic1(:,k));
        maxsvea = max(svea(:,k));
        if ( maxsvea >= minsvphic1 )
            fl = 1;
            break
        end
    end

    if ( fl == 1 )
        pl(i) = 1.;
    else
        pl(i) = 0.;
    end
end

end

plot(mu,pl,'*')
xlabel('mu value');ylabel('0 or 1')
title('<1>: PLOTS CROSSED OR <0>: DID NOT CROSS VS MU')

```

SVPHI

Command Line

```
[svphiocl] = svphi(a,b,c,selocl,w)
```

Description

calculates the singular values of:

```
sl - a + b*c   if slocl = 1  
sl - a         if slocl = 0
```

over the frequency range w

```
function [svphiocl] = svphi(a,b,c,selocl,w)
%SVPHI    [svphiocl] = svphi(a,b,c,selocl,w)

iii = eye(a);
j = sqrt(-1);
nw = length(w);

for i = 1:nw

    if ( selocl == 0 )
        phiocl = j*w(i)*iii - a;
    elseif ( selocl == 1 )
        phiocl = j*w(i)*iii - a + b*c;
    else
        error('choices are <1> closed or <0> open loop')
    end

    svwphiocl(:,i) = svd(phiocl);
    svphiocl(:,i) = 20*log10(svwphiocl(:,i));

end
```

SVPLOTS

Command Line

```
[svol,svse,svcl] = svplots(a,b,c,d,w,pl)
or [svol,svse,svcl] = svplots(a,b,c,d,w)
```

Description

calculates and plots the singular values for the open-loop TFM, sensitivity TFM, and closed-loop TFM for the entered matrices and frequency range, w

pl = 1 ==> plots sent to screen (default if pl not included in command line)

pl = 0 ==> plots not sent to screen

OL TFM:	$G(s) = C * INV(sI - A) * B + D$
SENS TFM:	$S(s) = INV(I + G(s))$
CL TFM:	$C(s) = INV(I + G(s)) * G(s)$

```

function [svol,svse,svcl] = svplots(a,b,c,d,w,pl)
%SVPLOTS [svol,svse,svcl] = svplots(a,b,c,d,w,pl)
%      or [svol,svse,svcl] = svplots(a,b,c,d,w)

narg = nargin;
if ( narg == 5 )
    pl = 1;
end

siz = min(size(c));
ii = eye(siz);
iii = eye(a);
j = sqrt(-1);
nw = length(w);

for i = 1:nw;
    tfm=c/(j*w(i)*iii-a)*b+d;
    svwol(:,i) = svd(tfm);
    svwse(:,i) = svd(ii/(ii+tfm));
    svwcl(:,i) = svd(ii/(ii+tfm)*tfm);
    svol(:,i) = 20*log10(svwol(:,i));
    svse(:,i) = 20*log10(svwse(:,i));
    svcl(:,i) = 20*log10(svwcl(:,i));
end;

if ( pl == 1 )

    semilogx(w,svol);title('SV OPEN-LOOP PLOT');
    xlabel('frequency');ylabel('db');grid;

    semilogx(w,svse);title('SV SENSITIVITY PLOT');
    xlabel('frequency');ylabel('db');grid;

    semilogx(w,svcl);title('SV CLOSED-LOOP PLOT');
    xlabel('frequency');ylabel('db');grid;

end

```

SVRETDIF

Command Line

```
[svrd] = svretdif(a,b,c,d,w,pl)
or      [svrd] = svretdif(a,b,c,d,w)
```

Description

calculates and plots singular values for the
return difference matrix given the entered matrices
and frequency range, w

pl = 1 ==> plots sent to screen (default if pl not
included in command line)

pl = 0 ==> plots not sent to screen

$G(s) = C \cdot \text{INV}(sI - A) \cdot B + D$

return difference matrix = (I + G(s))

```
function [svrd] = svretdif(a,b,c,d,w,pl)
%SVRETDIF [svrd] = svretdif(a,b,c,d,w,pl)
%      or [svrd] = svretdif(a,b,c,d,w)

narg = nargin;
if ( narg == 5 )
    pl = 1;
end

ii = eye(min(size(c)));
iii = eye(a);
j = sqrt(-1);
nw = length(w);

for i=1:nw
    g = c*inv(j*w(i)*iii-a)*b+d;
    rd = ii + g;
    svwrd(:,i) = svd(rd);
    svrd(:,i) = 20*log10(svwrd(:,i));
end
```

TARGHI

Command Line

```

[h,svolh,svseh,svclh] = targhi(a,b,c,d,w,pl)
or [h,svolh,svseh,svclh] = targhi(a,b,c,d,w)

```

Description

match singular values at high frequencies
for target loop using the lqe algorithm and the
expression: $l = c'/(c*c')$

h = target loop, kalman filter gain

$svolh$ = singular values for $Gkf(s) = C \cdot \text{inv}(sI - A) \cdot H$

$svseh$ = singular values for $S(s) = \text{inv}(I + Gkf)$

$svclh$ = singular values for $C(s) = \text{inv}(I + Gkf) \cdot Gkf$
(matched at high frequencies)

w = frequency range, u = scalar μ (entered at run
time)

$pl = 1 \Rightarrow$ plots sent to screen (default if
pl not included in command line)

$pl = 0 \Rightarrow$ plots not sent to screen

```

function [h,svolh,svseh,svclh] = targhi(a,b,c,d,w,pl)
%TARGHI [h,svolh,svseh,svclh] = targhi(a,b,c,d,w,pl)

narg = nargin;
if ( narg == 5 )
    pl = 1;
end

u = input(' Enter a value for mu, u = '),
if ( u < 0 )
    error(' mu, u, must be a positive scalar ')
end

l = c'/(c*c');
q = l*l';

if ( any(eig(q) < 0) )
    disp(' q is not a positive semi-definite matrix ')
    disp(' a revised q must be calculated to make it so'),
    newq = notsemid(q);
    q = newq;
end

r = u*eye(min(size(c)));

disp('*****')
disp(' BE PATIENT-- IT WILL TAKE A FEW SECONDS TO COMPUTE ')

disp(' H AND THE TARGET LOOP SINGULAR VALUE PLOTS ')
disp('*****')

h = lqe(a,eye(a),c,q,r);
[svolh,svseh,svclh] = svplots(a,h,c,d,w,pl);

```

TARGHLA

Command Line

```

[ha,svolh,svseh,svclh] = targhla(a,b,c,d,w,pl)
or [ha,svolh,svseh,svclh] = targhla(a,b,c,d,w)

```

Description

match singular values at both low and high frequencies for the target loop by creating the augmented matrices (integrators added at each input channel) and using the lqe algorithm and the expressions:

```

ll = -inv(c*inv(a)*b)
lh = c'*inv(c*c')
l = [ll;lh]

```

ha = target loop, kalman filter gain
for augmented plant

svolh = singular values for $G_{kf}(s) = C \cdot \text{inv}(sI - A) \cdot H$
svseh = singular values for $S(s) = \text{inv}(I + G_{kf})$
svclh = singular values for $C(s) = \text{inv}(I + G_{kf}) \cdot G_{kf}$
(matched at low and high frequencies)

aa,ba,ca,da = augmented matrices (integrators
added at each input channel)

a,b,c,d = original plant matrices
w = frequency range, u = scalar mu (entered at run
time)

pl = 1 ==> plots sent to screen (default if
pl not included in command line)
pl = 0 ==> plots not sent to screen

```

function [ha,svolh,svseh,svclh] = targhla(a,b,c,d,w pl)
%TARGHLA [ha,svolh,svseh,svclh] = targhla(a,b,c,d,w,pl)
%      or [ha,svolh,svseh,svclh] = targhla(a,b,c,d,w)

narg = nargin;
if (narg == 5)
    pl = 1;
end

u = input(' Enter a value for mu, u = ');
if ( u < 0 )
    error(' mu, u, must be a positive scalar ');
end

[aa,ba,ca,da] = augint(a,b,c,d);

l1 = c*inv(a)*b;
lh = c'*inv(c*c');
l = [-inv(l1);lh];

r = u*eye(min(size(ca)));
q = l*l';

if ( any(eig(q) < 0) )
    disp(' q is not a positive semi-definite matrix ')
    disp(' a revised q must be calculated to make it so'),
    newq = notsemid(q);
    q = newq;
end

disp('*****')
disp(' BE PATIENT-- IT WILL TAKE A FEW SECONDS TO COMPUTE ')

disp(' H AND THE TARGET LOOP SINGULAR VALUE PLOTS ')
disp('*****')

ha = lqe(aa,eye(aa),ca,q,r);
[svolh,svseh,svclh] = svplots(aa,ha,ca,da,w,pl);
^Z

```

TARGLOW

Command Line

```

[h,svolh,svseh,svclh] = targlow(a,b,c,d,w,pl)
or [h,svolh,svseh,svclh] = targlow(a,b,c,d,w)

```

Description

match singular values at low frequencies
 for target loop using the lqe algorithm and the
 expression: $l = -c' \cdot \text{inv}(c \cdot \text{inv}(a) \cdot c')$

h = target loop, kalman filter gain

$svolh$ = singular values for $Gkf(s) = C \cdot \text{inv}(sI - A) \cdot H$
 $svseh$ = singular values for $S(s) = \text{inv}(I + Gkf)$
 $svclh$ = singular values for $C(s) = \text{inv}(I + Gkf) \cdot Gkf$
 (matched at low frequencies)

w = frequency range, u = scalar μ (entered at run
 time)

$pl = 1 \Rightarrow$ plots sent to screen (default if
 pl not included in command line)
 $pl = 0 \Rightarrow$ plots not sent to screen

```

function [h,svolh,svseh,svclh] = targlow(a,b,c,d,w,pl)
%TARGLOW [h,svolh,svseh,svclh] = targlow(a,b,c,d,w,pl)
%      or [h,svolh,svseh,svclh] = targlow(a,b,c,d,w)

narg = nargin;
if ( narg == 5 )
    pl = 1;
end

u = input(' Enter a value for mu, u = '),
if ( u < 0 )
    error(' mu, u, must be a positive scalar ')
end

l = -c'*inv(c*inv(a)*c');
q = l*l';

if ( any(eig(q) < 0) )
    disp(' q is not a positive semi-definite matrix ')
    disp(' a revised q must be calculated to make it so'),
    newq = notsemid(q);
    q = newq;
end

r = u*eye(min(size(c)));

disp('*****')
disp(' BE PATIENT-- IT WILL TAKE A FEW SECONDS TO COMPUTE ')

disp(' H AND THE TARGET LOOP SINGULAR VALUE PLOTS ')
disp('*****')

h = lqe(a,eye(a),c,q,r);
[svolh,svseh,svclh] = svplots(a,h,c,d,w,pl);
^Z

```