MCR-72-282 Vol. I Final Report December 1972

AUTOMATED DESIGN AND OPTIMIZATION OF FLEXIBLE BOOSTER AUTOPILOTS VIA LINEAR PROGRAMMING

Prepared for:

George C. Marshall Space Flight Center
Huntsville, Alabama 35812

Prepared by:

Francis D. Hauser

Contract NAS8-28482
DCN 1-2-75-20071(IF)

MARTIN MARIETTA CORPORATION
P. O. Box 179
Denver, Colorado 80201

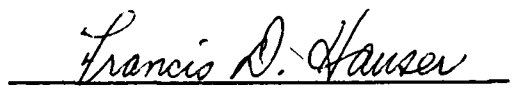
MCR-72-282 Vol. I Final Report December 1972

AUTOMATED DESIGN AND OPTIMIZATION OF FLEXIBLE BOOSTER AUTOPILOTS VIA LINEAR PROGRAMMING

Prepared for:

George C. Marshall Space Flight Center
Huntsville, Alabama 35812

Approved:



Francis D. Hauser
Author and Program Manager

Contract NAS8-28482
DCN 1-2-75-20071(IF)

MARTIN MARIETTA CORPORATION
P. O. Box 179
Denver, Colorado 80201

FOREWORD

This report, Volume I of 2 volumes, was prepared by the Guidance and Controls Section, Martin Marietta Corporation, Denver Division, under Contract NAS8-28482. It presents the historical background, the philosophy, the mathematical basis and example problems of the COEBRA program. The purpose of the contract was to convert the COEBRA program from the CDC 6400/6500 digital computer system to the UNIVAC 1108 at the George C. Marshall Space Flight Center, and to provide a manual and instruction on the use of the program. This contract was performed from March 1972 to December 1972, and was administered by the National Aeronautics and Space Administration, George C. Marshall Space Flight Center, Huntsville, Alabama, under the direction of Mr. D. K. Mowery, Dynamics and Control Division, Aeroastrodynamics Laboratory.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This report, Volume I of 2 volumes, presents the historical background, the philosophy, the mathematical basis and example problems of the COEBRA program. The following is an abstract of the COEBRA program's design algorithm.

A nonlinear programming technique has been developed by Martin Marietta Corporation, Denver Division, for the automated design and optimization of autopilots for large flexible launch vehicles. This technique, which resulted in the COEBRA program, uses the iterative application of linear programming, i.e., iterating from some starting point to the final solution by solving successive linear programming problems. The method deals directly with the three main requirements of booster autopilot design: to provide (1) good response to guidance commands; (2) response to external disturbances (e.g. wind) to minimize structural bending moment loads and trajectory dispersions; and (3) stability with specified tolerances on the vehicle and flight control system parameters.

The main design criteria are minimum gain/phase stability margins. The approach is to expand each rigid and flexible-body stability margin in a first order Taylor Series, to form a linear inequality constraint for each margin. A choice of two linear cost functions is also provided via Taylor Series expansions, in order to

(1) maximize all stability margins, and (2) minimize structural bending moment loads. A linear programming problem results, i.e., maximize margins or minimize loads, in the presence of constraints on each stability margin. The solution of this linear problem yields a new starting point for the next iteration.

Only practical control laws are considered, since this algorithm optimizes a pre-selected autopilot configuration, i.e., the number and types of feedback loops, and the number of gains and filters are pre-specified. This method finds the best values for the parameters within this feedback structure, and is able to constrain the minimum and maximum allowed values on each parameter.

The problem of handling several different flight conditions with the same time invariant control law is solved by forming a single linear programming problem in which the cost function and matrix of constraint equations is comprised of stability margins and wind responses from several flight conditions. By solving this single linear problem, several flight conditions are optimized together using a single time-invariant control law. A single control law can also be designed that is, at the same time, optimum under nominal airframe conditions and under malfunction conditions, e.g., actuator, sensor, or even engine failures.

The method is applicable to very high order systems (30th and greater per flight condition). Since it is a parameter optimization

technique, the complexity of the autopilot does not necessarily increase with an increase in the order of the fixed parts of the system.

Analog autopilots are designed in the S-plane, and digital autopilots in the W-plane. The method can design a drift minimum autopilot that meets stability margin requirements and has a maximum amount of bending moment load relief capability.

The method contains step-size optimization routines that allow convergence to local interior optima as well as to local exterior optima. The initial condition on the controller parameters can be unfeasible. Example problems are shown where the initial autopilot even yielded an unstable system. The step-size optimization routines also permit automatic self-termination of the design process.

Examples are given that demonstrate the successful application of this algorithm to the design of autopilots for both single and multiple flight conditions. The multiple flight condition problems range from a system that is 28th order at each of three flight conditions and has 23 individual autopilot parameters, to a system that is 20th order at each of three flight conditions with 28 individual autopilot parameters. The examples demonstrate the design of two types of autopilots, one where the objective is to maximize stability margins, and the other where the objective is to optimize structural bending moment load relief capability.

CHAPTER 1

INTRODUCTION

The objective that motivated the development of this nonlinear programming algorithm was to computerize an existing and flight-proven (hence practical) conventional autopilot design method for large highly flexible launch vehicles. Another objective was that the automated technique provide some measure of when the design is optimum, something that the engineer with his present techniques generally achieves only through experience.

Section 1.1 of this chapter presents an overview and the historical background of the conventional or classical autopilot design method that has been automated. Section 1.2 contains a statement of the problem and defines the details of the design criteria of this conventional autopilot design method. This chapter concludes with a section that outlines the remaining text of this report.

1.1 Historical Background

Historically, most booster control systems have been designed using classical open-loop frequency response stability margins as the principal design criteria. This linear design phase, which is the topic of this report, is then followed by a nonlinear analysis which includes trajectory simulations. Greensite [14] gives an excellent description of the philosophy and major problem areas as well as the equations involved in the linear analysis phase of launch vehicle auto-

pilot design.

In the linear design phase of this historically well-validated approach, time-varying booster plant dynamics are examined at selected "worst case" flight times along the trajectory. The linearized time-invariant approximation to the dynamics of the airframe at these times is used, and an autopilot feedback configuration is designed that satisfies the specified rigid and flexible body gain and phase stability margins at these flight times. Gain and/or filter scheduling is generally employed to achieve the required margins. The flight times along the trajectory are chosen to correspond to critical flight times at which stability margins are expected to be a minimum. The critical times most often used for a typical ascent stage are:

1. Liftoff
2. Max C_1/C_2 ($C_1 \equiv$ Aerodynamic moment coefficient and $C_2 \equiv$ Control moment coefficient)
3. The point of maximum aerodynamic pressure (max $\bar{q}$)
4. Gain and/or filter change times
5. Burnout

Experience has shown that an ability to demonstrate adequate control system stability margins at these flight times produces satisfactory performance during the complete flight.

1.2 Statement of the Problem

The conventional computer-aided approach to booster control

system design as outlined above has been to use digital computer generated frequency responses to determine the gain and phase margins of the airframe-plus-compensation for a given set of autopilot gains and filters. The design engineer then uses such computer-generated frequency responses to iteratively adjust the autopilot gains and filters until the specified stability margins are achieved.

As stated earlier, the objective of this control system optimization technique was to automate this design procedure.

The details of the design criteria to be satisfied by this automated technique are as follows. The general requirements of an elastic booster autopilot are to (1) provide good response to guidance commands, (2) design the response to external disturbances (e.g., wind) to minimize trajectory dispersions and to ensure that the structural integrity of the vehicle is not jeopardized, and (3) account for tolerances on the vehicle and flight control system parameters.

For this design technique, these general requirements are translated into particular requirements as follows:

1. The design specifications are:
    a. In the frequency domain in the form of minimum gain/phase stability margins, and in the form of so-called dominant closed-loop root locations. Figure 1.1 illustrates a typical gain/phase frequency response plot. Table 1.1 lists the requirements that might be placed

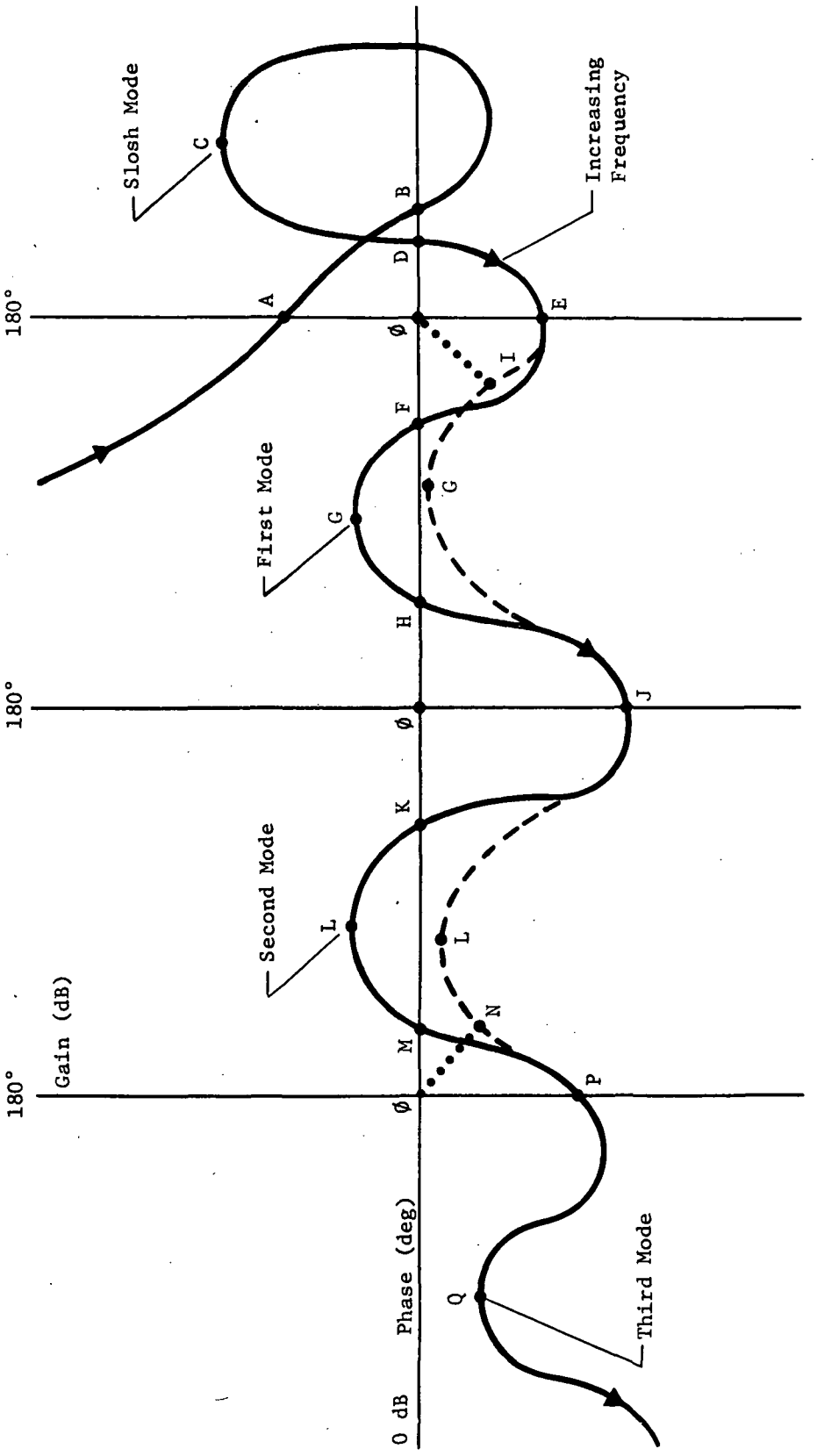


Figure 1.1. Typical Gain/Phase Frequency Response Plot

Table 1.1. Definitions of Gain and Phase Margins

Frequency Range	Reference in Figure 1.1	Typical Design Requirement	Typical Design Objective
Aerodynamic Gain Margin	$\lvert \emptyset A \rvert$	6 dB	6 dB
Rigid-Body Phase Margin	$\measuredangle \emptyset B$	30°	30°
Slosh Mode Backside Phase Margin	$\measuredangle \emptyset D$	20°	30°
Rigid-Body Gain Margin	$\lvert \emptyset E \rvert$	6 dB	6 dB
First Mode Frontside Phase Margin	$\measuredangle \emptyset F$	45°	45°
First Mode Peak Gain	$\lvert G \rvert$	None	20 dB
First Mode Peak Phase	$\measuredangle G$	None	0°
First Mode Backside Phase Margin	$\measuredangle \emptyset H$	60°	60°
First Mode Closest Approach Margin (If First Mode Peak < 0 dB)	$\lvert \emptyset I \rvert$	10 dB (letting 4° ≡ 1 dB)	10 dB
Second Mode 180° Crossover Gain Margin	$\lvert \emptyset J \rvert$	10 dB	10 dB
Second Mode Frontside Phase Margin	$\measuredangle \emptyset K$	60°	60°
Second Mode Peak Gain	$\lvert L \rvert$	None	20 dB
Second Mode Peak Phase	$\measuredangle L$	None	0°
Second Mode Backside Phase Margin	$\measuredangle \emptyset M$	60°	60°
Second Mode Closest Approach Margin (If Second Mode Peak < 0 dB)	$\lvert \emptyset N \rvert$	10 dB (letting 4° ≡ 1 dB)	10 dB
Second Mode 180° Crossover Gain Margin	$\lvert \emptyset P \rvert$	10 dB	10 dB
Third and Higher Mode Peak Gain	$\lvert Q \rvert$	-10 dB	-10 dB
Third and Higher Mode Peak Phase	$\measuredangle Q$	None	0°

on each margin. In addition to minimum requirements, note that Table 1.1 also lists what might be desired design objectives;

b. In the time domain in the form of maximum allowed vehicle structural loads;

c. In the time domain, with the requirement to minimize trajectory dispersions.

2. The technique must design with a user-selected autopilot configuration. In other words, the number and types of feedback loops and the number of gains and filters are selected by the user in an effort to keep the autopilot simple;

3. The technique must handle the problem of multiple time point design. To allow autopilot gains and/or filters to remain constant over intervals of flight while the airframe properties continue to change, it is necessary to design several vehicle states simultaneously;

4. The method must handle a very high order system (30th and greater) with up to eight structural bending and fuel slosh modes per time point;

5. The method must be able to design either a digital or an analog autopilot.

As evidenced from the design criteria, a parameter optimi-

zation technique is required in order to automate this design method. Also, the automated technique must come from the theory of constrained optimization. Finally, even though the system is represented in the time domain with a set of linear equations, the autopilot design problem is nonlinear in the frequency domain. That is, each stability margin is a nonlinear function of the autopilot gains and filters. Hence, the optimization technique must be able to take this into account.

1.3 Outline of the Remaining Text

Chapter 2 contains a discussion of the literature search that took place prior to and during the development of this nonlinear programming algorithm. The chapter concludes with a discussion of why the particular approach was taken that led to the development of the design method.

Chapter 3 contains a discussion of the algorithm itself. It concludes with the details of the step-size optimization routine that is the key to the design method.

This nonlinear programming algorithm, when applied to the booster autopilot design problem, led to the development of a digital computer program titled COEBRA. COEBRA is an acronym for Computerized Optimization of Elastic Booster Autopilots. Chapter 4 presents results of the COEBRA program. These results clearly demonstrate that this algorithm solves the problem of automated practical launch vehicle autopilot design. The results show that all

the items of the design criteria that are listed in Section 1.2 are satisfied.

Chapter 5 presents conclusions and recommendations for further development of the design method.

CHAPTER 2

THE LITERATURE SEARCH

The purpose of this chapter is to discuss the literature search that led to the development of the design algorithm. Three basic approaches to feedback control system optimization were researched.

The first approach is often referred to as Optimal Control of Linear Systems with Quadratic Criteria, or simply, the Regulator Problem. This approach falls under the general theory of Variational Calculus. This is an indirect method of optimization since it derives a so-called "free configuration" control law from the necessary conditions on derivatives of the cost function that must hold in order for an optimum to exist. The term "free configuration" means that no initial assumptions are made concerning the controller configuration.

The second and third approaches that were studied fall under the general category of parameter optimization techniques. These are referred to as direct methods of optimization that optimize the numerical values of parameters within a "fixed configuration" control law via various search procedures. The term "fixed configuration" means that the basic feedback structure of the control law is established as part of the initial assumptions, and optimization involves finding the best numerical values for the gains and filters within this feedback structure.

The second approach is classified as Unconstrained Parameter Optimization. Throughout this report, the use of the word "constraints" refers to the design criteria like minimum allowed gain and phase margins, etc., and does not refer to the set of linear equations that define the system. The term "unconstrained" is used to mean that the search procedure only seeks to minimize a cost function, and any constraints must be included in the cost function.

The third approach is classified as Constrained Parameter Optimization, where the search procedure seeks to minimize a cost function in the presence of constraints.

2.1 Optimal Control of Linear Systems with Quadratic Criteria

This section discusses why this approach was not pursued. Despite recent advances [Anderson and Moore, 1], this approach does not directly treat the problem of control system design with fixed configuration control laws and stability margin design criteria. However, the literature in this area was searched since so many attempts at flight control system design using this method have been made, even though these attempts have largely been unsuccessful.

Kalman [20] showed that the solution (feedback control law) of the single-input time-invariant Regulator Problem, yields a return difference with a magnitude that is greater than or equal to unity at all frequencies. (This assumes that the weighting factors in the quadratic cost function have been chosen so that the closed-loop solution is

assymptotically stable.) Kalman viewed this as resulting in reduced sensitivity to plant variations.

Anderson and Moore [1, Chapter 5] extended this result to show that the Regulator Problem results in a system that has the following gain/phase stability margins. All gain margins that are greater than unity have at least a 6 db margin of stability. All gain margins that are less than unity have an infinite margin of stability. All phase margins are greater than or equal to 60 degrees. This is an astonishing result, and answers the critics who for years discarded this theory because they thought it had no direct relationship to the classical design methods that use stability margin design criteria. However, though at first this result may appear very attractive for the booster autopilot design problem, it must be noted that this theory requires state feedback. For the booster autopilot design problem, state feedback is not considered a viable candidate for two reasons. First, a requirement of this design method is to let the user select the feedback configuration. Second, a state estimator like a Kalman filter, is only as good as the system model, and presently, structural bending mode parameters on a large flexible launch vehicle are not known with much precision. Hence, since state feedback is not considered viable, the results of Anderson and Moore are not useful or applicable to the booster autopilot design problem.

Several other important reasons why this approach was not

pursued are as follows. This design method cannot handle multiple time points and/or vehicle states with the same time-invariant control law. Also, with this design method, it is not possible to directly constrain the values of the feedback gains. This becomes important when considering such things as nonlinearities, offsets, and control device deflections Finally, use of this theory is still an art. Since the "intelligence" of the design method is in the cost function, there is a skill or art involved in the selection of the proper weighting factors.

The references relating to the Regulator Problem are categorized according to two classifications. The first is the time domain solution to the Regulator Problem which involves solution of the Ricatti equation. The second is the frequency domain solution to the time invariant Regulator Problem which involves solution of the Wiener-Hopf equation. This frequency domain method relates closed-loop roots to the control law gains without finding or specifying the cost function weighting factors. The references deal with both booster and airplane flight control system design.

Many attempts [3, 4, 8, 12, 16, 24, 25, 26, 36, 37] have been made at designing flight control systems using the time domain solution to the Regulator Problem. Burris and Bender [4] are the authors of the celebrated "LAMS" effort. The works of Stein and Henke [36] and Vandierendonck [37] are worth noting since they use a blended

approach to the problem of multiple vehicle state design with the same time-invariant control law. In these works, the "Ricatti equation" approach is used first to derive an optimal control law for each single flight condition. These control laws then serve as the starting point for the second phase of design which uses a gradient search method with a quadratic cost function, to solve the multiple vehicle state problem.

Several attempts [18, 31, 32, 38] at flight control system design have been made using the frequency domain approach to solving the Regulator Problem.

2.2 Unconstrained Parameter Optimization Methods

As stated earlier, these are direct optimization methods where the search procedure seeks only to minimize a cost function, and any constraints must be included in the cost function. This section discusses why these methods were not pursued.

2.2.1 The Algorithm of Stear and Lefkowitz

The work of Stear and Lefkowitz [35] contains a computerized algorithm which the authors claim results in selection of gains and filters within a given autopilot feedback structure that will stabilize a booster by producing a gain/phase frequency response that will satisfy a set of specified stability margins at critical flight times. This algorithm is based on minimizing a nonlinear cost function which is a function of the violations of the specified stability margins.

Gradient search techniques are used to minimize this cost function.

The authors have made this algorithm work on a relatively simple single-time-point problem with a fourth order filter in a single-loop feedback structure.

There are three reasons why it was felt that this technique, which has all the intelligence in the cost function, was not direct enough to handle the multiple time point problem of a highly flexible launch vehicle with a very large number of stability margin constraints: (1) It does not deal directly with each individual margin constraint; (2) The objective is to not only meet stability margin requirements, but also to optimize or maximize all margins. In other words, it is desired to have minimum margin requirements plus desired margin objectives; (3) An additional objective is to optimize structural load relief capability during the periods of high aerodynamic loading on the vehicle. Since optimizing load relief capability reduces stability margins, this objective requires a cost function that behaves nearly like the inverse of a cost function that only contains stability margin information.

2.2.2 AUTO

Coffee [7] developed an automated booster autopilot design algorithm called AUTO. Coffee's approach is to choose a predetermined open-loop frequency response by selecting and specifying the loop gain and phase at various frequencies, i.e., by selecting points

on a gain-phase diagram corresponding to a set of pre-selected frequencies. He then chooses an autopilot configuration of prescribed complexity, but with nonoptimum gain and filter values, and formulates a cost function which is the mean-square difference of the predetermined frequency response curve and the actual frequency response curve at the given frequencies. Gradient search techniques are used to find the autopilot gain and filter values which minimize the cost function.

The algorithm, which seeks to fit an actual frequency response to a desired frequency response in the least mean-squared-error sense, is considered inadequate since it is impossible to select a priori the desired frequency response of a multiple time point problem for a highly flexible launch vehicle. In other words, it is not possible to select beforehand, the resonant location (gain and phase) of all bending and fuel slosh modes at all time points. Further, it is not clear that a particular frequency response profile is even desirable. The design problem is primarily concerned with stability margins which correspond to selected points on the frequency response profile such as crossover points and gain peaks, and not with achieving a particular desired frequency response.

Even if the desired frequency response could be specified a priori, this method does not directly treat each individual margin and hence is considered inadequate for the multiple time point problem.

Finally, Coffee's algorithm does not provide for load relief optimization in the presence of constraints on the stability margins.

2.2.3 The Algorithm of Stapleford, et al

The algorithm developed by Stapleford, et al [34], uses a version of the parallel tangents method (partan) [Wilde and Beightler, 39, Chapter 7] to minimize a quadratic cost function. The cost function includes pilot tracking errors and control device deflections. The cost function, which is expressed in the time domain with infinite terminal time, is evaluated using Parseval's theorem [Chang, 6, Chapter 2]. The method of finite differences is used to calculate the direction of steepest descent from the cost function. This method was not selected primarily since all the "intelligence" lies in the cost function.

2.2.4 Others

The classical works of Fletcher and Powell [13], Davidon [10], Shah [33], and Hooke and Jeeves [19], are categorized as unconstrained direct optimization methods. Since these methods contain all the intelligence in the cost function, they were not considered adequate for this problem which is really dominated by constraints.

2.3 Constrained Parameter Optimization Methods (Nonlinear Programming)

The methods classified under nonlinear programming are direct optimization methods where the search procedure seeks to

minimize a cost function in the presence of constraints. This section contains a brief description of three of the nonlinear programming algorithms that were found in the literature. The first is Rosen's Projected Gradient Method [29, 30]. The second is the Differential Algorithm method of Wilde and Beightler [39, Chapter 3]. The third is the Multiple-Gradient-Summation Technique developed by Klingman and Himmelblau [21].

2.3.1 The Projected Gradient Algorithm (PGA)

Rosen's projected gradient algorithm [29, 30] is an iterative technique designed to solve a general class of nonlinear programming problems. PGA employs cost-function and constraint gradient information to replace the multidimensional optimization problem by an equivalent sequence of one-dimensional searches. In this manner, PGA solves a difficult multidimensional problem by solving a sequence of simpler problems. In general, at the initiation of the iteration sequence, PGA is primarily a constraint satisfaction algorithm. This is because the initial search point generally does not fall within the feasible region. As the iteration process proceeds, the emphasis changes from constraint satisfaction to cost-function reduction.

The projected gradient method uses two basic search directions. For the purpose of this discussion, they will be termed the constraint and optimization directions, respectively. PGA proceeds by taking successive steps in one or the other of these two directions.

The constraint direction is used when the present search point is unfeasible. A step in this direction ignores the cost function, and is taken so as to "get feasible" in the shortest distance possible. When a feasible point is reached, the optimization direction is used. Obviously the steepest descent direction would be the best local search direction for reducing the cost function. Such a direction, however, would generally produce unacceptable constraint violations. To avoid this difficulty, PGA orthogonally projects the unconstrained cost function gradient into a direction parallel to the local linearized constraint boundary. By searching in this projected gradient "optimization" direction the algorithm attempts to avoid further constraint violations. For an "unconstrained" optimum the algorithm will converge to the local optimum directly via the cost function gradient. For a "constrained" optimum, the algorithm will converge to the local optimum where the projection of the cost function gradient onto a vector parallel to the linearized constraint is zero.

2.3.2 The Differential Algorithm

This technique, developed by Wilde and Beightler [39, Chapter 3], employs the concept of the constrained derivative. It seeks the local "constrained" or "unconstrained" optimum at which point the so-called positivity and complementary slackness conditions are satisfied. Wilde and Beightler point out that their algorithm is really the

nonlinear version of the Simplex Algorithm [9]. They also point out that if the constraints are equalities, their algorithm is really the Method of Undetermined Lagrange Multipliers [39].

2.3.3 The Multiple-Gradient-Summation Technique

This method, developed by Klingman and Himmelblau [21], is briefly defined as follows. Assuming that the initial point is feasible, the technique, when optimizing the cost function, employs the method of Pattern Search until a constraint is encountered. At the constraint boundary, the search direction becomes the vector sum of the normalized cost function gradient and the normalized gradient of the "encountered" constraint.

If the initial point is not feasible, the algorithm first "gets feasible" by ignoring the cost function and basically proceeding in the direction perpendicular to the feasible region.

2.4 Conclusions of the Literature Search

This section contains a discussion of the three main factors that led to the development of this new nonlinear programming algorithm. These factors, which evolved from the literature search, point out why a new method was developed and why the nonlinear programming methods that were found in the literature were not used. This section will conclude with a summary and overview of this new procedure, and how it was applied to the problem of booster autopilot design.

The three factors are now itemized.

1. While it was felt that any of the three nonlinear programming methods of Section 2.3 could be made to work, the high dimensionality of the control vector (i.e., the large number of autopilot gains and filters) and the large number of constraints in the booster autopilot design problem pointed to the desirability of Linear Programming [Dantzig, 9]. The size of the control vector and constraint matrix make the schemes in the literature very complicated, and experience with them was limited to low dimensional problems.
2. The autopilot design problem is highly nonlinear, but the constraints on the stability margins cannot be expressed as explicit analytical functions of the autopilot gains and filters. The Taylor Series expansion is the logical expression to use in writing equations for the margins, but anything higher than first order terms would be difficult and expensive to obtain. The reason for this is because the method of finite differences would have to be used in obtaining any derivatives, and the accuracy with which second and higher order derivatives could be calculated is questionable.
3. The nonlinear programming methods that were found in

the literature linearize the problem at each step anyway by treating only the gradients of the cost function and the constraints. Nonlinear programming methods also usually linearize the constraint equations in order to determine step sizes.

These three factors led to the concept that is the basis of this nonlinear programming algorithm, namely, iterating from some starting point to the final solution by solving successive linear programming problems.

The method of approach is to expand each stability margin in a first order Taylor Series about its nominal value that results from some initial autopilot. Note that the nominal value for each margin is found by searching the frequency response, and note that the first derivatives of the stability margins with respect to the autopilot variables are calculated by the method of finite differences. Via this Taylor Series expansion, an inequality constraint is put on each margin. In order to maximize stability margins, the cost function is formed so that when it is maximized, the first order terms in the Taylor Series are also maximized, thereby maximizing the margins themselves. The constraint matrix also includes minimum and maximum allowed values on the autopilot variables. A linear programming problem results from this linear cost function which is to be maximized in the presence of linear constraints on each individual

stability margin and autopilot variable. The solution to this linear programming problem is a new "nominal" autopilot. The problem is relinearized about this new nominal via a "new" Taylor Series expansion. This yields a "new" linear programming problem, whose solution is another new nominal autopilot, etc.

Using this approach, the cost function for optimizing load relief can be different from the cost function that optimizes stability margins. One idea for this cost function is to use a separate transient response routine to calculate angle of attack and control deflection response due to a specified wind profile. These responses would be expanded in a first order Taylor Series, and the cost function would be formed from the first order terms. The linear programming problem would adjust autopilot gains and filters so as to minimize this cost function (which is a measure of structural bending moment loads [Harris, 15]) in the presence of constraints on the stability margins and autopilot variables.

From the outset, only practical controllers are considered, since the designer specifies the basic feedback structure, and puts constraints on each individual autopilot gain and filter.

The multiple time point problem is handled as follows. Stability margin requirements from several time points can be used to form a single cost function and a single constraint matrix. Autopilot gains and/or filters can be constrained to be shared between the several

time points. By solving this single problem (that includes margin and autopilot constraints and cost function objectives from several time points) these several time points are then optimized all at the same time. Hence, the requirement of multiple time point design with the same time-invariant control law is satisfied.

Since this is a parameter optimization scheme, the complexity of the autopilot does not necessarily increase with an increase in the order of the fixed parts of the system. Inclusion of actuator and sensor dynamics, etc., is straightforward, and only increases the computations required by the algorithm.

CHAPTER 3

THE DESIGN ALGORITHM

This chapter will outline the details of this nonlinear programming algorithm as it is applied to the problem of booster autopilot design. Simply stated, the basis of the optimization technique is the iterative application of linear programming. As stated in Chapter 2, the constraint equations consist of stability margin design requirements and constraints on the values of the individual autopilot variables. The cost function is formed to either maximize stability margins or maximize load relief capability.

Section 3.1 will briefly discuss linear programming in general. The remaining sections will define how the problem of booster autopilot design is adapted to the iterative application of linear programming. Section 3.2 illustrates the stability margin and autopilot variable constraint equations. Section 3.3 illustrates the two types of cost functions that are required in launch vehicle autopilot design (maximize stability margins and maximize load relief capability). Section 3.4 will discuss a general flow chart of the design algorithm.

Section 3.5 discusses the two design "mechanisms" that really make the algorithm work. These two mechanisms have to do with step-size optimization, and allow steady convergence to a local optimum, particularly to an interior optimum. This chapter then concludes

with a discussion of detailed flow charts of the design method.

3.1 Linear Programming

A linear programming algorithm solves the following problem: maximize the linear cost function, y, where

$$y = \sum_{j=1}^{n} a_j X_j$$

subject to a matrix of linear constraint equations

$$\sum_{j=1}^{n} b_{ij} X_j \left(\begin{matrix} \geq \\ \leq \\ = \end{matrix} \right) c_i \qquad (i = 1, \ldots, m)$$

and $$X_j \geq 0 \qquad (j = 1, \ldots, n)$$

For the booster autopilot design problem, the variables (X_j) are the autopilot gains and filters.

Dantzig [9] is the author of the Simplex Method which is a technique for solving the linear programming problem. Wilde and Beightler [39, p. 138] show that the Simplex Method is a special case of their Differential Algorithm which was discussed in Section 2.3.2 of Chapter 2.

3.2 Constraint Equations

This section illustrates the stability margin and autopilot variable constraint equations.

3.2.1 Stability Margin Constraint Equations

The types of margins that are normally considered in launch vehicle autopilot design are illustrated in Figure 1.1 of the first chapter. Basically, these margins can be categorized as rigid-body, flexible-body, and fuel slosh margins.

A routine can be used to search the frequency response and identify each stability margin that is to be treated. Each stability margin, $M_i(\bar{X})$, is then expanded in a first order Taylor Series about its nominal value, $M_{io}(\bar{X}_o)$. It is required that $M_i(\bar{X})$ be greater than or equal to some specified value, M_{is}. Hence each margin constraint equation can be written as follows:

$$M_i(\bar{X}_o + \Delta\bar{X}) = M_{io}(\bar{X}_o) + \sum_{j=1}^{n} \left(\frac{\partial M_i}{\partial X_j}\right)_o \Delta X_j \geq M_{is}$$

$$(i = 1, \ldots, m)$$

This expression is now rewritten in the form of the constraint equations of Section 3.1. Note that $\Delta X_j = X_j - X_{jo}$, where X_{jo} is the value about which the Taylor Series if formed.

$$\sum_{j=1}^{n} \left(\frac{\partial M_i}{\partial X_j}\right)_o * X_j \geq M_{is} - M_{io}(\bar{X}_o) + \sum_{j=1}^{n} \left(\frac{\partial M_i}{\partial X_j}\right)_o * X_{jo}$$

$$(i = 1, \ldots, m)$$

Since explicit expressions are not available for each margin

as a function of the autopilot parameters, the method of finite differences must be used to calculate the partial derivatives. Computing the derivatives this way allows more freedom of configuration with both the airframe equations and with the autopilot equations. Obviously, the matrix of stability margin constraint equations can be formed from margins at several different vehicle states. This satisfies the requirement that the design algorithm must be able to handle the problem of multiple vehicle state design with the same time-invariant control law.

3.2.2 Autopilot Variable Constraint Equations

A requirement of this automated design technique is that the user must be able to specify the autopilot configuration. In other words, he must be able to specify the number and types of feedback loops and the number of gains and filters within each loop. Further, the user must be able to specify which autopilot parameters are to be treated as constants and which are to be treated as variables. He must then be able to individually constrain each variable. The algorithm then optimizes each variable within the constraints.

Constraint equations on each autopilot variable can be written as follows:

$$X_{jl} \leq X_j \leq X_{ju} \qquad (j = 1, \ldots, n)$$

Section 3.5 will define X_{jl} and X_{ju} in more detail, but basically they are functions of step-sizes, and minimum and maximum allowed values.

As with the margin constraint equations, the autopilot variable constraint equations can include variables from several different vehicle states. Also, to handle the problem of multiple vehicle state design, a single constraint equation can be used for a variable that is to have the same value at several different vehicle states.

3.3 Cost Functions

This section will illustrate the two types of cost functions that are required in launch vehicle autopilot design. The two types are: (1) a cost function to maximize stability margins; and (2) a cost function to maximize structural bending moment load relief capability.

3.3.1 Stability Margin Cost Function

When the objective is to maximize stability margins, the cost function is formed so that when it is increased, all the margins at all the time points will tend to increase together, and each structural bending mode will tend to resonate near zero degrees phase. To accomplish this, the cost function is written so that when it is increased, the first order terms in the Taylor Series expansion of each margin about its nominal value, will also be increased.

The objective is to maximize the following expression, which is a "weighted" linear combination of the "variable" portion of the first order terms in the Taylor Series expansion.

$$Y_1 = \sum_j \sum_t \sum_i W_1(i,\ t) * W_2(i,t) * S(i) * \left(\frac{\partial M(i,t)}{\partial X_j} \right)_o * X_j$$

In the above expression:

(1) j refers to the summation over all the autopilot variables;

(2) t refers to the summation over all the time points or vehicle states;

(3) i refers to the summation over all the stability margins at all of the time points;

(4) $W_1(i,t)$ refers to a weighting factor. For each margin, it is simply a ratio of the desired margin over the actual margin. Hence, if a margin is not met, $W_1(i,t)$ will be greater than unity. It becomes less than unity when a margin exceeds its desired objective. It is noted at this time that in the expression for Y_1, i also indexes the phase angle at which each structural bending mode resonates. For these values of i, the partial derivative indicates the rate of change of each modal peak phase with respect to each autopilot variable, and $W_1(i,t)$ is written so that the algorithm will attempt to force each mode to resonate near zero degrees phase. $W_1(i,t)$ will be large for modes that resonate near 180 degrees, and zero for modes that resonate at zero degrees. For some arbitrary angle like 90 degrees, $W_1(i,t)$ can equal unity.

(5) $W_2(i,t)$ refers to a weighting factor that might be selected by the user. This would give the user the capability to

eliminate certain margins from the optimization process or to emphasize other margins.

(6) S(i) refers to a scale factor. It serves to scale the margins and modal peak phases so that phase margins and gain margins can be optimized together. For example, it might be desired to equate a five degree increase in the rigid-body phase margin with a one decibel (12.2%) increase in the rigid body gain margin. S(i) would be used to reflect this desired scaling.

In summary, Y_1 is a "weighted" linear combination of the "positive" changes in each margin and modal peak phase. Note that this linear combination can incorporate margins from all of the vehicle states that are being designed together. The design algorithm will maximize Y_1 (and hence seek to maximize all stability margins and seek to force all modes to resonate near zero degrees phase) in the presence of the constraint matrix which includes constraints on each individual margin and each autopilot variable at each time point.

The following paragraph discusses an advantage of a design algorithm that maximizes a cost function in the presence of constraints, as opposed to one that includes the constraints in the cost function [7, 35]. With a separate cost function and constraint matrix, the constraint equations can specify the minimum requirements on each design goal, while the cost function can seek to maximize each design goal.

In other words, the cost function can reflect the desired objectives (in the weighting factors) while the constraint matrix can reflect the minimum requirements.

3.3.2 Load Relief Cost Function

Structural bending moment loads on a launch vehicle are largely due to axial acceleration, aerodynamic loading, and control device deflections [Harris, 15]. Obviously, the booster autopilot can do little to affect axial acceleration, and therefore the main objective of a so-called load relief autopilot is to reduce aerodynamic loading due to angle of attack and to keep control device deflections to a minimum.

Hence, for this design algorithm, when the objective is to maximize structural bending moment load relief capability, the cost function is comprised of the response of the angle of attack (β) and the control deflections (δ) due to the wind forcing function (β_w). When the cost function is maximized, the peak values of β and δ are minimized.

A separate transient response routine is used to calculate the peak values of angle of attack (β_p) and control deflection (δ_p) due to β_w. As with stability margins, the method of finite differences is used to compute the first partial derivatives of β_p and δ_p with respect to the autopilot variables. The cost function is then formed from the first order terms of the Taylor Series expansions of β_p and δ_p about their nominal values.

As with the stability margin cost function, the load relief cost function (Y_2) is a weighted linear combination of the variable portion of the first order terms in the Taylor Series. Y_2 is given as follows:

$$Y_2 = \sum_j \sum_t \left[W_1(t) * \left(\frac{\partial \beta_p(t)}{\partial X_j} \right)_o + W_2(t) * \left(\frac{\partial \delta_p(t)}{\partial X_j} \right)_o \right] * X_j$$

In the above expression:

(1) j refers to the summation over all the autopilot variables;

(2) t refers to the summation over all the vehicle states;

(3) $W_1(t)$ and $W_2(t)$ refer to weighting factors that are input by the user.

When maximizing load relief capability, the design algorithm will maximize the negative of Y_2 in the presence of the constraint equations on the minimum allowed gain/phase stability margins and on the allowed ranges of the individual autopilot variables. Note that multiple time point design is handled just as it is when maximizing stability margins. Some final notes on the load relief cost function are now listed.

Since the so-called "rigid-body" (as opposed to flexible-body) angle of attack (β) and control deflection (δ) are the principal factors in determining structural bending moment loads, it is felt that only the rigid-body airframe equations of motion [Harris, 15] need to be used in the transient response routine that is used to calculate angle of

attack and control deflection. Note also that these rigid-body airframe equations can include planar coupling (e.g. between the yaw and the roll planes), and hence the cost function can include control deflections from several planes (e.g. the yaw plane control deflections (δ_ψ) and the roll plane control deflections (δ_ϕ)).

The wind forcing function can be a series of steps and/or ramps that approximate the commonly used synthetic wind profile [15]. The wind forcing function could also be stochastic, and the design algorithm would then minimize the rms values of β and δ. This would be done via Wiener's theorem and the filtering property of power spectral density functions [Chang, 6].

3.4 General Flow Chart

Figure 3.1 is a general flow chart summarizing the main steps involved in the algorithm. It shows the general flow from the initial autopilot for each iteration through the following routines:

(1) The routine that generates the frequency response and finds the stability margins, and the routine that generates the transient response and finds peak β and δ;

(2) The routine that computes sensitivities or partial derivatives;

(3) The routines that set up the linear programming problem and solve it; and

(4) The routines that determine whether the design is com-

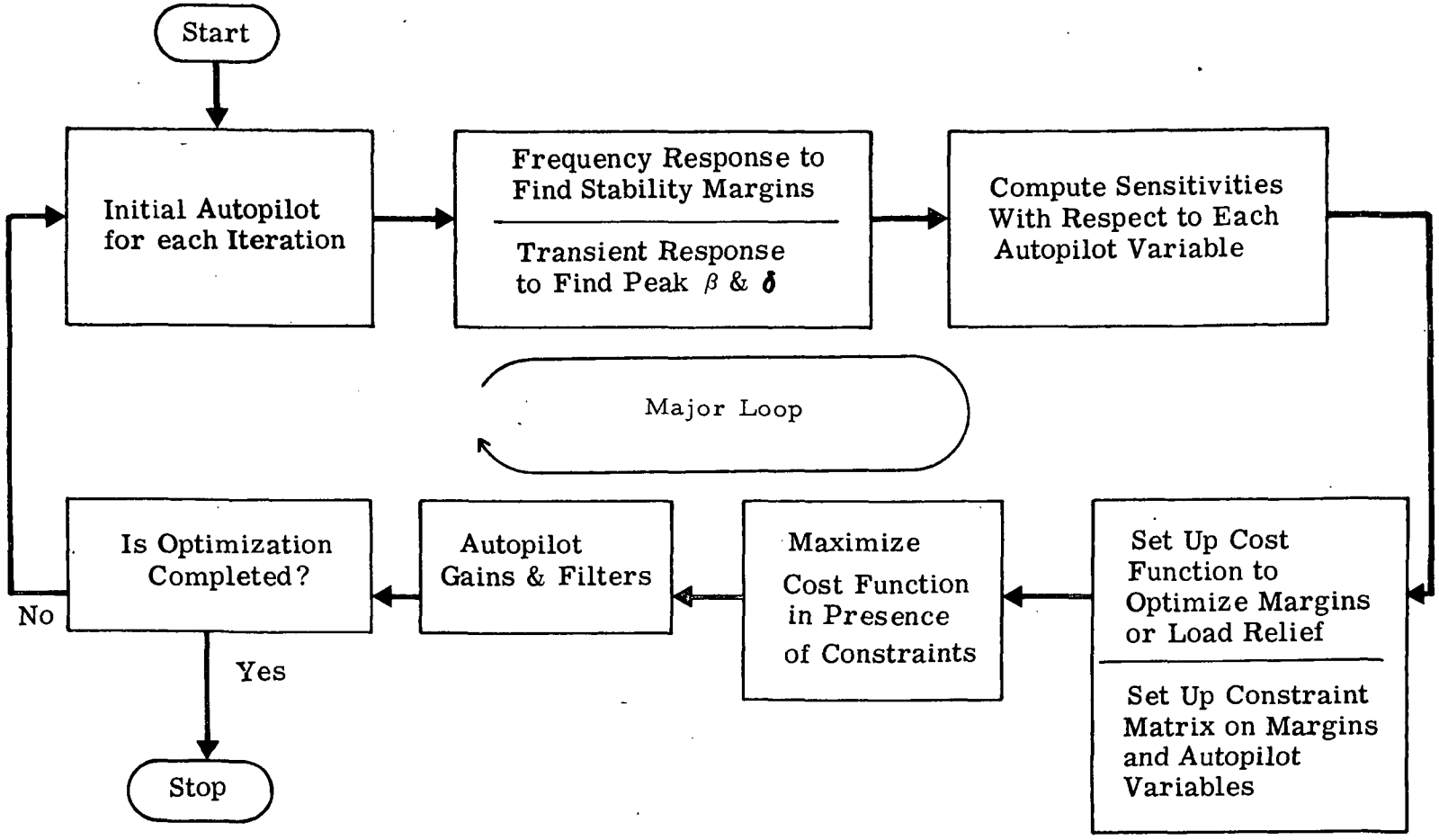


Figure 3.1. General Flow Chart of the Design Algorithm

plete. These routines are discussed in the next section (Section 3.5, Step-size Optimization).

If the design if not complete, another major iteration is begun with the best answer obtained in the previous iteration. In other words, the problem is relinearized about the best answer of the previous major loop, and another cycle through the major loop is performed. This iterative process continues until the local optimum is found.

Note that this design process satisfies the five main elements of the design criteria as outlined in Section 1.2 of Chapter 1.

(1) The method directly treats stability margin requirements and objectives, and structural bending moment load reduction.

(2) The method directly handles the user-selected autopilot configuration.

(3) The method directly handles the multiple time point design problem.

(4) The method is not limited by the order of the system. Note that since this is a parameter optimization routine, the order of the autopilot does not necessarily increase with an increase in the order of the fixed parts of the system.

(5) The method can design either a digital autopilot (via the W-plane) or an analog autopilot (via the S-plane).

3.5 Step-size Optimization

This section discusses the details of the step-size optimization routine that is the key to this design algorithm. This step-size optimization routine is divided into two parts: (1) the Minor loop which is the autopilot variable step-size loop; and, (2) the Inner loop which might be referred to as the stability margin step-size loop.

3.5.1 The Minor Loop

Section 3.2.2 presented the general expression for the constraint equation for each autopilot variable (X_j). The following is the detailed expression for this constraint equation.

$$\text{MAX} \left\{(1+P)^{-1} * X_{jo},\ X_{j\,min}\right\} \leq X_j \leq \text{MIN} \left\{(1+P) * X_{jo},\ X_{j\,max}\right\}$$

$$\text{for } (j = 1, \ldots, n)$$

where:

(1) $X_{j\,min}$ and $X_{j\,max}$ refer to the minimum and maximum values ever allowed for X_j.

(2) X_{jo} refers to the initial value of X_j on each iteration. Note that X_{jo} is the point about which the partial derivatives are computed, and about which the Taylor Series is expanded.

(3) P refers to the autopilot variable step-size for each iteration.

In words, if $X_{j\,min}$ and $X_{j\,max}$ are not encountered on a particular iteration, the above constraint equation says that X_j is allowed to vary no more than about $\pm$ P% from X_{jo} on any iteration. Since it is desirable to maximize the step size on each iteration, thereby getting the maximum "mileage" out of each set of partial derivatives, it is desirable to have a Minor Loop that increases the size of P until improvement in that "search direction" is no longer possible. In other words, the Minor Loop serves to maximize the autopilot variable step-size. In maximizing P, the Minor Loop uses two "indicators": (1) a counter that keeps track of the number of stability margins that are already met, and (2) a figure-of-merit that is a linear combination of the actual margins. If the number of "met margins" increases, obviously the value of P can be increased. If the number of "met margins" does not change, the figure-of-merit is used to decide whether P can be further increased. In other words, the margin counter is used to reward those steps that result in an increase in the number of "met margins". Conversely, the counter prohibits those steps that result in a loss in the number of "met margins". Finally, the figure-of-merit is used to break ties when the margin counter does not change from one step to another.

Section 3.5.3 contains a graphical illustration of the Minor Loop, but basically the Minor Loop serves to either keep the problem linear on each major iteration, or to take advantage of the neglected

nonlinearities when they might be helpful. In other words, the Minor Loop serves to keep the nonlinearities from "hurting" the steady convergence to a local optimum.

A major benefit of the minor loop is that it allows the algorithm to converge steadily to an "interior" optimum. This is explained as follows. Since the solution to the linear programming problem always lies at a vertex of the feasible region defined by the constraint equations, it is the Minor Loop that allows the algorithm to converge to a local optimum that is interior to the stability margin constraint equations. As mentioned earlier, Section 3.5.3 will graphically illustrate the mechanics of the Minor Loop, but first a brief discussion of the Inner Loop is in order since the Minor and Inner Loops work together. Section 3.5.3 will then illustrate this interaction.

3.5.2 The Inner Loop

The second part of the step-size optimization routine can be illustrated by the following detailed expansion of a particular margin constraint equation.

$$M_{io}(\bar{X}_o) + \sum_{j=1}^{n} \left(\frac{\partial M_i}{\partial X_j}\right)_o * (X_j - X_{jo}) \geq M_{is} \quad (i = 1, \ldots, m)$$

In the above expression, there are two cases for M_{is}.

(1) If the particular margin is already met, then for the next iteration,

$$M_{is} = SPEC(i)$$

where SPEC(i) is the minimum allowed value for the i^{th} margin.

(2) If the particular margin is not yet met,

$$M_{is} = M_{io}(\bar{X}_o) + STEP * \left[SPEC(i) - M_{io}(\bar{X}_o)\right]$$

Before defining the purpose of the equations for M_{is}, note that in the second equation, (a) if STEP = 1, M_{is} = SPEC(i), and (b) if STEP = 0, $M_{is} = M_{io}(\bar{X}_o)$.

The purpose of these equations is now illustrated. For a given value of P (autopilot variable step-size), there may not be a feasible solution to the linear programming problem if the present autopilot does not meet all of the stability margin constraints. In other words, the feasible region defined by the margin constraint equations may not overlap the feasible region defined by P. By automatically reducing the value of STEP, the margin constraints are "loosened", until a feasible solution is possible for a given value of P. In this way, the value of P can be increased in a steady and rational manner, and the algorithm will be allowed to converge to a solution in a progressively improving manner.

3.5.3 Graphical Illustration of Step-size Optimization

Figures 3.2 through 3.7 graphically illustrate the mechanics and the interaction of the two step-size optimization routines (the Minor Loop and the Inner Loop).

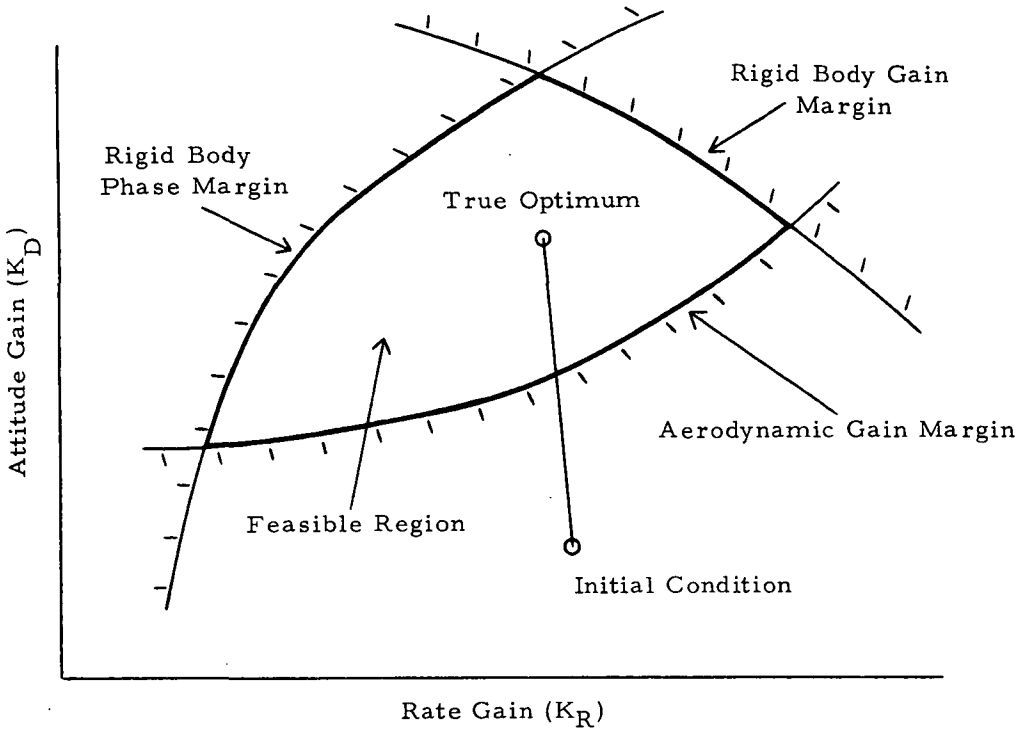


Figure 3.2. Nonlinear Constraints

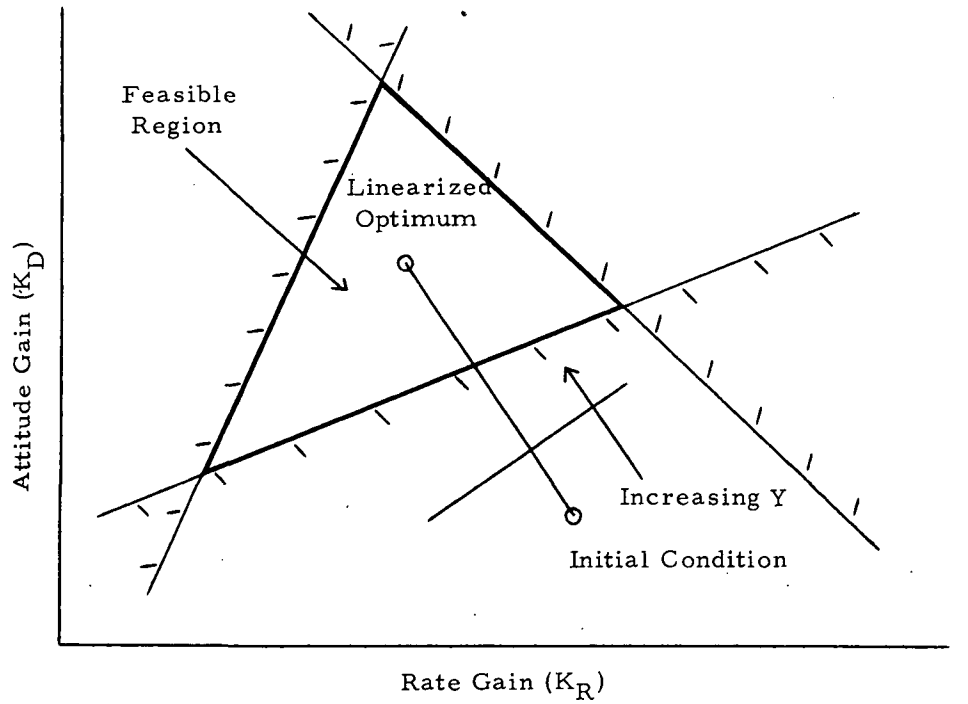


Figure 3.3. Linearized Constraints

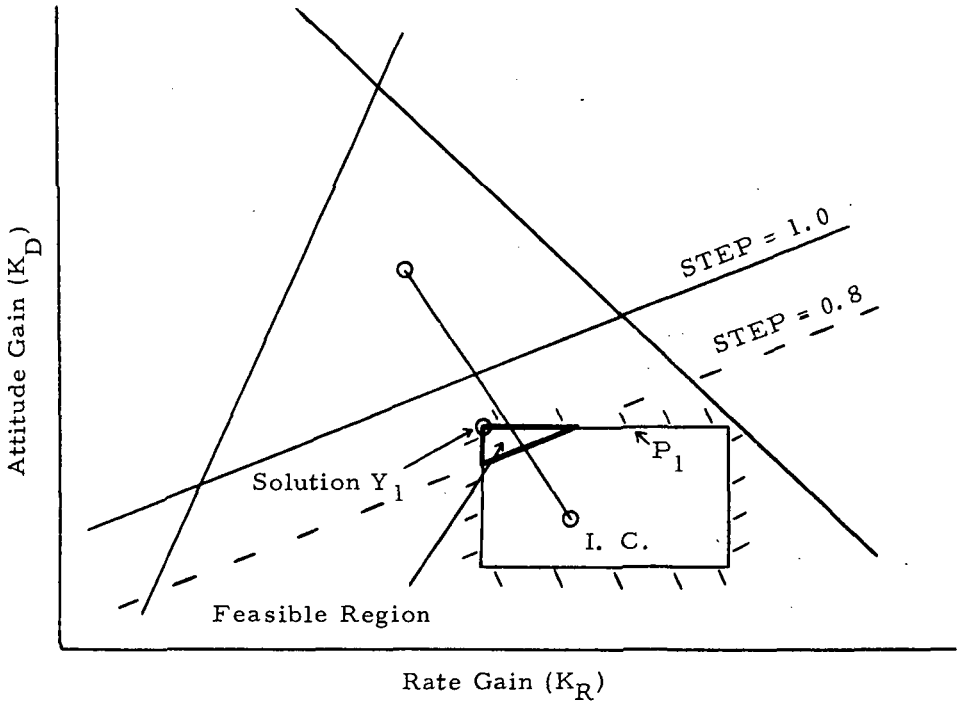


Figure 3.4. Minor Loop Step-size #1 (P_1)

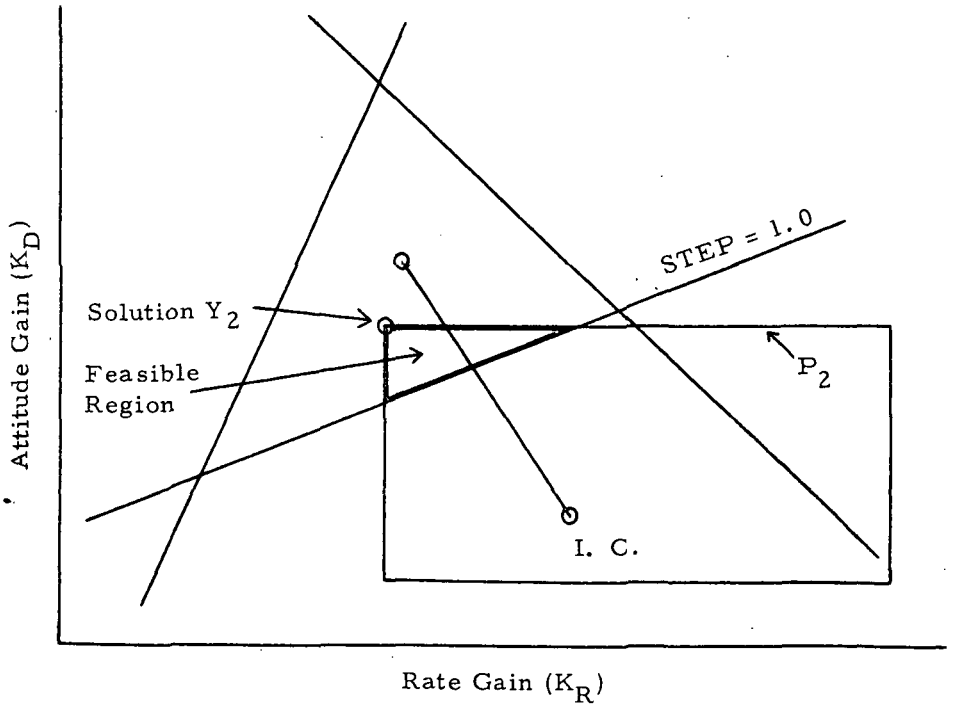


Figure 3.5. Minor Loop Step-size #2 (P_2)

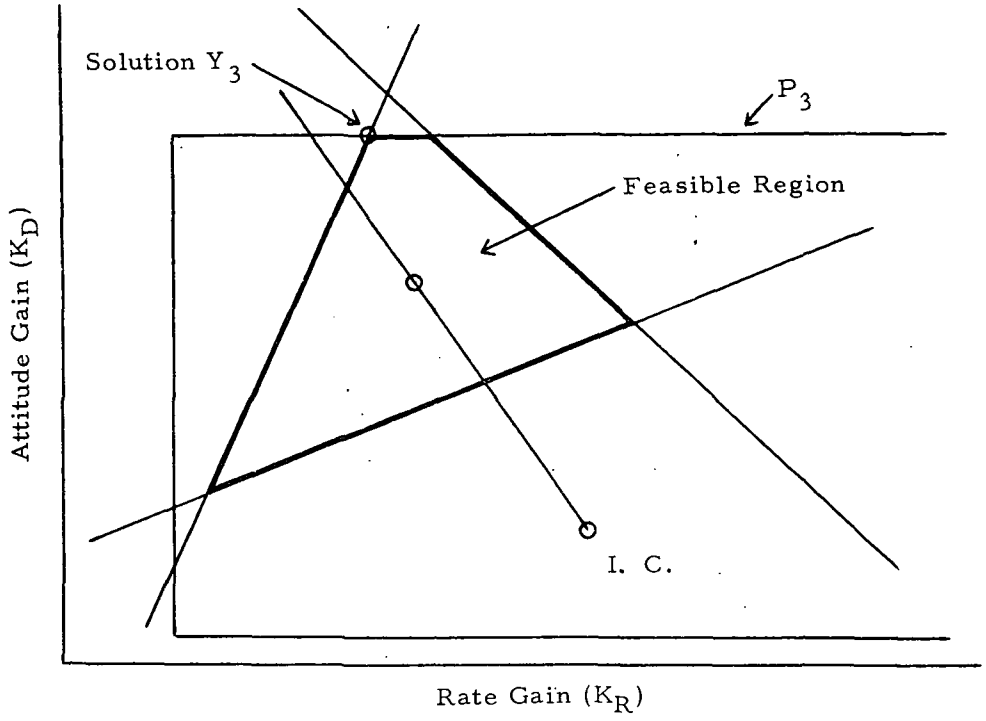


Figure 3.6. Minor Loop Step-size #3 (P_3)

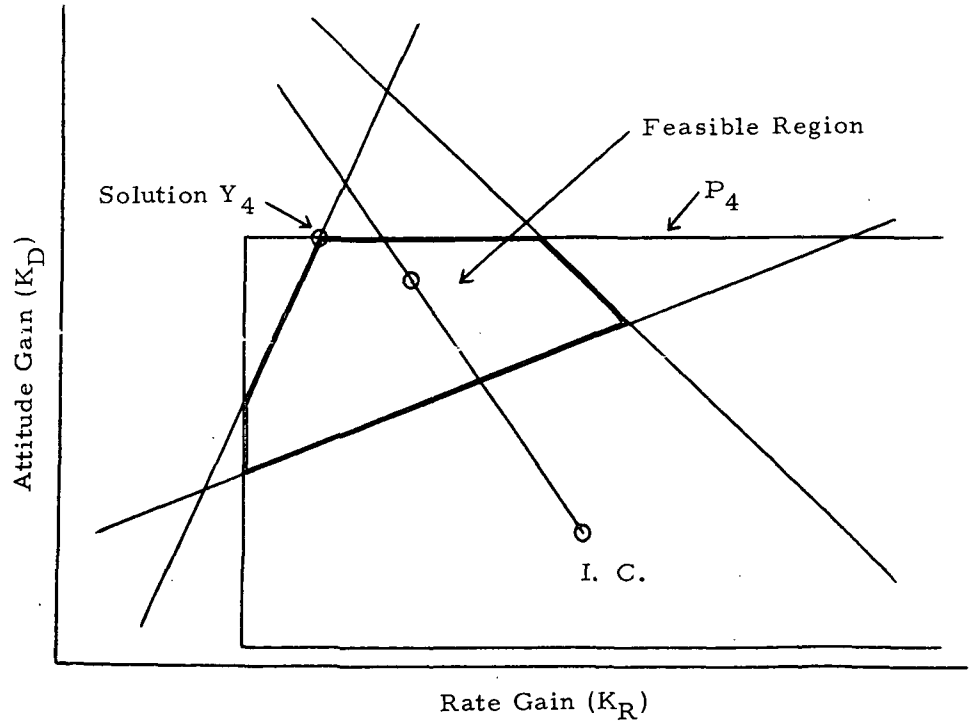


Figure 3.7. Minor Loop Step-size #4 (P_4)

Figure 3.2 shows a hypothetical two dimensional condition that might exist for a rigid-body autopilot design problem. Figure 3.2 is a plot of the attitude error gain (K_D) versus the attitude rate gain (K_R). Plotted on the figure are three nonlinear stability margin constraint equations: (1) the aerodynamic gain margin; (2) the rigid-body phase margin; and, (3) the rigid-body gain margin. Figure 3.2 also shows where the "true" local optimum condition might be, where the objective is to maximize stability margins, and where all three stability margins are equally weighted. Obviously, the "true" optimum for this hypothetical case lies inside the feasible region where all three margin requirements are satisfied. The figure also shows what might be the "first guess" or initial condition on K_D and K_R.

Figure 3.3 shows what the constraints might look like when they are linearized about the initial condition. The figure also shows the linearized interior optimum, where again, all margins have been equally weighted. Note that the linearized optimum is not the same as the nonlinear or "true" optimum for this initial condition. Finally, Figure 3.3 shows the slope of the linearized cost function (Y) and the direction in which it increases.

Figure 3.4 illustrates the feasible region defined by the autopilot variable constraint equations on K_D and K_R for step-size #1 (denoted P_1). This feasible region does not overlap the feasible region defined by the stability margin constraint equations. With the

initial condition, the rigid body phase and gain margin constraints are satisfied, but the aerodynamic gain margin is not. Hence, the design method enters the inner loop, and relaxes the aerodynamic gain margin constraint until a feasible region exists for both the margin constraints and the autopilot variable constraints. This relaxation is accomplished by reducing the parameter denoted as STEP.

When STEP is unity, no relaxation exists. When STEP is reduced to 0.8, the aerodynamic gain margin constraint is relaxed enough so that a feasible region exists. When STEP is 0.8, this means that an "80% improvement" is required for the margin that is not yet satisfied. This so-called "required margin improvement" becomes very important when the optimum is exterior to the feasible region. This is the case most of the time for launch vehicle autopilot design.

As indicated on Figure 3.4, the optimum solution for the first step of the minor loop exists at Y_1. Comparing Y_1 with the nonlinear cost function and constraints shown on Figure 3.2, it is seen that stability at Y_1 is better than at the initial condition. This "improved stability" is indicated by the figure-of-merit which, as discussed in Section 3.5.1, is a linear combination of the stability margins. Note that the so-called "met-margin counter" indicates that at Y_1, there are still only two margins that are satisfied. Note that the margin counter and the figure-of-merit are formed from an actual evaluation

of the frequency response. In other words, they are not computed from the linearized cost function and the linearized constraint equations.

Hence, since Y_1 is better than the initial condition, the design process advances, using the same set of partial derivatives, and hence the same linearized cost function and margin constraint equations that were calculated at the initial condition.

As shown in Figure 3.5, the design method now increases P from P_1 to P_2. Figure 3.5 shows the feasible region defined by the autopilot variable constraint equations for P_2. An overlap exists between the feasible regions defined by the margin and autopilot constraint equations, and hence the inner loop need not be used. For P_2, the optimum solution exists at Y_2. By comparing Y_2 with the nonlinear constraint equations of Figure 3.2, the margin counter indicates that there are now three margins that are satisfied. Since improved stability has again been achieved, P is further increased from the original initial condition.

As shown in Figure 3.6, P is now increased to P_3. For this step, overlap also exists, and the optimum solution is at Y_3. By comparing Y_3 to the nonlinear constraints of Figure 3.2, it is seen that the rigid-body phase margin requirement is no longer satisfied. The margin counter indicates that only two margins are satisfied at Y_3. Hence, Y_3 is not as good as Y_2 and P must be reduced.

Figure 3.7 shows the autopilot constraint equations for P_4, where $P_2 < P_4 < P_3$. Again, the inner loop is not needed, and the optimum solution exists at Y_4. By comparing Y_4 to Figure 3.2, the margin counter shows that there are three margins that are met. But, the figure-of-merit shows that Y_2 is better than Y_4. Postulating that the difference between P_2 and P_4 is less than some convergence criterion, the algorithm stops this so-called major iteration at Y_2. The values of K_D and K_R at Y_2 become the initial condition for the next major iteration. At Y_2, the problem is relinearized. A new set of partial derivatives is computed, and a new cost function and new constraint equations are formed. As the design progresses, the linearized optimum gets closer and closer to the nonlinear or "true" optimum. As will be discussed in Section 3.5.5, convergence criteria can be used to terminate this iterative design process.

Table 3.1 summarizes the results of Figures 3.2 through 3.7. These figures have been used to demonstrate steady convergence to a local interior optimum. Section 3.5.4 will illustrate convergence to a local exterior optimum.

As a final note, Figures 3.2 through 3.7 demonstrate that this algorithm does not require that the initial condition lie within the feasible region.

3.5.4 Convergence to an Exterior Optimum

This section illustrates how the algorithm converges to an

Table 3.1. Summary of Figures 3.2 Through 3.7

Figure	Minor Loop Step Size	Constraint Relaxation?	Solution	Numbers of Margins Met	Criterion on Fig.-of-Merit	Best Solution So Far
3.3	Initial Condition	-	Y_o	2	-	I.C.
3.4	P_1	Yes (STEP=.8)	Y_1	2	$Y_1 > Y_o$	Y_1
3.5	P_2 where $P_2 > P_1$	No	Y_2	3	-	Y_2
3.6	P_3 where $P_3 > P_2$	No	Y_3	2	-	Y_2
3.7	P_4 where $P_2 < P_4 < P_3$	No	Y_4	3	$Y_2 > Y_4$	Y_2
Y_2 serves as initial condition for next major iteration since $\lvert P_4-P_2 \rvert$ satisfies convergence criteria.						

exterior or "constrained" optimum. Figure 3.8 shows a case that might exist when optimizing load relief capability since for this phase of design, the optimum solution almost always is exterior to the feasible region defined by the margin constraint equations. Figure 3.8 is a hypothetical two-dimensional case where (1) the nonlinear margin constraint might represent the so-called aerodynamic gain margin, (2) X_1 might represent the attitude error gain, and (3) X_2 might represent the so-called load relief loop gain. Figure 3.8 also shows the nonlinear constrained optimum.

In Figure 3.8, the initial condition on X_1 and X_2 is outside the feasible region. Figure 3.9 shows what the margin constraint and the nonlinear optimum might look like when the problem is linearized about the initial condition. The first step of the algorithm is to "get feasible", and Figure 3.9 will show that in so doing, the algorithm still attempts to approach the optimum.

Referring to Figure 3.9, after a series of iterations through the minor and the inner loops, the solution is shown to exist at Y_1. With the linearized margin and cost function as shown, this is the best solution this major iteration can achieve without violating the nonlinear margin constraint.

At Y_1, the problem is relinearized as shown in Figure 3.10. Figure 3.10 shows that any step in the direction of the gradient to the linearized optimum would yield an unfeasible solution. Referring to

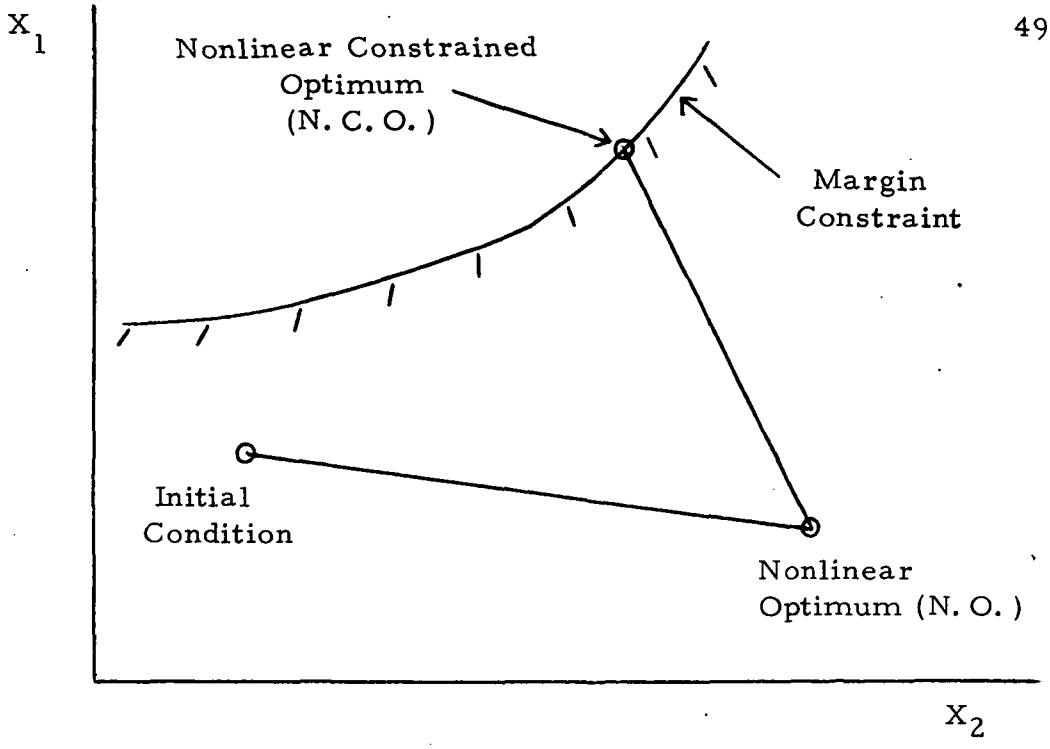


Figure 3. 8. Nonlinear Constrained Optimum

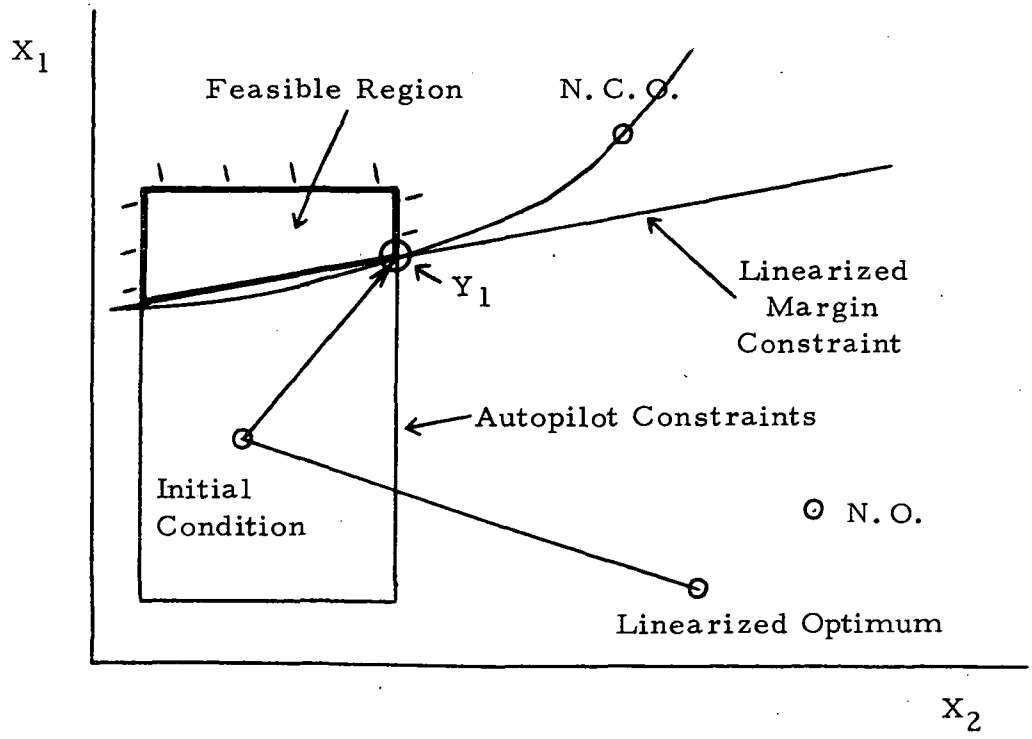


Figure 3. 9. First Major Iteration

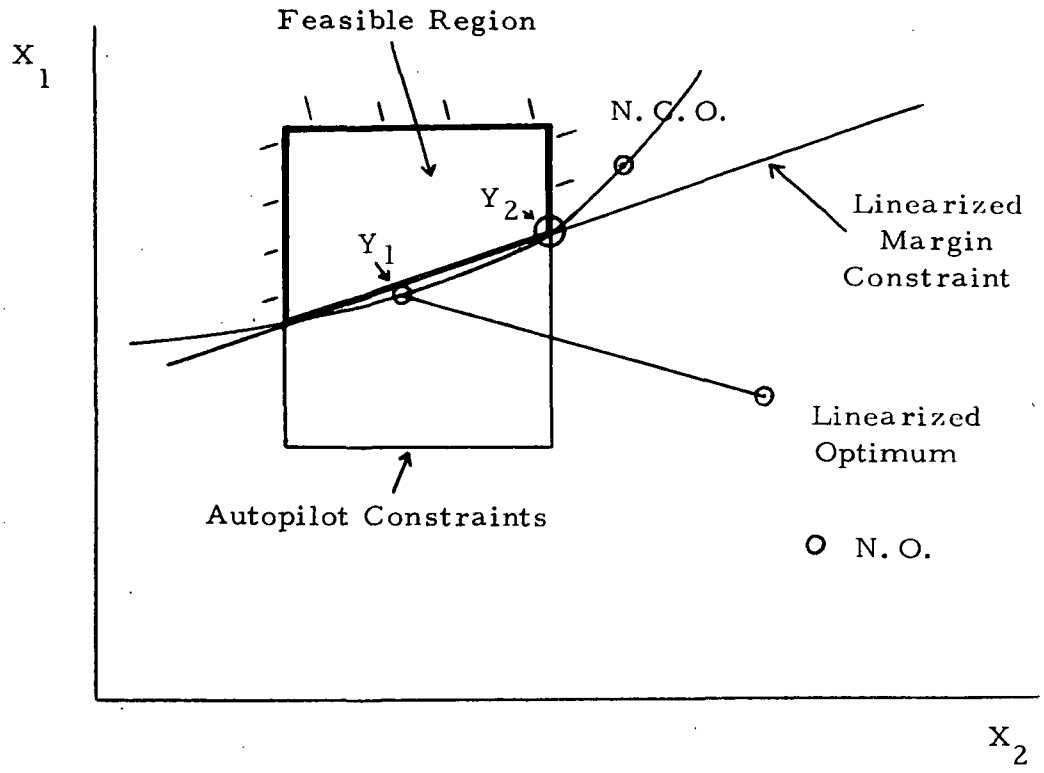


Figure 3.10. Second Major Iteration

Figure 3.10, again after a series of iterations through the minor and the inner loops, the solution exists at Y_2. With the linearized margin and cost function as shown, this is the best solution this major iteration can achieve without violating the nonlinear margin constraint. With each major iteration, the linearized constrained optimum approaches the nonlinear constrained optimum.

This section illustrates how the algorithm first "gets feasible", and then moves along or parallel to a constraint for the case of a constrained optimum. Example 6 of Chapter 4 will dramatically demonstrate this situation.

Because of the weighting factors in the cost function and figure-of-merit, situations with an exterior optimum can also exist when optimizing stability margins.

3.5.5 Termination

This section discusses the two ways in which the design process can be terminated.

The first way might be referred to as self-termination, where the algorithm finds a local optimum and can achieve no improvement over the initial autopilot for a given iteration. The so-called margin counter and the figure-of-merit define this "improvement." Recall that the counter "counts" the number of "met margins", and the figure-of-merit is a linear combination of the actual margins. The counter never allows the algorithm to lose a margin, and an iteration is con-

sidered better if the counter increases. The figure-of-merit is used to break ties in the counter. An iteration is considered no good if the figure-of-merit decreases. An iteration is considered better only if the figure-of-merit increases by a certain percentage (as specified by the user). The following discusses this.

The user may wish to "reward" a certain margin only up to a certain desired value. In other words, up to a desired value, the figure-of-merit will include the actual value of the margin. When the margin exceeds this desired value, the figure-of-merit will only include this desired value. If this happens for all margins, the figure-of-merit will not change at all from one iteration to the next, and the algorithm will have found an optimum that not only meets the requirements, but also satisfies the desired objectives. Examples 1 and 2 of Chapter 4 will illustrate cases where this happened.

The case most likely to be encountered is when not all margins exceed their desired values, and the figure-of-merit improves only slightly from one iteration to the next. The percent improvement is less than that required by the user and the design process terminates. For this case, the local optimum may yield a solution that satisfies all the design requirements, in which case the problem is solved. However, if the local optimum does not satisfy the requirements, the user must then either (1) try another initial condition for the autopilot variables, (2) add more complexity to the autopilot, and/or (3) relax some

of the design requirements and/or alter some of the design objectives.

The second way that the design may be terminated is directly by the user. He may specify termination after a certain number of major iterations or after a certain amount of computer time.

3.6 Detailed Flow Charts

This section now presents two detailed versions of the General Flow Chart that was given in Figure 3.1.

3.6.1 Overall Flow Chart

Figure 3.11 is a detailed overall flow chart of this nonlinear programming algorithm as it is applied to the problem of launch vehicle autopilot design. It shows the flow of information from the initial autopilot, all the way through the step-size optimization routines, through the termination routine, and back to the initial autopilot for the next major iteration. Some of the details of Figure 3.11 are now discussed.

In the block showing where the partial derivatives are calculated, reference is made to closed-loop roots. This illustrates the possibility of not only optimizing stability margins, but also of optimizing locations of at least the so-called dominant closed-loop roots. This also illustrates the possibility of adding a routine that could put some of the following requirements on the response of the attitude of the vehicle due to a guidance command:

(1) Attitude rise time;

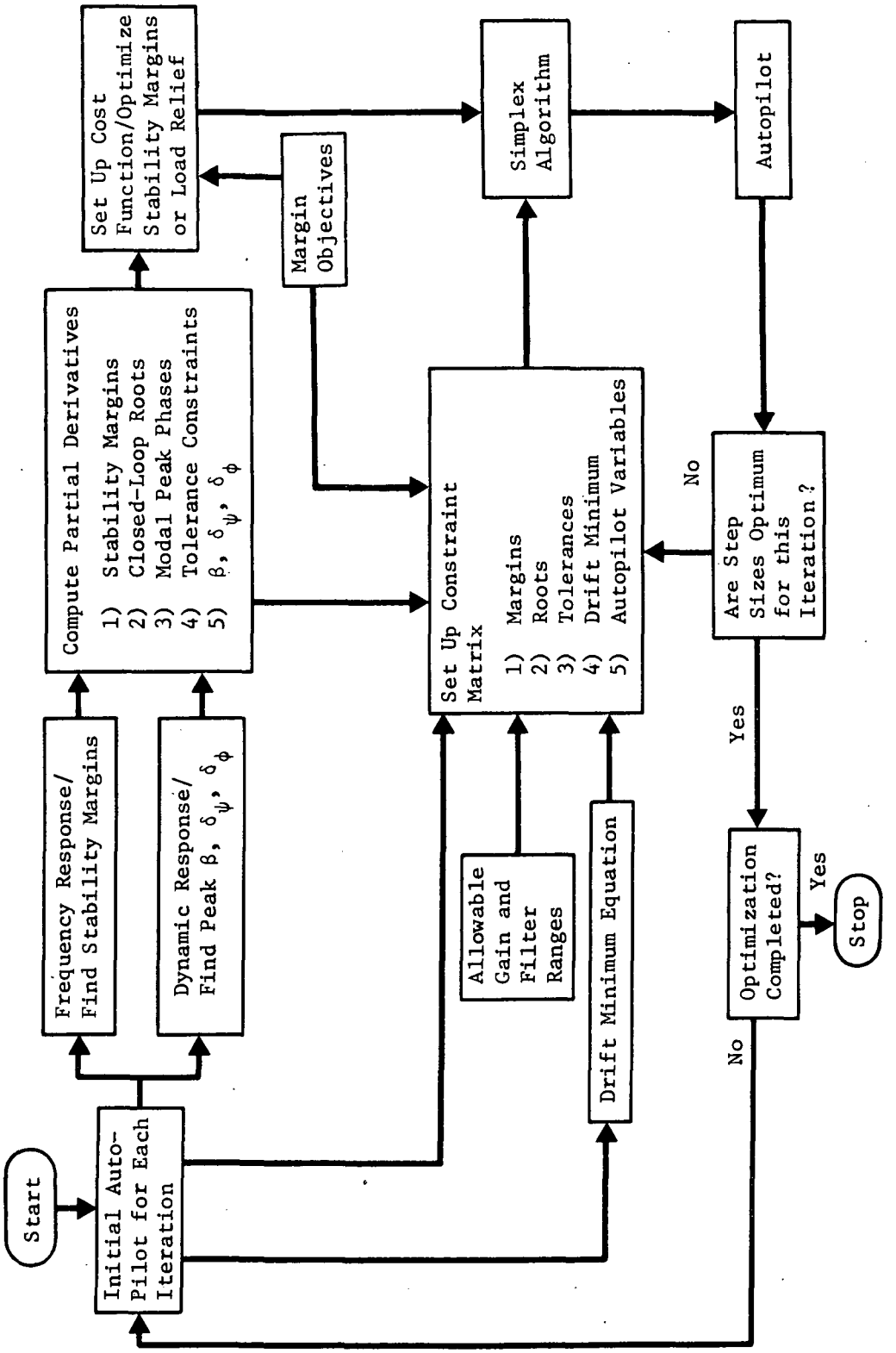


Figure 3.11. Overall Flow Chart of the Design Algorithm

(2) Attitude overshoot;

(3) Attitude settling time;

(4) Steady-state attitude error; and

(5) Peak values of the control device deflections required to follow the guidance command.

Also in the block for computing partial derivatives, reference is made to "modal peak phases". As discussed in Section 3.3.1, these sensitivities are used in the cost function in an attempt to force all bending modes to resonate near zero degrees phase.

In that same block, reference is made to so-called tolerance constraints. This refers to a routine that is used to keep the individual autopilot vectors (e.g., the attitude error vector, the rate vectors, the accelerometer loop vector, etc.) from getting very much larger than the total resultant vector at all frequencies. When the individual vectors get much larger than the resultant, vector cancellation results, and this can lead to problems when tolerances on the airframe parameters are considered.

In the block showing where the constraint matrix is set up, reference is made to the "Drift Minimum" condition. (Greensite [14] and Hoelker [17] define this condition, which basically is a steady-state relationship between the autopilot gains. This steady-state relationship results in a mix between the forces due to gravity, aerodynamics, and control deflections, that yields a zero net force per-

pendicular to the vehicle's velocity vector.) This illustrates that it is possible not only to put constraints on margins and root locations and on min-max values of the autopilot variables, but also to constrain the autopilot variables to satisfy other conditions like the Drift Minimum condition.

Figure 3.11 shows a block labeled "Simplex Algorithm". This is in reference to Dantzig's method [9] for solving linear programming problems.

Figure 3.11 also shows a block labeled "Margin Objectives". This refers to the fact that design objectives are used to form the cost function, while design requirements are used to form the constraint matrix.

Finally, Figure 3.11 shows a block that refers to the step-size optimization routine. The details within this block are shown in Figure 3.12 which is discussed in the next section. Figure 3.11 also shows a block labeled "Optimization Completed?" which refers to the termination routine.

3.6.2 Step-size Optimization Flow Chart

Figure 3.12 illustrates the step-size optimization routines (namely, the Minor Loop and the Inner Loop) and their relationship to the other routines shown in Figure 3.11. As discussed in Section 3.5, the Inner Loop is used to relax constraints in order to yield a feasible solution to the linear programming problem. The Minor Loop is used

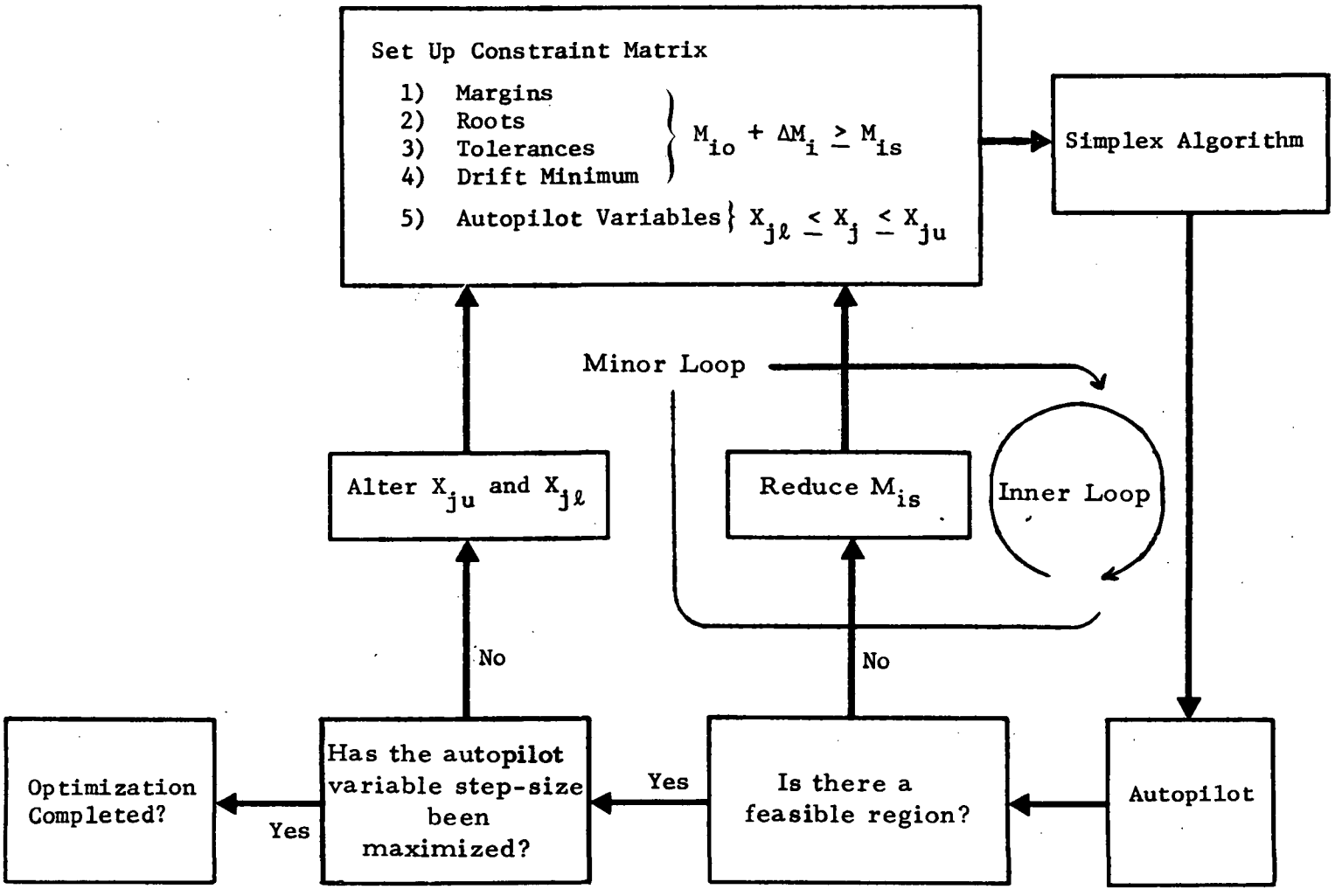


Figure 3.12. Step-size Optimization Flow Chart

to maximize the autopilot variable step-size in order to get the "maximum mileage" out of each set of partial derivatives, and in order to allow the algorithm to steadily converge to a local optimum, particularly to an interior optimum.

CHAPTER 4

RESULTS

This nonlinear programming algorithm, when applied to the problem of launch vehicle autopilot design, led to the development of a digital computer program called COEBRA. COEBRA is an acronym for Computerized Optimization of Elastic Booster Autopilots.

This chapter presents some of the results obtained from the COEBRA program. The first two examples are frequency domain design problems that were found in textbooks. The last five examples demonstrate the application of COEBRA to the booster autopilot design problem.

Each example also shows the computer time required. This is the time required on a CDC 6400/6500 computer system.

Throughout this chapter, the following shorthand notation will be used to represent compensator transfer functions (S-plane or W-plane). The transfer function given by

$$\frac{K\,(1+\tau_1 S)\left(1+\dfrac{2\zeta_1}{\omega_1}S+\dfrac{S^2}{\omega_1^2}\right)}{(1+\tau_2 S)(1+\tau_3 S)\left(1+\dfrac{2\zeta_2}{\omega_2}S+\dfrac{S^2}{\omega_2^2}\right)}$$

will be abbreviated as follows:

$$\frac{[K]\,(T1,\ \zeta 1,\ \omega 1)}{(T2,\ T3,\ \zeta 2,\ \omega 2)}$$

4.1 Example #1

Example #1 illustrates the application of COEBRA to a problem found in B. C. Kuo's textbook on sampled-data control systems [23, example 9-3, page 291]. This problem was selected in order to demonstrate the COEBRA program to those readers who are not familiar with the details of and the solutions involved in the booster autopilot design problem.

The overall problem that Kuo was illustrating was the design of a digital compensator for a sampled-data control system. The first step in Kuo's design was to compute the Z-transform of the plant (the fixed parts of the system). He then selected the loop gain to yield a certain desired velocity constant. The next step was to transform the problem to the W-plane. The block diagram of the system in the W-plane is shown in Figure 4.1. Kuo's last step (before actually implementing or realizing the final solution) was to adjust the two time constants in the compensator to achieve a phase margin greater than or equal to 50 degrees. The problem given to the COEBRA program was to adjust the compensator time constants (T1 and T2) until the following results from Kuo's compensator were matched:

(1) Phase margin $\geq$ 50 degrees

(2) Gain margin $\geq$ 12 decibels

(3) Frequency at the phase margin (ω_c) $\geq$ 0.2

Figure 4.2 compares Kuo's results with those obtained from

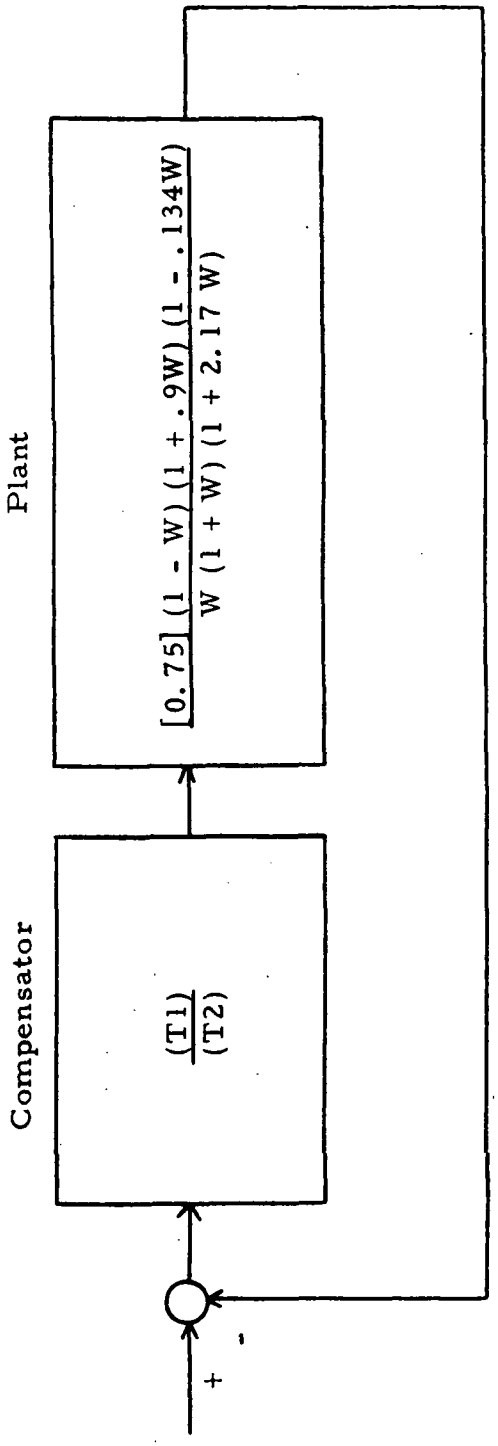


Figure 4.1. System Block Diagram for Example #1

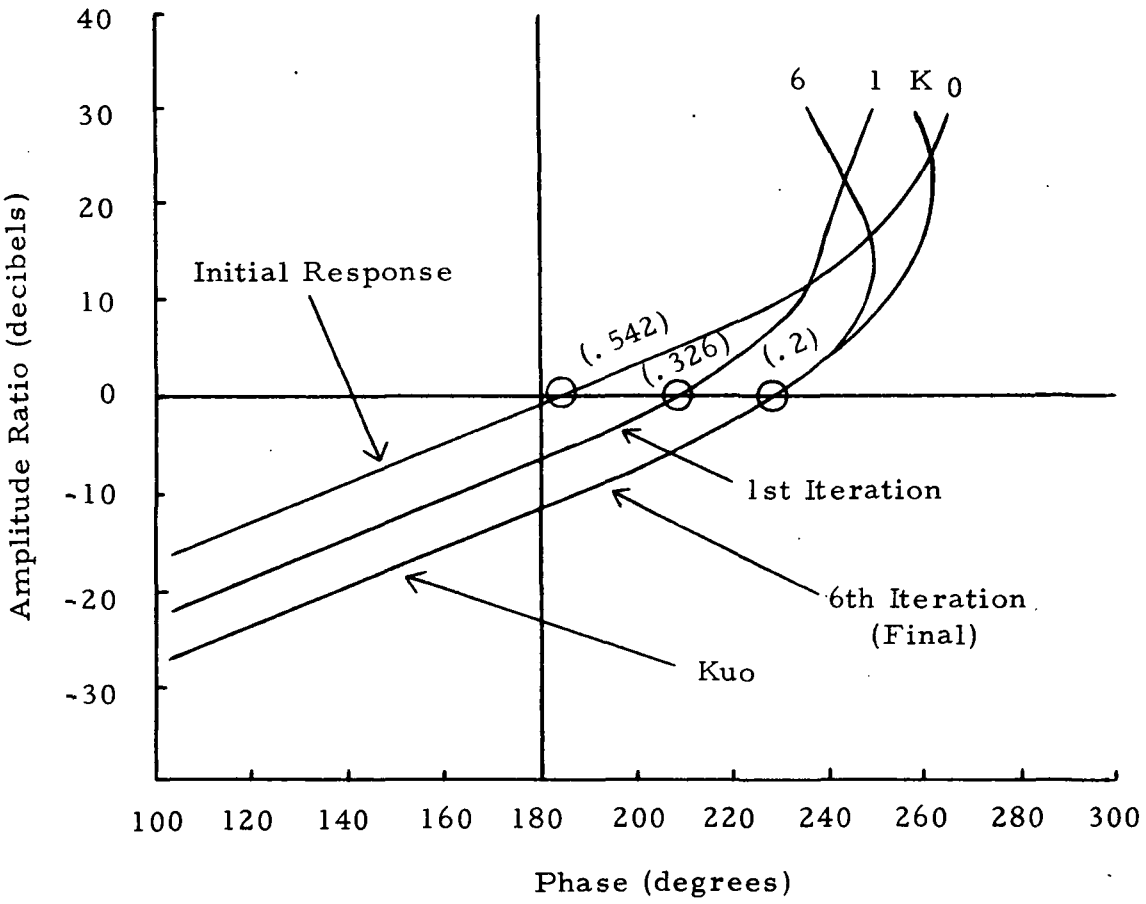


Figure 4.2. Example #1, Gain-Phase Frequency Response Plot Comparing Results of Kuo with COEBRA Run #1

the first COEBRA run. For this COEBRA run, both T1 and T2 were initialized with a value of 20, and hence the initial response shown in Figure 4.2 is identical to the response of the plant only. Note that ω_c was 0.542 for the initial response. Figure 4.2 shows the results of the first and sixth (final) major iterations of COEBRA. The sixth iteration was the final one since COEBRA was not "rewarded" for doing better than Kuo's result. In other words, recalling the discussion on termination in Section 3.5.5 of Chapter 3, the figure-of-merit was not allowed to increase once Kuo's results were matched. The results of the fifth and sixth iterations were identical, since it took COEBRA one iteration to decide that improvement was no longer possible or "permitted".

Table 4.1 summarizes the initial and final compensators, as well as the final stability margins obtained from COEBRA run #1. Note that as with Kuo, the final answer from Run 1 was a phase-lag compensator. Due to the circumstances of the problem as pointed out by Kuo, this minimum complexity (first order) compensator had to be a phase-lag model. In other words, phase-lead compensation would be ineffective.

Additional COEBRA runs were made in an attempt to "map the hill", or in other words, to see what COEBRA would do with different initial compensators. As shown in Table 4.1, Runs 2, 3 and 4 achieved essentially the same results as did Run 1 and Kuo. With the initial

Table 4.1. Example #1 Summary of Results

Run	Initial Compensator		Final Compensator		Final Margins		
	T1	T2	T1	T2	Gain Margin (db)	Phase Margin (deg)	Phase Margin Frequency
KUO	-	-	50.	100.	12.	50.	0.2
1	20.	20.	80.5	283.	12.	50.	0.2
2	50.	50.	60.4	212.	12.	50.	0.2
3	100.	100.	76.3	268.	12.	50.	0.2
4	5.	5.	67.3	237.	12.	50.	0.2
5	2.	2.	1.02	0.0245	7.	30.	0.61
6	2.	4.	83.	291.	12.	50.	0.2

compensator of Run 5, COEBRA climbed a local optimum that did not satisfy the design requirements. Run 5 automatically terminated after 14 major iterations when the margin counter and the figure-of-merit essentially ceased to increase. Note that the final answer from Run 5 was not a phase-lag compensator. Run 6 was made with the denominator time constant (T2) of Run 5 changed to a value of 4., so that the initial compensator was a phase-lag compensator. Table 4.1 shows that Run 6 achieved essentially the same answer as did Kuo.

Since a first order phase-lag compensator is the minimum-complexity compensator that can solve this problem, it is not difficult to understand why COEBRA could not converge to a final solution from every initial condition. This points out that the difficulty of any problem is dictated more by the degrees of freedom in the compensator than by the complexity or order of the plant.

Table 4.2 summarizes the computer time required to make COEBRA runs 1 through 6.

4.2 Example #2

Example #2 illustrates the application of COEBRA to a problem found in Schaum's Outline Series on feedback control systems [DiStefano, 11, problem 16-10, page 309]. This problem was selected for the same reason as Example #1.

The system block diagram for Example #2 is shown in Figure 4.3. For this example, the design is performed in the S-plane. After

Table 4.2. Example #1 Computer Time

Run	Number of Iterations		Computer Time (sec)
	Major	Minor	
1	6	39	118
2	5	27	89
3	5	21	73
4	7	51	154
5	14	72	252
6	8	53	159

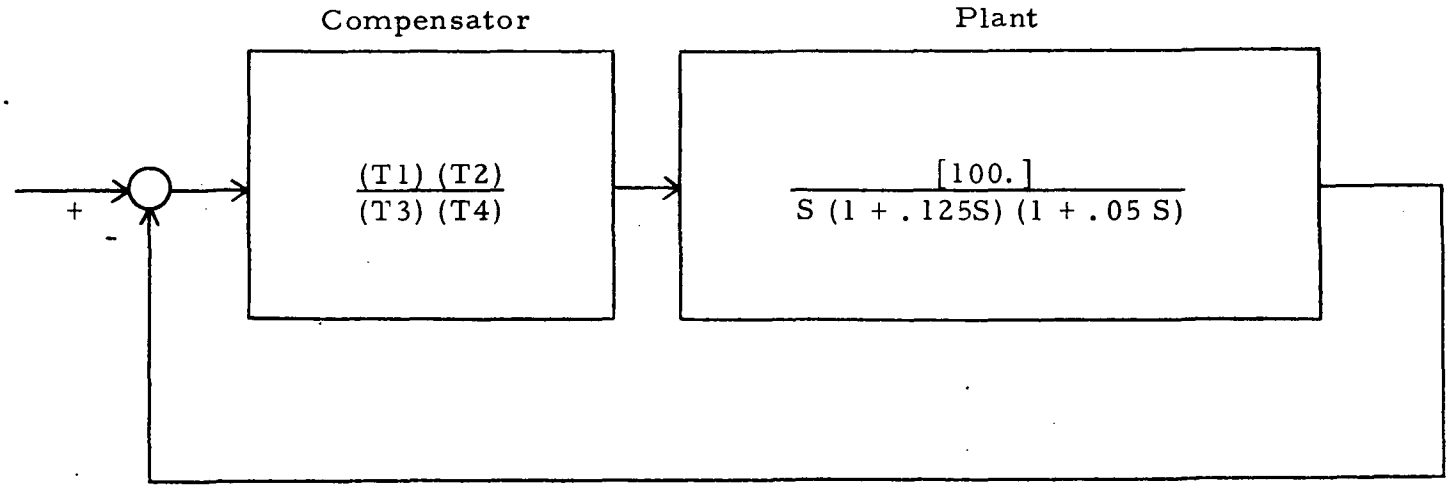


Figure 4.3. System Block Diagram for Example #2

DiStefano had selected the Bode gain to yield a certain velocity constant, he then adjusted the four time constants in the compensator to meet the following design specifications:

(1) Phase margin $\geq$ 45 degrees

(2) Gain margin $\geq$ 10 decibels

(3) Frequency at the phase margin (ω_c) $\geq$ 10 rad/sec

The problem given to the COEBRA program was to adjust the four compensator time constants until the above three design requirements were satisfied.

Table 4.3 compares DiStefano's results with those obtained from the first COEBRA run. With all the time constants in the compensator initialized to unity, COEBRA, in six major iterations, climbed to a local optimum that did not meet the design requirements. DiStefano showed that the minimum-complexity compensator that is required to solve this problem, is a lag-lead compensator, and the final answer for the unsuccessful Run 1 is <u>not</u> a lag-lead compensator. COEBRA terminated after six iterations when the margin counter and the figure-of-merit ceased to improve.

COEBRA was reinitialized to the compensator shown for Run 2 in Table 4.3. As can be seen from the table, Run 2 achieved all the design objectives. It did so with a lag-lead compensator.

Figure 4.4 compares DiStefano's results with those obtained from Run 2. Since the numerator and denominator of the initial com-

Table 4.3. Example #2 Summary of Results

Run	Initial Compensator				Final Compensator				Final Margins		
	T1	T2	T3	T4	T1	T2	T3	T4	Gain Margin (db)	Phase Margin (deg)	Phase Margin Freq.(rps)
DiStefano	-	-	-	-	1.	.2	10.	.02	14.	52.	12.
1	1.	1.	1.	1.	9.91	7.86	20.9	20.9	4.	11.4	10.
2	10.	.2	10.	.2	1.48	1.35	74.	.027	12.	57.	14.3
3	1.5	.7	1.5	.7	1.055	.718	3.02	1.41	1.7	4.6	10.
4	5.	.4	5.	.4	.83	.82	30.6	.0654	11.	53.	10.4

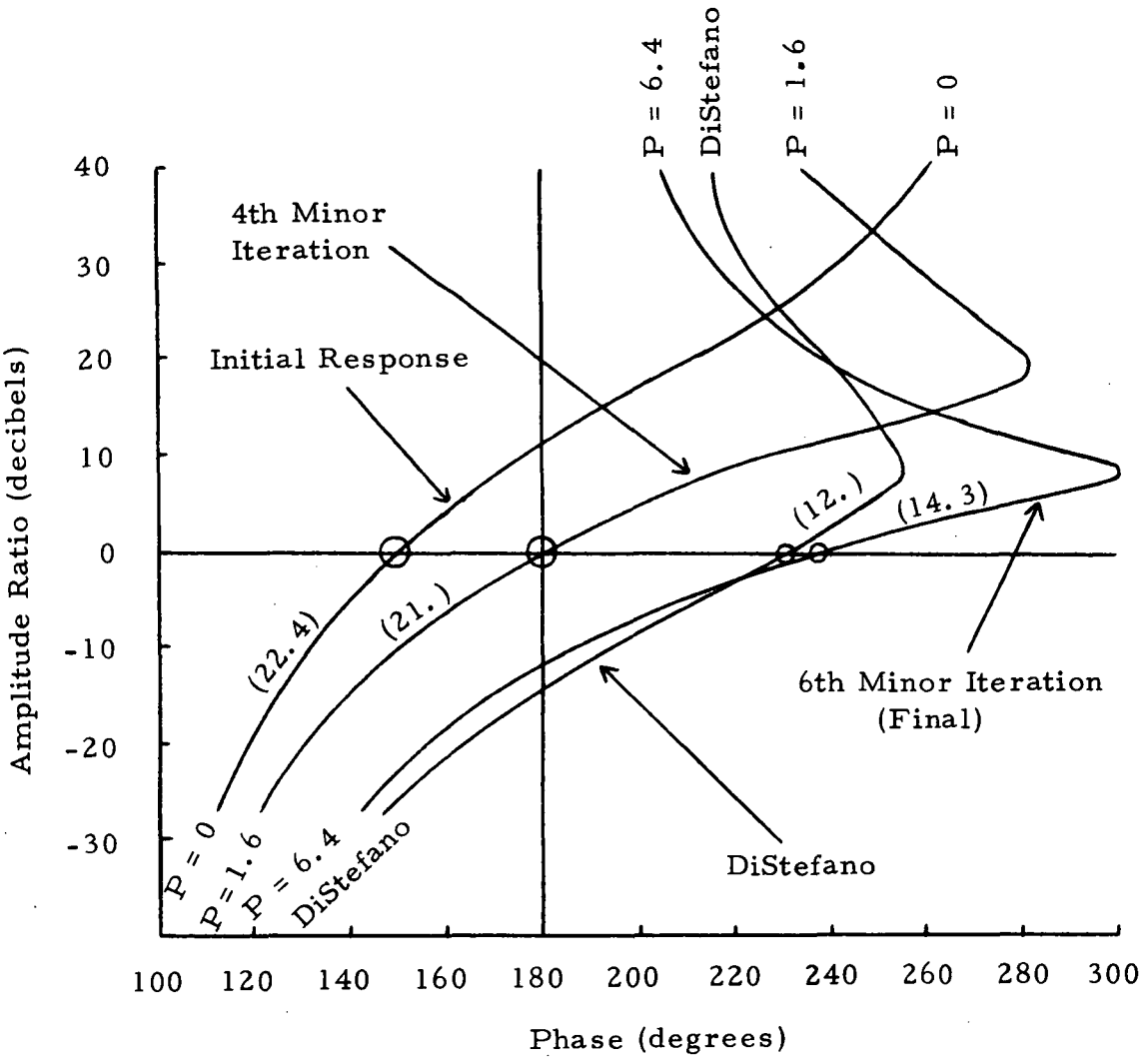


Figure 4.4. Example #2, Gain-Phase Frequency Response Plot Comparing Results of DiStefano with COEBRA Run #2

pensator for Run 2 are identical, the initial response shown in Figure 4.4 is identical to the response of the plant only. As can be seen, the system with a unity compensator (COEBRA's initial compensator) is unstable. Note that ω_c for the initial response is 22.4 rad/sec. Figure 4.4 shows the results of the fourth and sixth minor iterations in the first major iteration of Run 2. The results of the sixth minor iteration in the first major iteration satisfy all the design requirements, and these results were the best COEBRA was "allowed" to achieve. As with Example #1, the reason for this was that COEBRA was only "rewarded" up to the design requirements. In other words, recalling the discussion on Termination in Section 3.5.5 of Chapter 3, the figure-of-merit was not allowed to increase once DiStefano's results were matched. Run 2 ran for two major iterations since it took COEBRA one iteration to decide that further improvement was not allowed.

Two more COEBRA runs were made on this problem. Their initial compensators were chosen to "lie between" the initial compensators of Run 1 and Run 2. From Table 4.3, it is seen that the initial compensator for Run 3 was "close" to that for Run 1. As with Run 1, Run 3 climbed a local optimum that did not yield a feasible solution. Run 3 went four major iterations before the counter and figure-of-merit indicated that no further improvement was possible. The final answer from Run 3 was not a lag-lead compensator.

Run 4 was made with an initial compensator that was "between"

the initial compensators for Runs 2 and 3. As can be seen from Table 4.3, Run 4 achieved satisfactory results with a lag-lead compensator.

Since a second order lag-lead compensator is the minimum-complexity compensator that can solve this problem, it is not difficult to understand why COEBRA could not converge to a final solution from every initial condition. As with Example #1, this points out again that the difficulty of any problem is dictated more by the order of the compensator than by the order of the plant. In most cases, the "optimum hill" broadens and smoothes out as the order or complexity of the compensator increases.

Table 4.4 summarizes the computer time required to make COEBRA runs 1 through 4.

4.3 Example #3

Example #3 illustrates the application of COEBRA to a single-time-point autopilot design problem where the initial autopilot was so poor that it resulted in a rigid-body instability. The objective of the COEBRA run was not only to stabilize the system, but also to optimize all stability margins.

The airframe (or system to be controlled) included rigid-body dynamics and eight structural bending modes. The block diagram of the airframe/autopilot system is shown in Figure 4.5. Since this is a so-called analog autopilot, the design is performed in the S-plane.

Table 4.4. Example #2 Computer Time

Run	Number of Iterations		Computer Time (sec)
	Major	Minor	
1	6	37	148
2	2	17	63
3	4	17	81
4	3	21	83

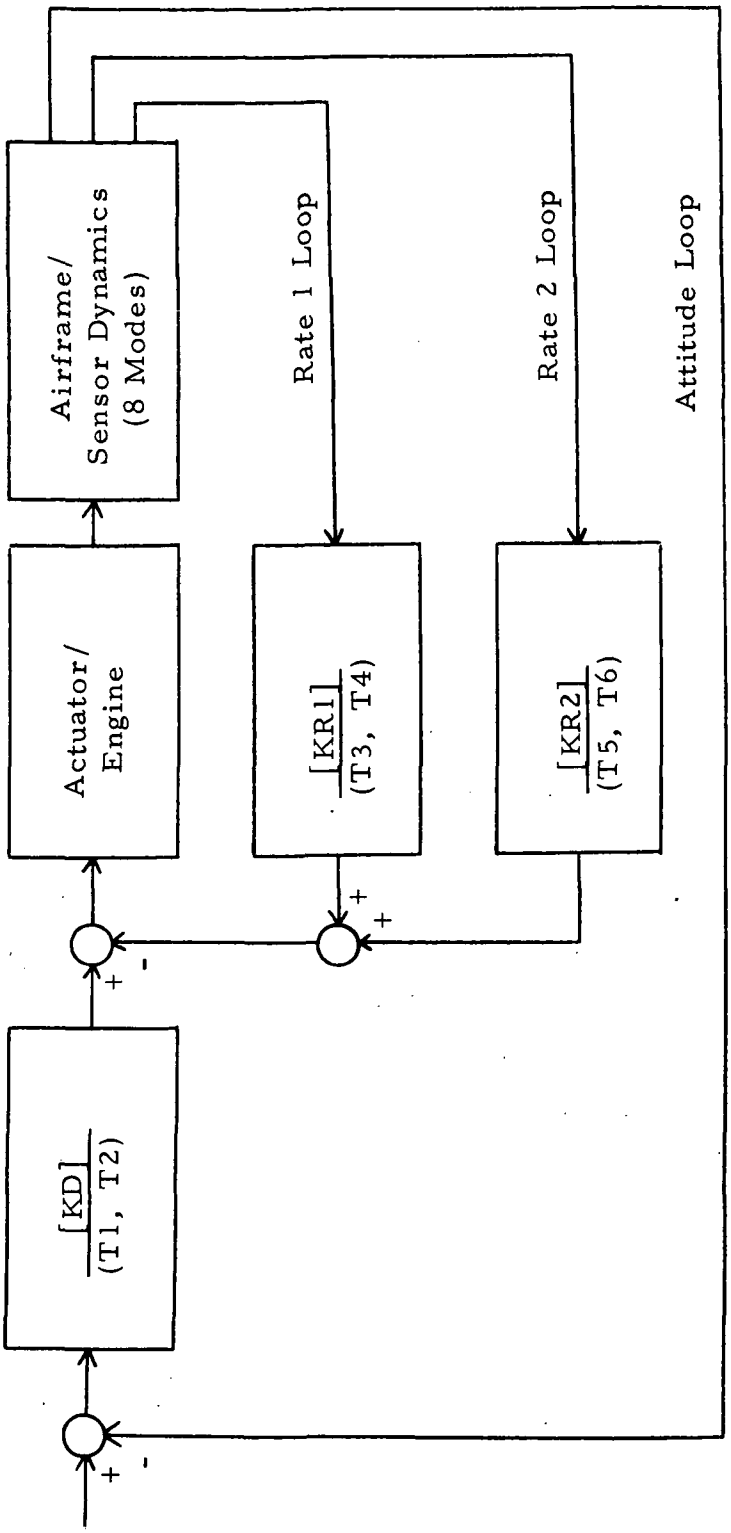


Figure 4.5. System Block Diagram for Example #3

Figure 4.5 shows the attitude loop with a gain and two filters, and two rate loops (for bending mode stability [Harris, 15]), each with a gain and two filters. The problem given to COEBRA was to adjust these nine autopilot parameters until all stability margins were optimized.

Figure 4.6 shows the open-loop frequency response resulting from the initial autopilot. This figure shows that the initial autopilot did result in a rigid-body instability. The resonances of the eight structural bending modes are indicated in Figure 4.6, which also shows that the rigid-body phase margin frequency (ω_c) was 4.62 rad. per second.

Arrows around the critical point in Figure 4.6 illustrate the required rigid-body and first mode stability margins. It was also required that ω_c be greater than 2. rad/sec and that modes 2 through 8 be gain stabilized with their peaks resonating below "-10" decibels.

Figure 4.7 shows the frequency response after the first major iteration. The system is now stable, with ω_c = 2.07 rad/sec.

Figure 4.8 shows the frequency response that resulted from the third and final iteration. COEBRA self-terminated after all design requirements were met, and after the margin counter and figure-of-merit ceased to significantly improve following the second major iteration. In other words, the results of the second and third iterations were identical since it took COEBRA one iteration to determine that

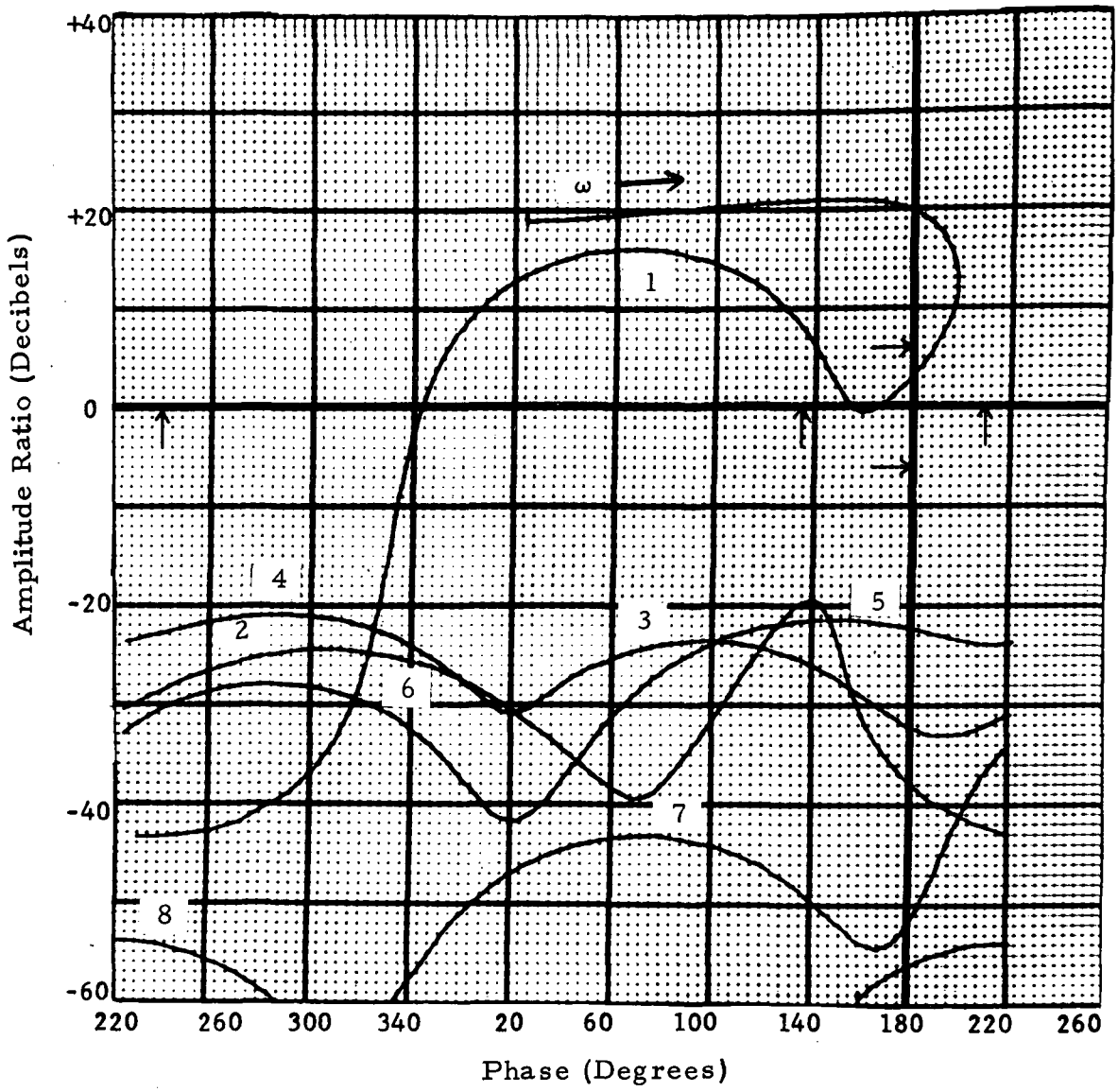


Figure 4.6. Example #3, Gain-Phase Frequency Response Plot Resulting From Initial Autopilot (ω_c = 4.62 rps)

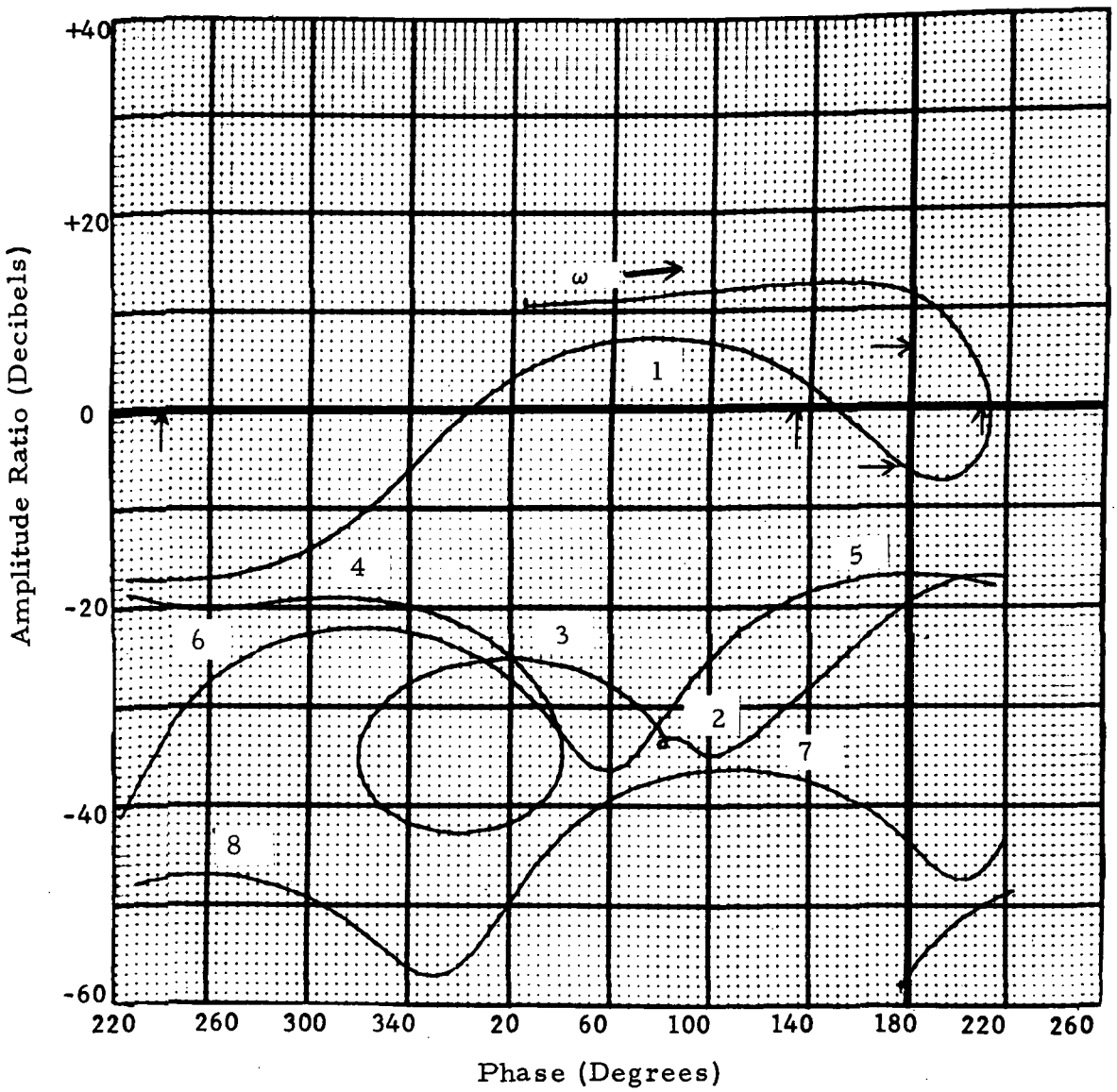


Figure 4.7. Example #3, Gain-Phase Frequency Response Plot Resulting From First Major Iteration (ω_c = 2.07 rps)

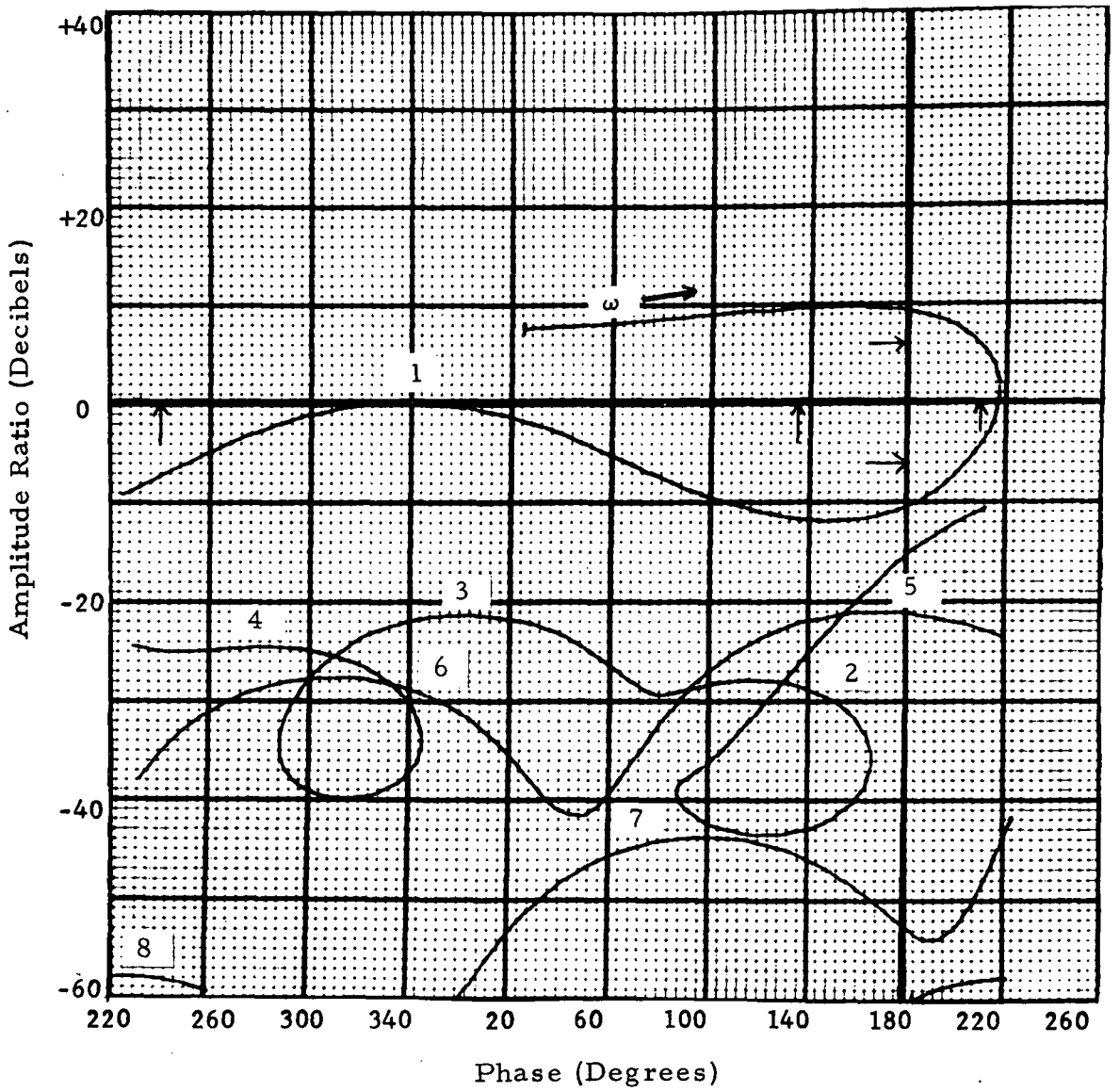


Figure 4.8. Example #3, Gain-Phase Frequency Response Plot Resulting From Third (Final) Major Iteration (ω_c = 2.03 rps)

design improvement was no longer possible. Further improvement would have been rewarded, but COEBRA was unable to achieve it. The margins that prohibited further improvement were ω_c and the phase margin on the "backside" of the first mode.

Table 4. 5 summarizes the results of this example. It shows the values of all nine parameters for both the initial and the final autopilots. Table 4. 5 shows that a satisfactory design was achieved in 493 seconds of computer time.

The following is a discussion of how COEBRA presently treats the requirement on the dominant rotational rigid-body closed-loop roots. Up to the present, the time domain response due to guidance commands of a large aerodynamically unstable flexible launch vehicle, has not been too critical. The major concern has been with stability under tolerances and with structural bending moment loads. The main reasons for specifying dominant closed-loop root locations have been to (1) keep the autopilot frequencies sufficiently separated from the guidance loop frequencies for stability purposes, and (2) merely provide "somewhat adequate" response to guidance commands. Historically, it has been found that if the rigid-body phase margin is greater than a certain value, and the phase margin frequency is greater than a certain value, then the rigid-body rotational closed-loop roots will be sufficiently damped at a high enough frequency. For example, on launch vehicles like the one represented in this

Table 4.5. Example #3 Summary of Results

	KD	T1	T2	KR1	T3	T4	KR2	T5	T6
Initial Autopilot	4.	.0333	.0333	1.6	.2	.2	.8	.2	.2
Final Autopilot	1.1	.018	.018	.86	.108	.108	.22	.108	.108

- Objective: Maximum Margins
- Single Time Point, 27th Order System
- 3 Major Iterations and 19 Minor Iterations
- Computer Time: 493 sec or 164 sec per Major Iteration
- Note: Initial Autopilot Yielded Rigid-Body Instability

example problem, if the phase margin is greater than 30 degrees, with a frequency greater than 2.0 radians per second, then it is almost certain that the rotational closed-loop roots will have a frequency greater than 1.5 radians per second and a damping ratio greater than 0.30. The M circles for unity feedback systems tend to indicate why this has been. Hence, rather than finding the actual roots, COEBRA treats the requirement on the rotational closed-loop roots by putting minimum allowed values on the rigid-body phase margin and its frequency (ω_c). This approach was taken in order to avoid the computer time required to find the actual closed-loop roots.

It is recognized that the rigid-body response of a launch vehicle is comprised of a so-called first-order drift root as well as the second-order rotational roots [Greensite, 14, and Harris, 15]. Hence, since the rigid-body response is third-order, the location of the rotational roots alone is not sufficient to ensure adequate response to guidance commands. However, since most launch vehicles are aerodynamically unstable, the instability of the vehicle generally serves to keep the attitude gain high enough, and the flexibility of the vehicle generally serves to keep the rate gain low enough, so that the rotational roots dominate the drift roots. In this way, the location of the rotational roots themselves can be used to indicate response to guidance commands. For this example problem, where the final autopilot yielded a phase margin of 38. degrees at a frequency of 2.03

radians per second, the closed-loop rotational roots had an effective damping ratio of 0.68 and an undamped natural frequency of 2.9 radians per second. This satisfied the design requirements, and a transient response showed that these roots dominated the drift root which had a time constant of 11.5 seconds.

4.4 Example #4

Example #4 illustrates the application of COEBRA to the same airframe that was used in Example #3, but this time the initial autopilot was so poor that it resulted in a first-mode instability. As with Example #3, the objective of the COEBRA run was not only to stabilize the system, but also to optimize all stability margins.

The block diagram of the airframe/autopilot system is the same as that of Example #3, and is shown in Figure 4.5. The problem given to COEBRA was to adjust the nine S-plane autopilot parameters until all stability margins were optimized.

Figure 4.9 shows the open-loop frequency response resulting from the initial autopilot. This figure shows that the initial autopilot did result in a first-mode instability. The resonances of the eight structural bending modes are indicated in Figure 4.9, which also shows that the rigid-body phase margin frequency (ω_c) was 1.17 rad. per second.

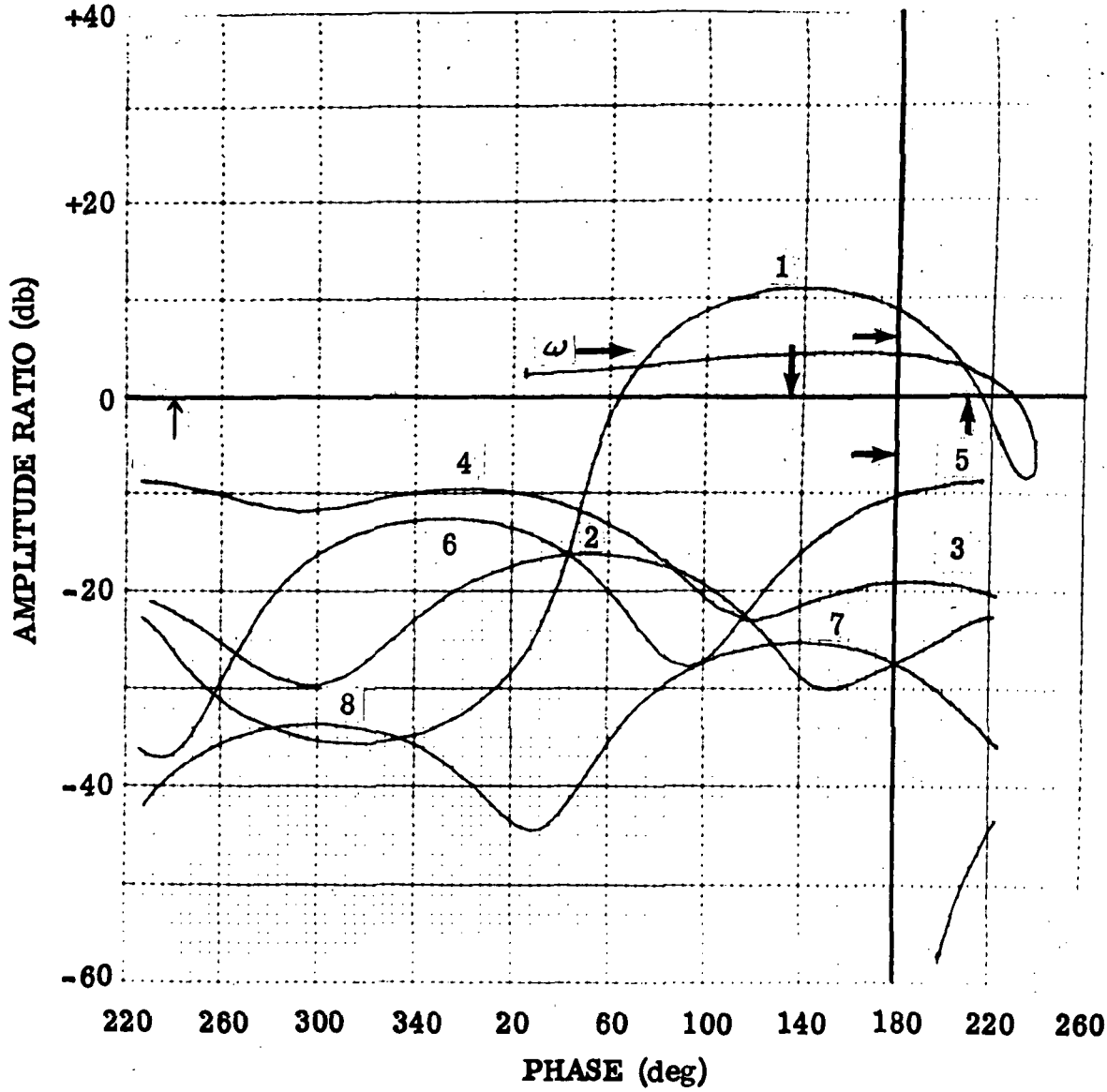


Figure 4.9. Example #4, Gain-Phase Frequency Response Plot Resulting From Initial Autopilot (ω_c = 1.17 rps)

As with Example #3, the arrows around the critical point in Figure 4.9 illustrate the required rigid-body and first mode stability margins. It was also required that ω_c be greater than 2.0 rad/sec, and that modes 2 through 8 be gain stabilized with their peaks resonating below "-10" decibels. Figure 4.9 shows that with the initial autopilot, the fourth and fifth modes exceed this requirement.

Figure 4.10 shows the frequency response after the second major iteration. The system is now stable, with ω_c = 1.3 rad/sec.

Figure 4.11 shows the frequency response that resulted from the fifth and final major iteration. COEBRA self-terminated after all design requirements were met, and after the margin counter and figure-of-merit ceased to significantly improve following the fourth major iteration. In other words, the results of the fourth and fifth iterations were identical since it took COEBRA one iteration to determine that design improvement was no longer possible. Further improvement would have been rewarded, but COEBRA was unable to achieve it. The margins that prohibited further improvement were ω_c, the phase margin on the "backside" of the first mode, and the modal peaks of the third and fifth modes.

Table 4.6 summarizes the results of this example. It shows the values of all nine parameters for both the initial and the final autopilots. Comparing these values with those obtained from Example #3, it is seen that COEBRA "climbed" to a different local optimum than it

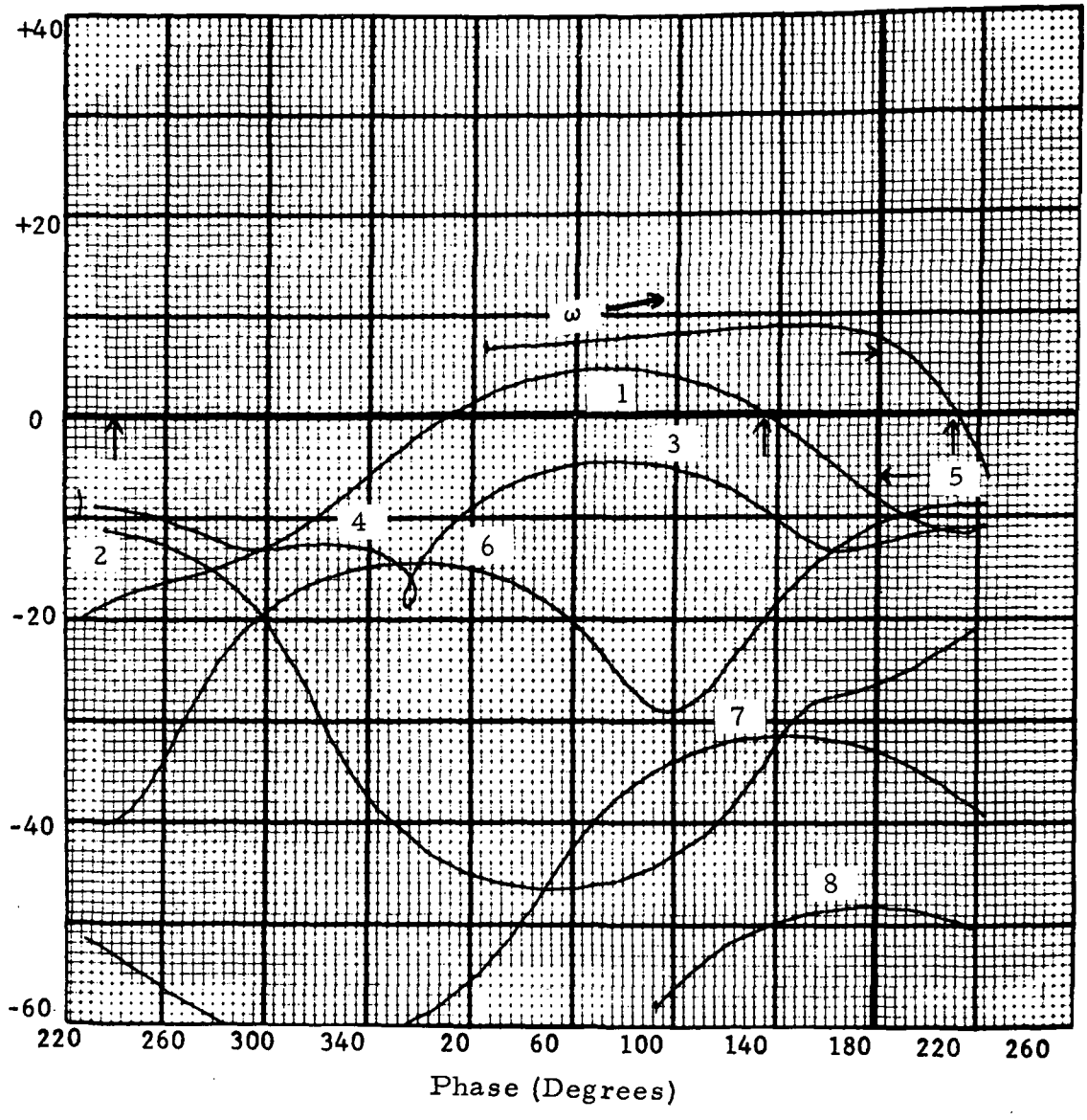


Figure 4.10. Example #4, Gain-Phase Frequency Response Plot Resulting From Second Major Iteration (ω_c = 1.3 rps)

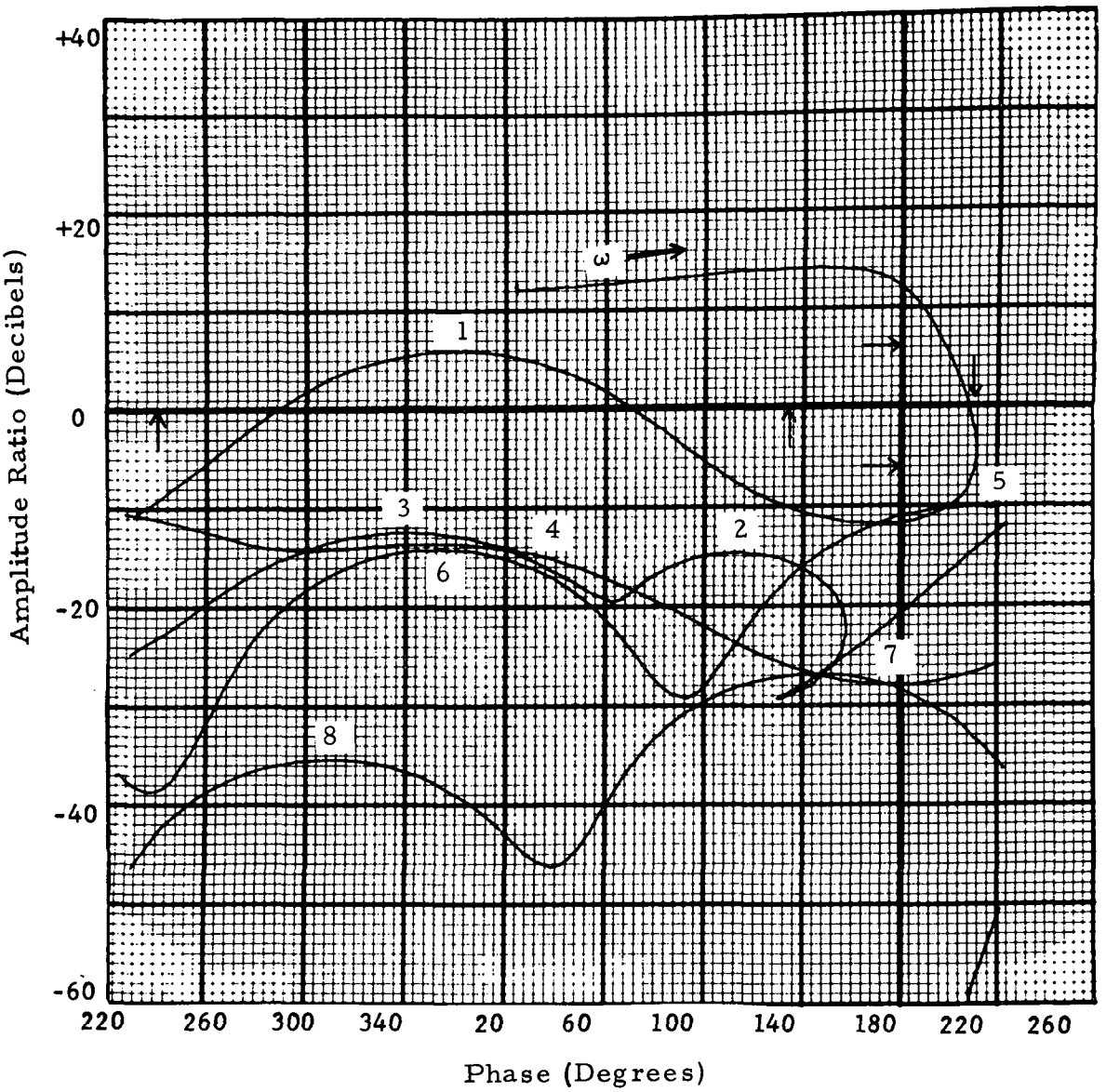


Figure 4.11. Example #4, Gain-Phase Frequency Response Plot Resulting From Fifth (Final) Major Iteration (ω_c = 2.0 rps)

Table 4.6. Example #4 Summary of Results

	KD	T1	T2	KR1	T3	T4	KR2	T5	T6
Initial Autopilot	.6	.0333	.0333	.3	.0333	.0333	.5	.0333	.0333
Final Autopilot	1.83	.093	.061	.7	.053	.053	.32	.039	.021

- Objective: Maximize Margins
- Single Time Point, Same 27th Order System as Example #3
- 5 Major Iterations and 29 Minor Iterations
- Computer Time: 975 sec or 195 sec per Major Iteration
- Note: Initial Autopilot Yielded First Mode Instability

did for Example #3. However, some features of the two results are similar. For both, the attitude gain (KD) is about 1.5, the total rate gain (KR1+KR2) is about unity, and KR1 is greater than KR2 in order to "center" the first mode around zero degrees.

Table 4.6 shows that a satisfactory design was achieved in 975 seconds of computer time.

4.5 Example #5

Example #5 illustrates the application of COEBRA to a three-time-point autopilot design problem, where the objective was to optimize structural bending moment load relief capability. COEBRA was initialized with an autopilot that had previously been designed by engineers. The reason for this COEBRA run was to determine if design improvement could be achieved. Design improvement was defined as an autopilot that had more load relief capability, but still met satisfactory stability margins.

The example deals with the max-$\bar{q}$ portion of flight where aerodynamic loads are critical. The three vehicle states that are designed together are: (1) the time at which the load relief loop (the accelerometer feedback loop) is switched in; (2) the max-$\bar{q}$ time point; and (3) the time at which the load relief loop is switched out. The airframe consists of rigid-body dynamics plus three bending modes at each time point.

Figure 4.12 shows the block diagram of the airframe/autopilot

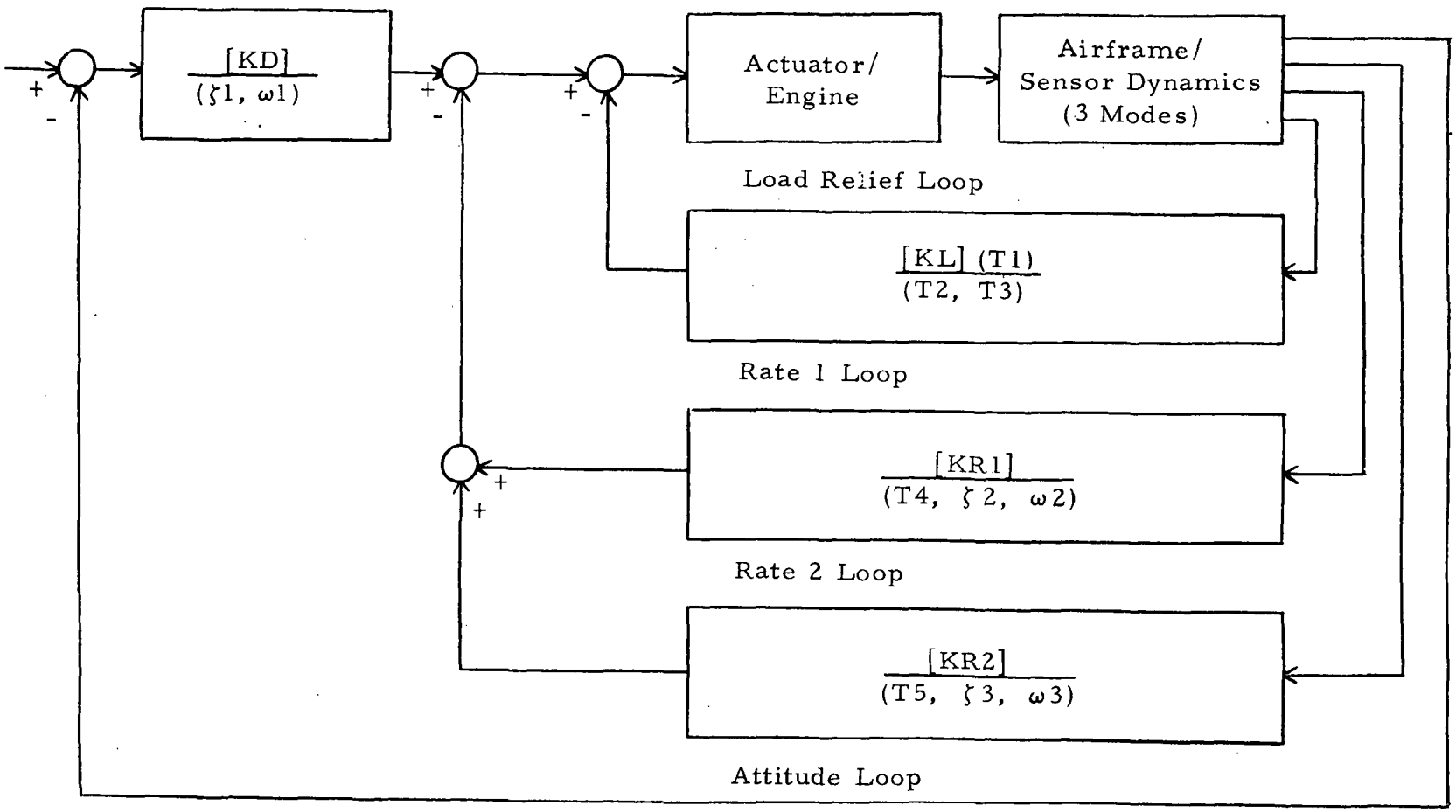


Figure 4.12. System Block Diagram for Example #5

system. In addition to the attitude loop and two rate loops, Figure 4.12 shows the so-called load relief loop. This is a feedback loop on a lateral body-mounted accelerometer signal, and Harris [15] discusses how this loop is used to reduce the angle of attack and control deflections (hence bending moment loads) in the presence of the wind.

COEBRA is allowed to vary the gains and filters shown in Figure 4.12. This is a digital autopilot design problem, and hence, these gains and filters are defined in the W-plane. Of course, when the design is complete, these gains and filters will be transformed to the Z-plane where they will be mechanized as coefficients in difference equations.

Figure 4.12 shows that, at each time point, 15 autopilot parameters can be varied. Since this is a digital autopilot, the four gains (KD, KL, KR1 and KR2) can be different at each of the three time points. The 11 filter network values, though they may be varied, must have the same values at all three time points.

Figure 4.13 shows the open-loop frequency response plot that results at the max-$\bar{q}$ time point from the engineer's final autopilot (initial autopilot for the COEBRA run). This figure, as well as the frequency responses at the other two time points (not shown), show that all margin requirements are satisfied. When a six-degree-of-freedom (6 DOF) trajectory was run using this "final" autopilot, the load relief indicator (which is a product of the dynamic pressure times

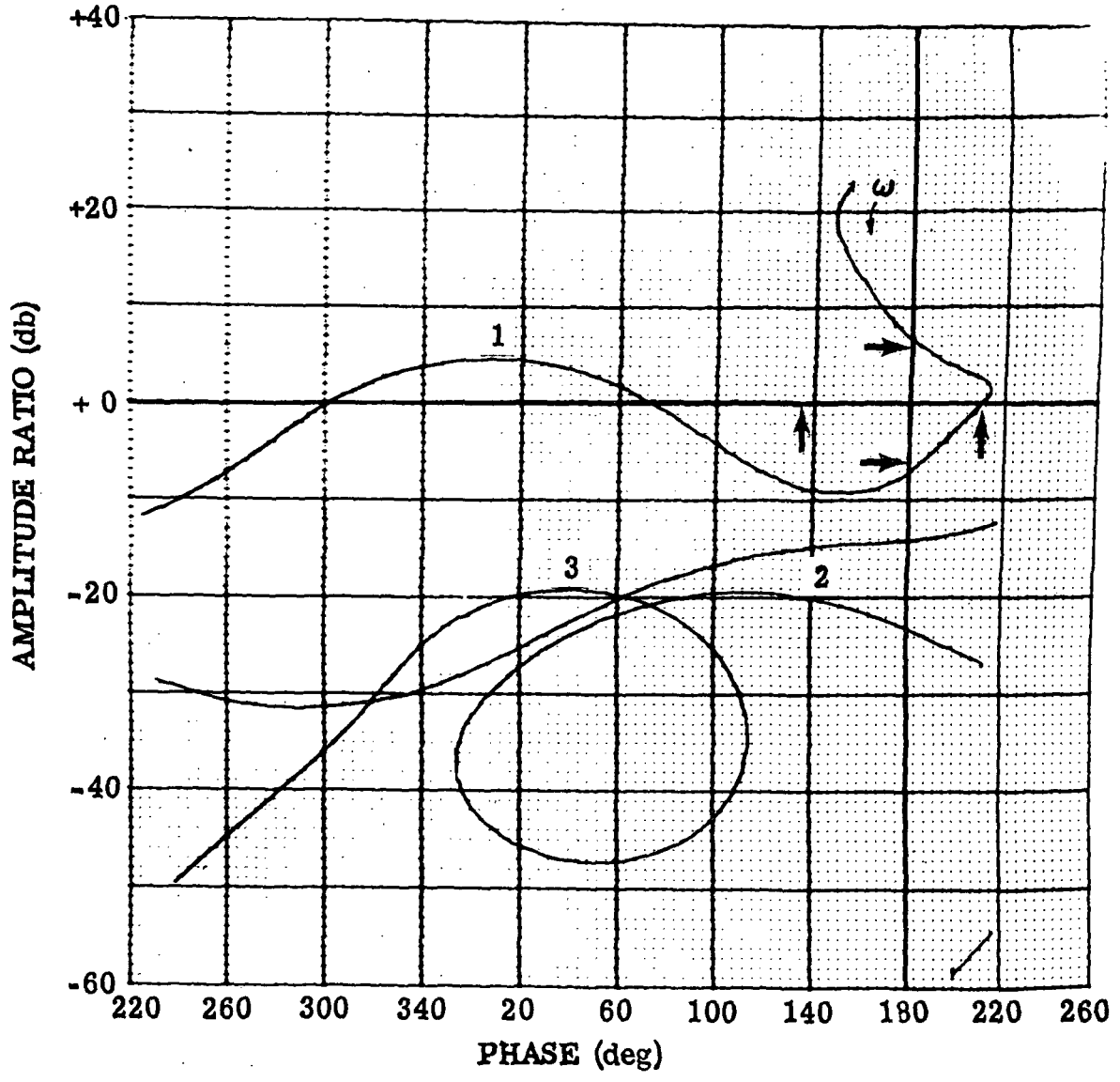


Figure 4.13. Example #5, Gain-Phase Frequency Response Plot Resulting From Initial (Engineer's Final) Autopilot at Max-$\bar{q}$

the angle of attack and is indicated as $q\alpha$) was 4908 pounds per square foot.

Figure 4.14 shows the frequency response plot that resulted at the max-$\overline{q}$ time point from the third and final iteration of the COEBRA run. Stability margin requirements are met at this time point, as well as at the other two time points. For this final COEBRA autopilot, a 6 DOF trajectory simulation showed that $q\alpha$ had been reduced to 4765 pounds per square foot.

Figure 4.14 shows that the stability margins from COEBRA's final autopilot, though satisfactory, are less than those from the engineer's final autopilot (Figure 4.13). This demonstrates the tradeoff that does exist between stability and load relief.

The conclusion of this example is that, starting from the engineer's final autopilot, COEBRA was able to achieve an improved design by adjusting the values of gains and filters within an engineer's established configuration. It is noted that the COEBRA improvement in load relief did not result because the engineer was incapable, but rather because he was not required to obtain more load reduction.

Table 4.7 summarizes the results obtained from this example. It shows that a satisfactory result was obtained after 21.3 minutes of computer time, or 7.1 minutes per iteration.

One final note is mentioned at this time. Another COEBRA run was made on this problem, beginning with the same initial auto-

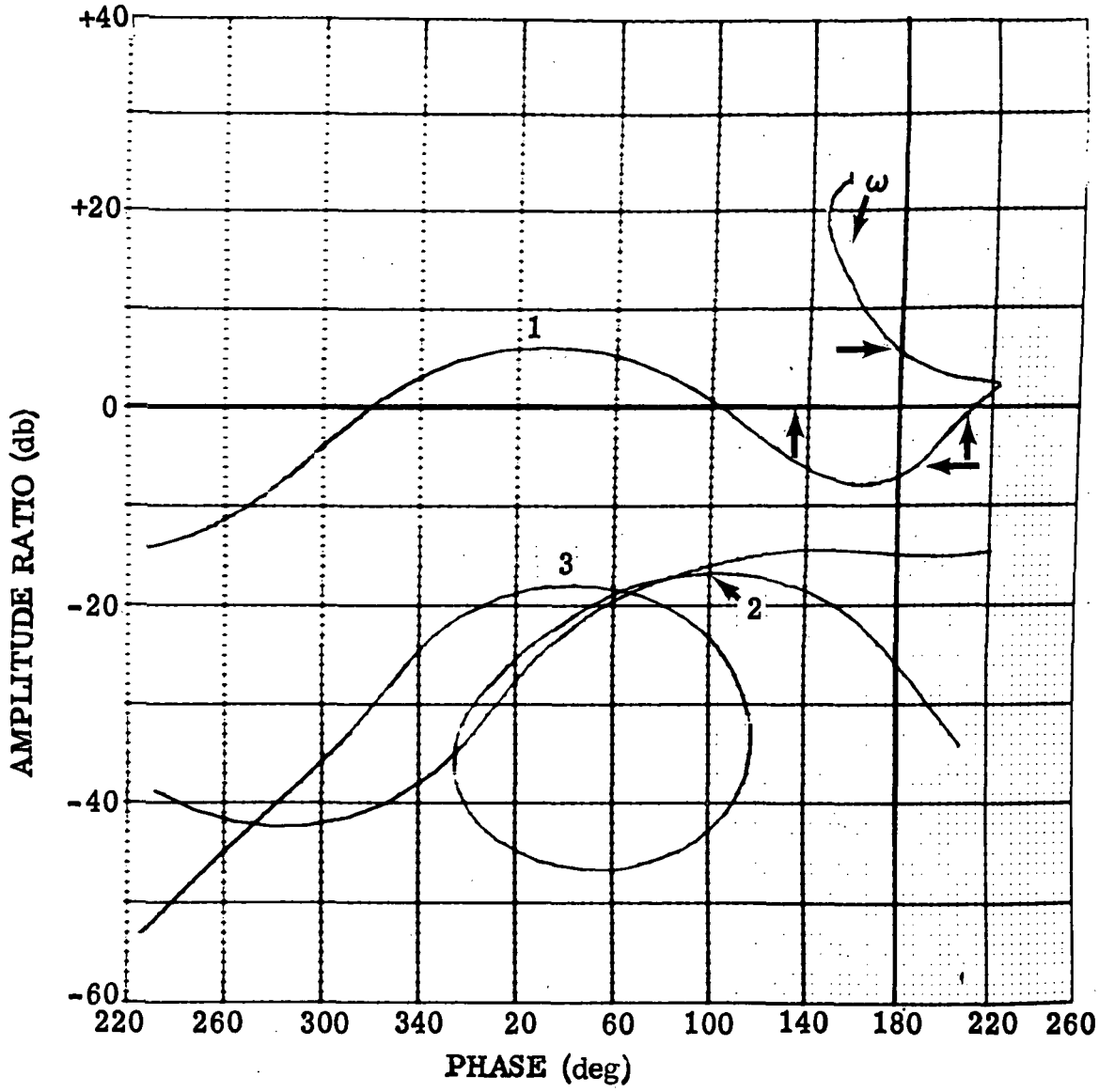


Figure 4.14. Example #5, Gain-Phase Frequency Response Plot Resulting From Third (Final) Iteration at Max-$\bar{q}$

Table 4.7. Example #5 Summary of Results

- Objective: Maximize Load Relief Capability

Autopilot	Stability Margins	q_a
COEBRA Initial (Engineer's Final)	Satisfactory	4908
COEBRA Final	Satisfactory	4765

- System Order:
  - 28th Order at Each of the Three Time Points.
  - 15 Autopilot Variables at Each Time Point (4 gains can have different values at each, 11 filters must have same values at each).
- 3 Major Iterations
- Computer Time: 21.3 Minutes or 7.1 Minutes per Iteration.

pilot, but with the objective changed to maximizing stability margins instead of load relief capability. The result of this second run was a design with improved stability margins, but with reduced load relief capability. Computer time for this run was 2.3 minutes per iteration.

4.6 Example #6

Example #5 demonstrated the effectiveness of the structural load relief optimization phase of COEBRA when the initial autopilot met all of the margin requirements. Example #6 was run to see how the load relief phase performs when the initial autopilot does not meet the margin requirements.

The initial autopilot for this COEBRA run was obtained as follows. In several booster autopilots, there is a feedback loop that is used solely for high frequency stabilization. This loop is "washed out" at frequencies below the rigid-body phase margin, and serves to compensate for the load relief loop gain at high frequencies so that the load relief loop gain can be increased. So for Example #6, a previously designed autopilot case was chosen, and this "high frequency" loop was zeroed out. This resulted in unacceptable stability margins. The design objective for the COEBRA run was to not only return to the condition where all margins are met, but also to achieve at least the same amount of load relief that was achieved with the engineer's original autopilot that used this high frequency feedback loop.

For Example #6, three flight conditions were designed together:

(1) load relief loop switch-in; (2) max-$\bar{q}$; and (3) load relief loop switch-out. The airframe included four bending modes at two of the time points, and three modes at the third.

Figure 4.15 is a block diagram of the airframe/autopilot system. This is an analog autopilot design problem, and therefore the design is performed in the S-plane. There are 14 autopilot variables, but since this is an analog autopilot, each of these variables must have the same value at all three time points.

Figure 4.16 is the open-loop frequency response plot that resulted from COEBRA's initial autopilot at the max-$\bar{q}$ flight condition. It shows that not all margin requirements are met. This same situation exists at the other two flight conditions (not shown).

The load relief indicator for the engineer's original autopilot that used the so-called high frequency feedback loop was 4490 pounds per square foot. This result was obtained from a 6 DOF trajectory simulation. COEBRA's first step was to "get feasible", but in so doing, it had to give up load relief capability. COEBRA met all margins after three iterations, but $q\alpha$ (from a 6 DOF simulation) increased to 4580 pounds per square foot. However, from the 4th to the 8th iteration, all margin requirements remained satisfied, and $q\alpha$ began decreasing, until on the 8th and final iteration, it had decreased to 3975 pounds per square foot. Again, this was obtained from a 6 DOF simulation, and this $q\alpha$ was 12% less than that of the original autopilot

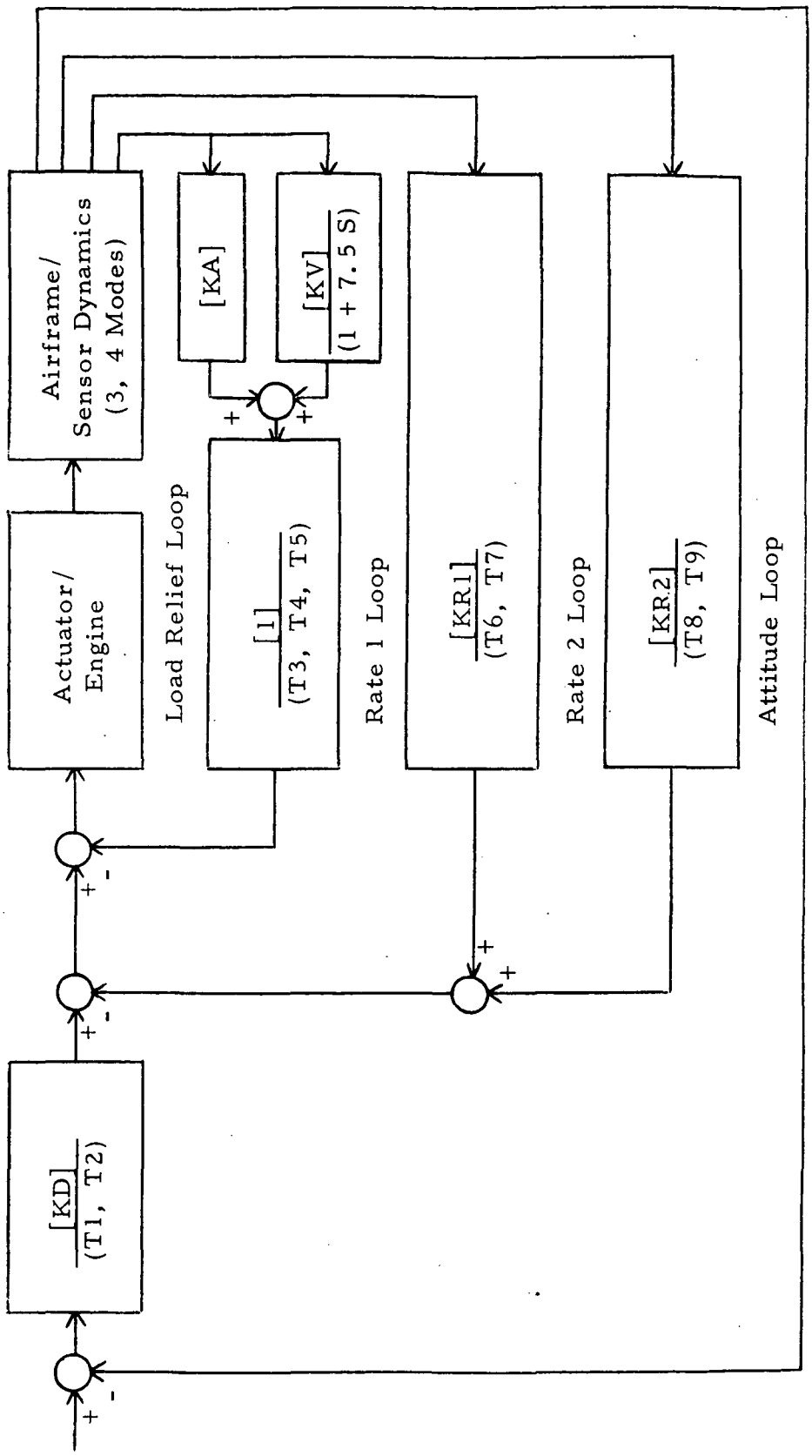


Figure 4.15. System Block Diagram for Example #6

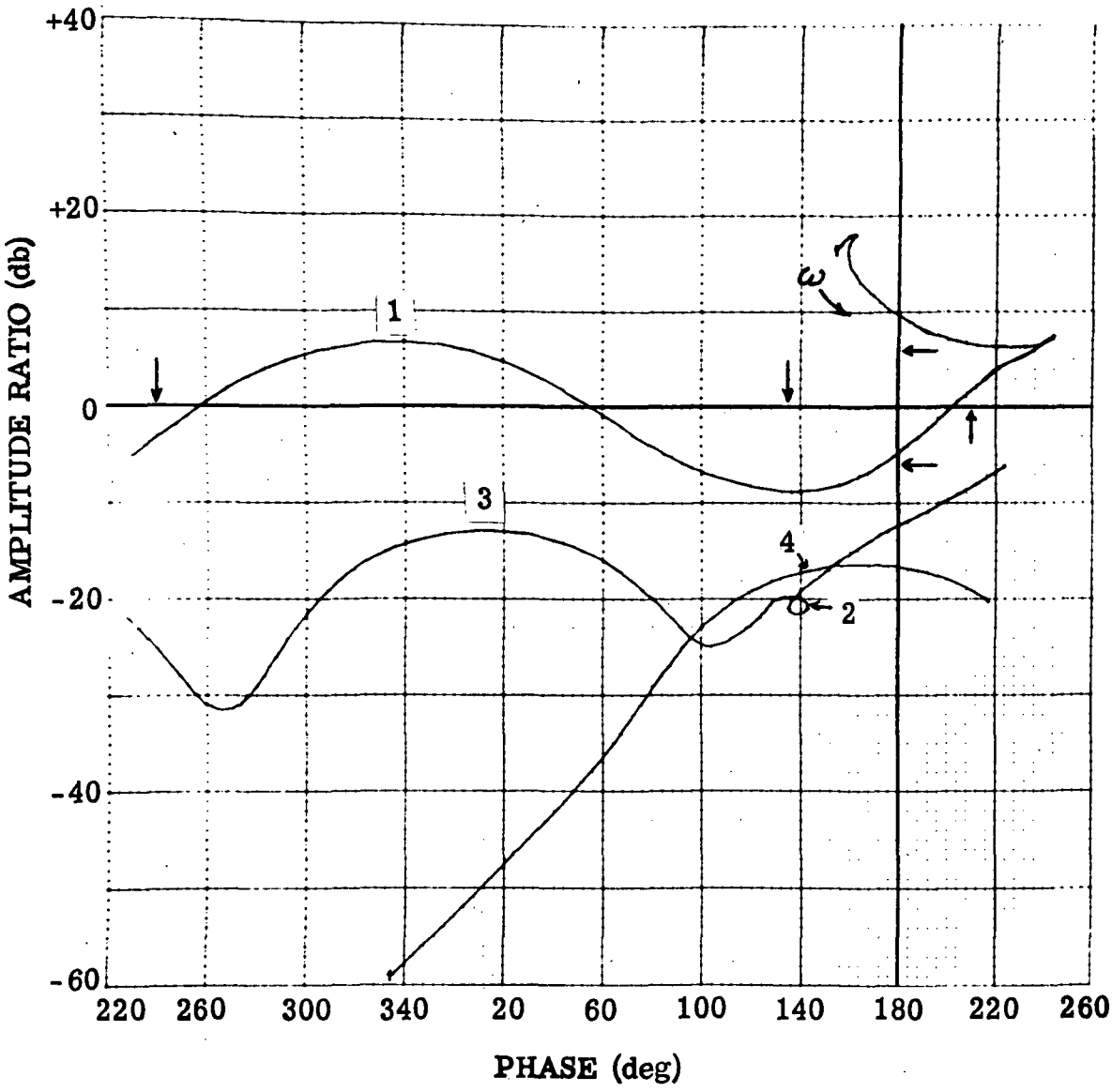


Figure 4.16. Example #6, Gain-Phase Frequency Response Plot Resulting From Initial Autopilot at Max-$\bar{q}$

with the high frequency feedback loop. Figure 4.17 does show that all margins are met at the max-$\bar{q}$ time point with the results of the 8th and final COEBRA iteration. The same situation existed at the other two flight times.

The following discussion refers to Section 3.5.4 of Chapter 3 on Convergence to an Exterior Optimum. This example has dramatically demonstrated how the COEBRA algorithm converges to a constrained optimum from an unfeasible initial point. The first three iterations were required in order to reach a feasible solution. In "getting feasible", load relief capability was reduced. This did not necessarily have to happen, since the algorithm does try to optimize while "getting feasible". Once the feasible region was reached, the algorithm moved along or parallel to the constraint boundaries until the constrained optimum was reached. The fact that this actually occurred is known because load relief capability steadily increased from the 4th to the 8th iteration, while several stability margins remained "tight against" their requirements. Two of these "tight margins" can be seen in Figure 4.17. These two margins are called the rigid-body phase margin, and the phase margin on the "backside" of the first structural bending mode. These margins are indicated by arrows in Figure 4.17. These margins were tight after three iterations, and remained tight from the 4th to the 8th and last iteration.

Table 4.8 summarizes the results of Example #6. This table

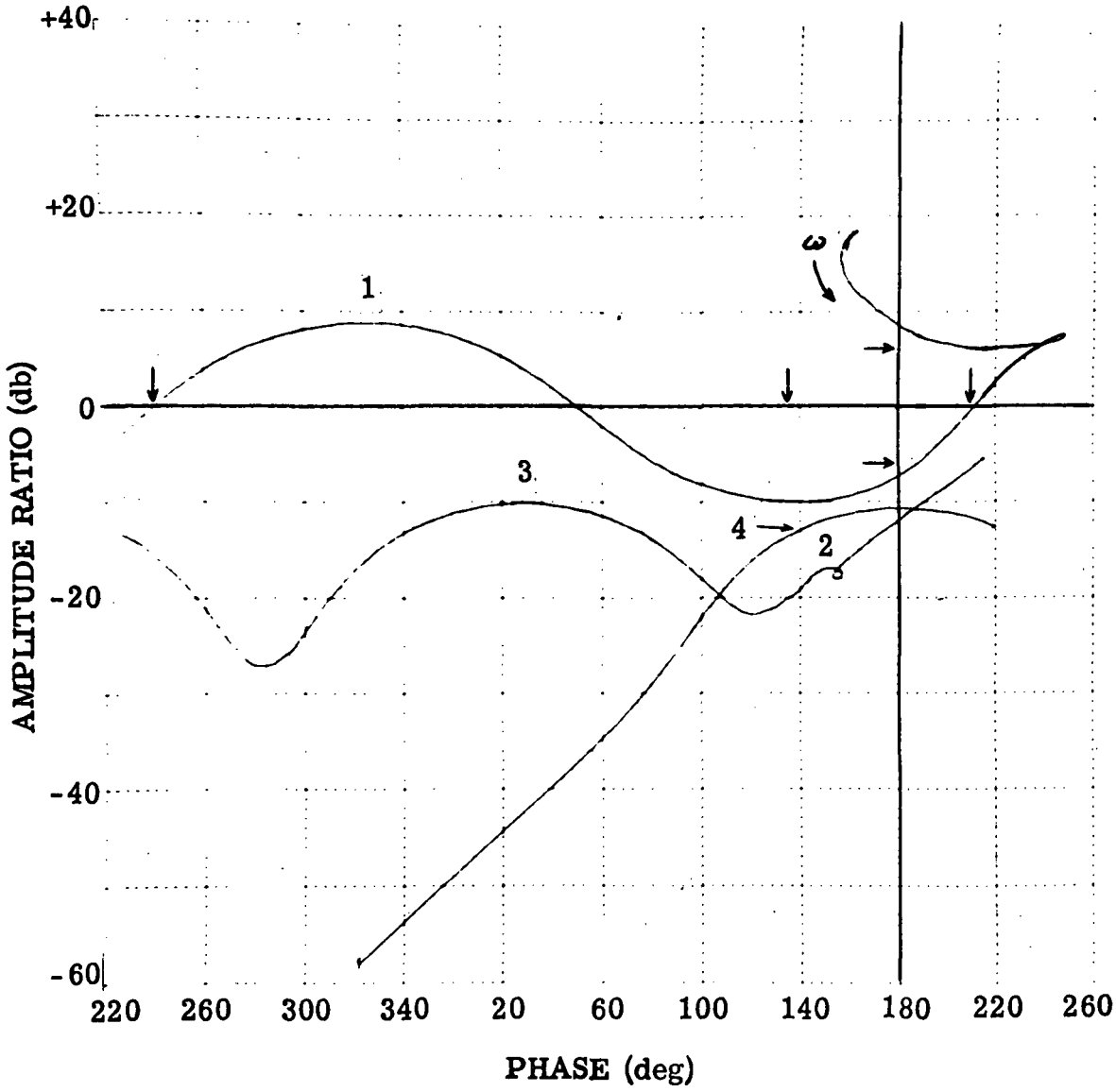


Figure 4.17. Example #6, Gain-Phase Frequency Response Plot Resulting From Eighth (Final) Iteration at Max-$\overline{q}$

Table 4.8. Example #6 Summary of Results

- Objective: Maximize Load Relief Capability

Autopilot	Stability Margins	$q\alpha$
Engineer's Final	Satisfactory	4490
COEBRA Initial	Unsatisfactory	4490
Third Iteration	Satisfactory	4580
Eighth (Final) Iteration	Satisfactory	3975

- System Order:
  - 25th Order at 2 Time Points
  - 23rd Order at 1 Time Point
  - 14 Autopilot Variables Which Must Have Same Value at Each Time Point
- 8 Major Iterations
- Computer Time: 59.2 Minutes or 7.4 Minutes per Iteration.

shows that the computer time for this example was 59.2 minutes, or 7.4 minutes per iteration.

4.7 Example #7

Example #7 illustrates using COEBRA to design a load relief autopilot in two phases: (1) the initial phase being to first meet all margin requirements; (2) the second phase being to optimize load relief capability. For this example, this approach was considered essential because the "first guess" or initial autopilot was very poor.

This example is taken from a recent effort to design an autopilot for a space shuttle booster configuration. COEBRA was used to design the autopilot for all three channels (pitch, yaw, and roll) at all the critical flight conditions during the first two minutes of ascent. At all the flight times, the airframe included from seven to eight structural bending modes.

The flight condition for this example is the yaw channel during the max-$\bar{q}$ portion of flight. Three time points were designed together: (1) load relief switch-in; (2) max-$\bar{q}$; and (3) load relief switch-out. The airframe included seven modes at each of the time points. While all the results obtained from this design effort are worth noting, this example was selected since it illustrates the two phased approach to load relief autopilot design.

Figure 4.18 is the airframe/autopilot block diagram for this example. It shows the attitude loop, a rate loop, and the load relief

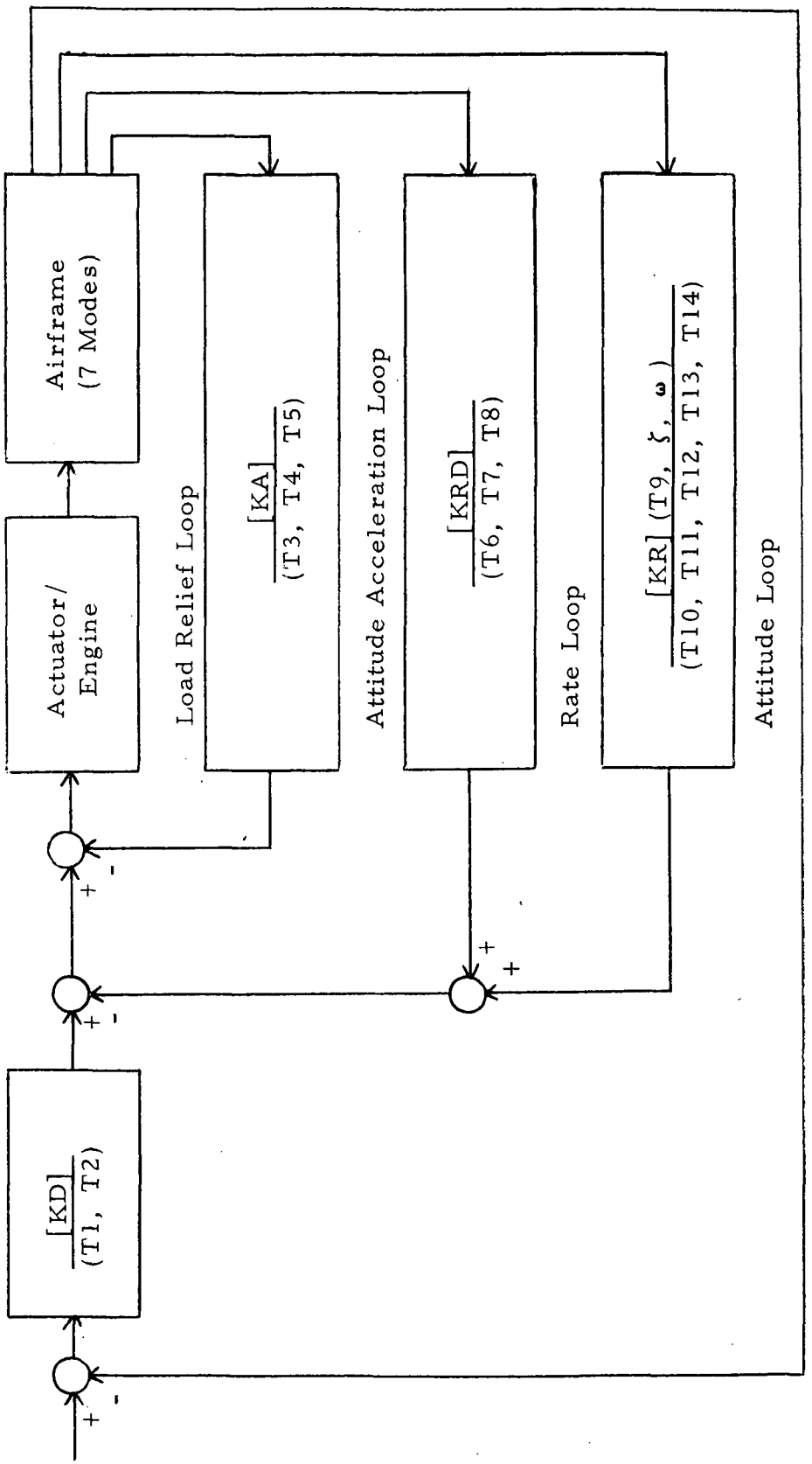


Figure 4.18 System Block Diagram for Example #7

loop. In addition, it shows an attitude acceleration loop. This is the so-called high frequency loop that was referred to in Example #6. Figure 4.18 shows that there are 20 autopilot variables at each time point. The four gains can have different values at each time point, but the 16 filter parameters must have the same value for all the time points.

Figures 4.19, 4.20 and 4.21 show the frequency response plots for the initial autopilot. The system is stable, but the initial autopilot is very poor. The basic margin requirement is that modes 3 through 7 be gain stabilized with a peak amplitude below "-10" decibels. Only the first and second modes can be phase stabilized, but if they are gain stabilized, their so-called "closest approach" distance to the "-1" point must be equivalent to 10 decibels.

The first COEBRA run was made to optimize stability margins. After one iteration, all margins were met. The next COEBRA runs were made to optimize load relief. After six more iterations, an autopilot resulted that yielded the plots of Figures 4.22, 4.23, and 4.24. All margin requirements are met. A 6 DOF trajectory simulation was not made, but estimates based on linear transient response results indicate that bending moment loads were reduced 25% from the initial to the final autopilot. Computer time required to do this job was 98 minutes, or 14 minutes per iteration.

This example points to another way in which the COEBRA

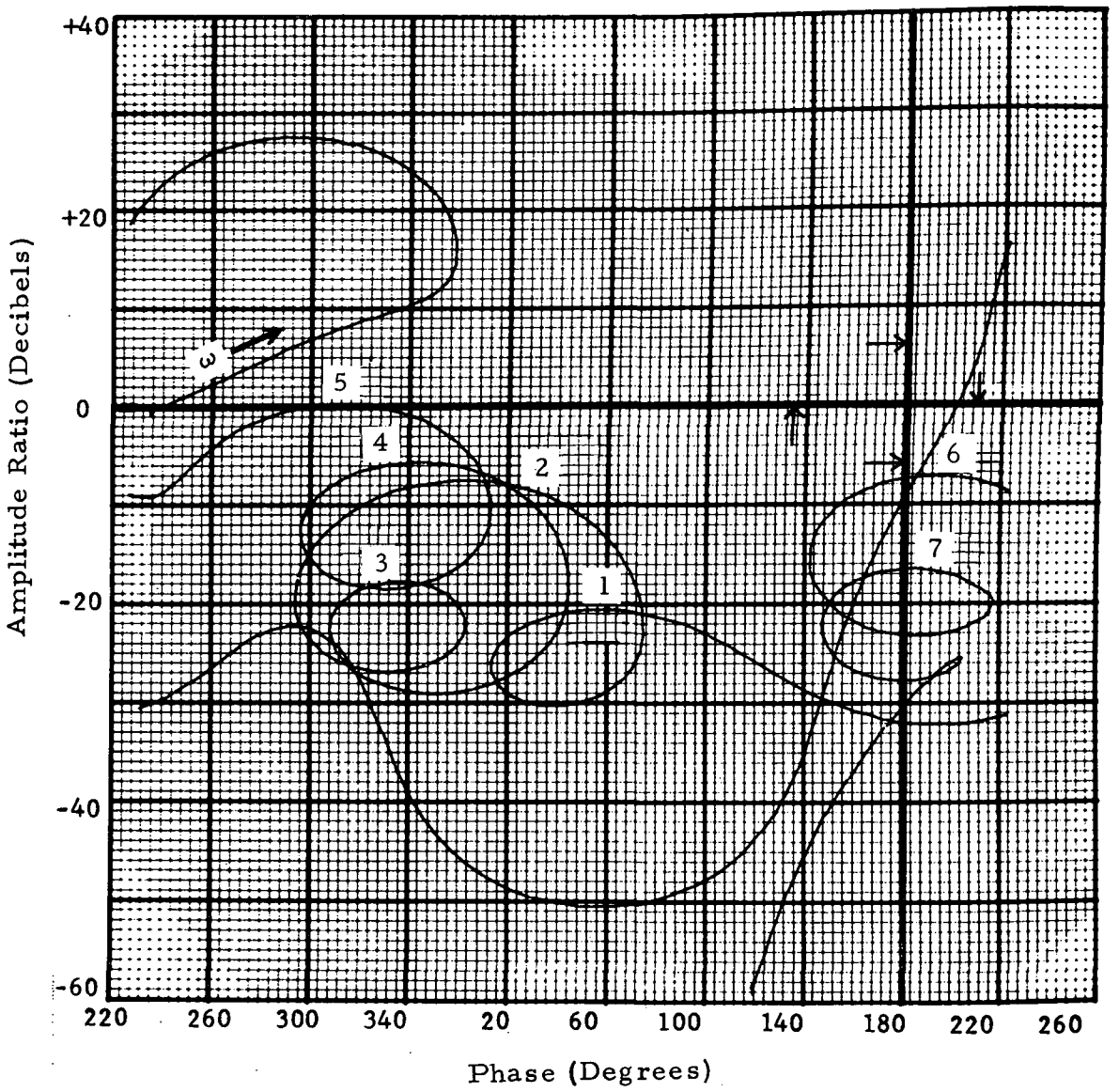


Figure 4.19. Example #7, Gain-Phase Frequency Response Plot Resulting From Initial Autopilot at Load Relief Switch-in

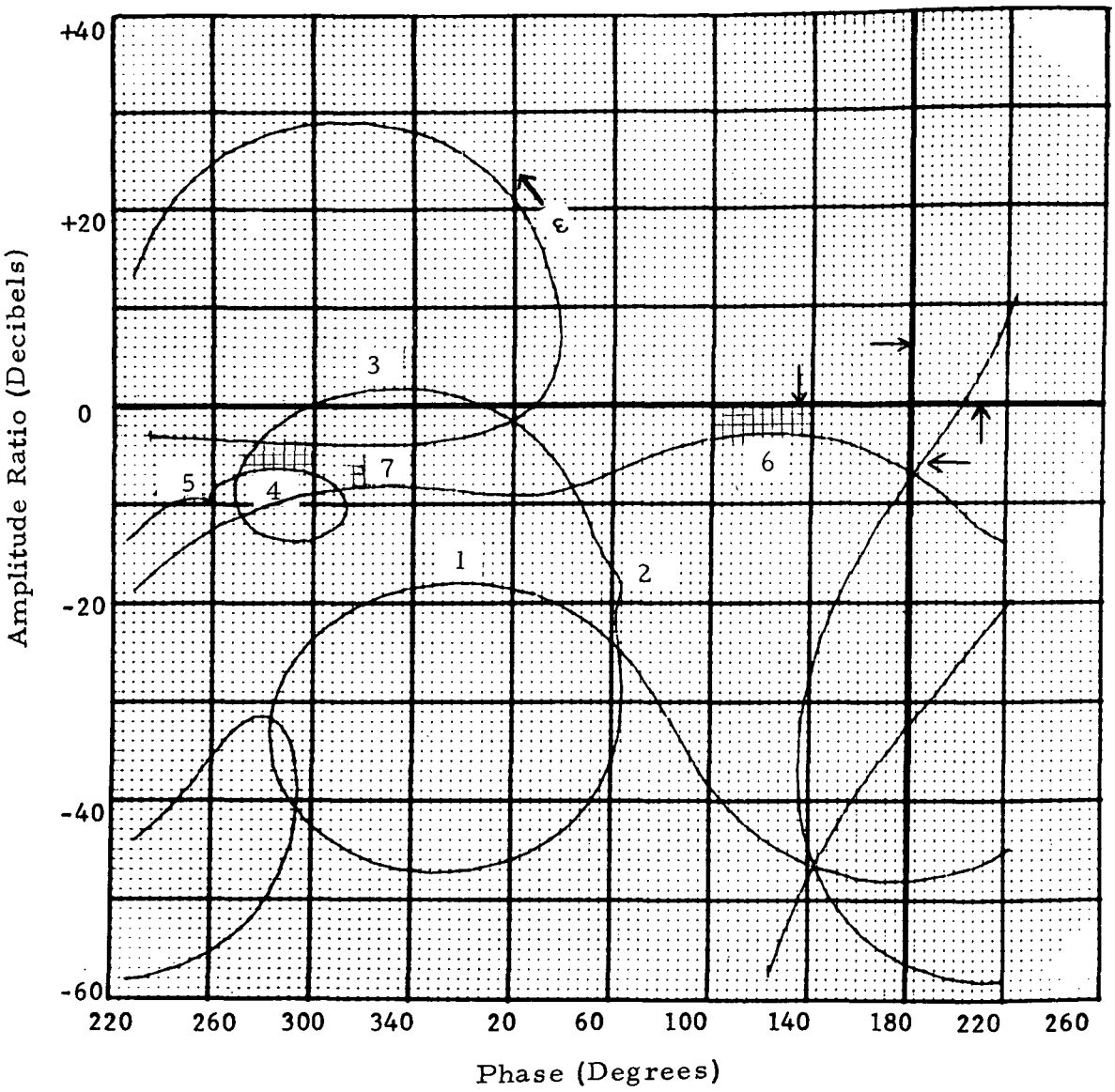


Figure 4.20. Example #7, Gain-Phase Frequency Response Plot Resulting From Initial Autopilot at Max-$\bar{q}$

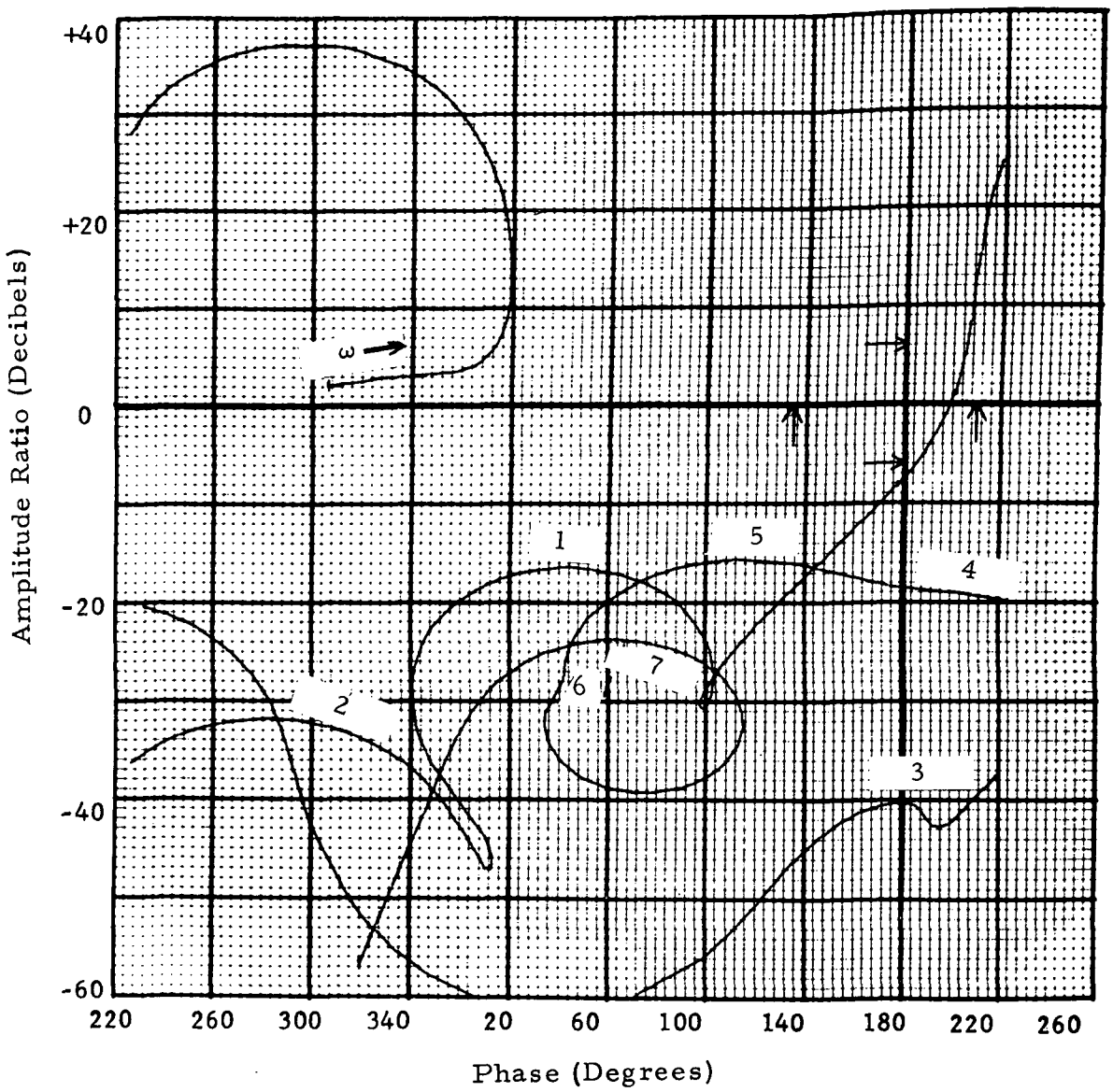


Figure 4.21. Example #7, Gain-Phase Frequency Response Plot Resulting From Initial Autopilot at Load Relief Switch-out

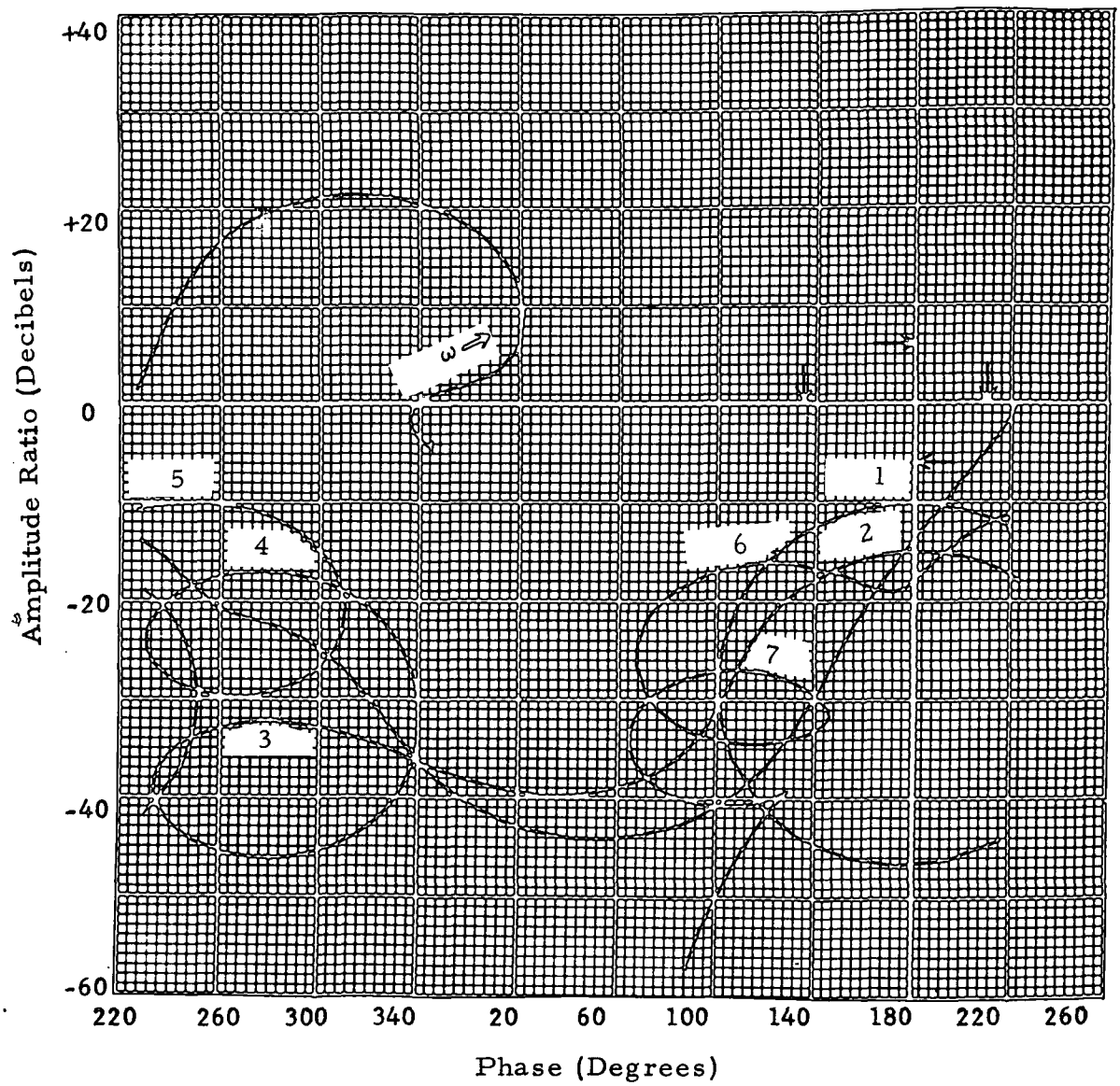


Figure 4.22. Example #7, Gain-Phase Frequency Response Plot Resulting From Seventh (Final) Iteration at Load Relief Switch-in

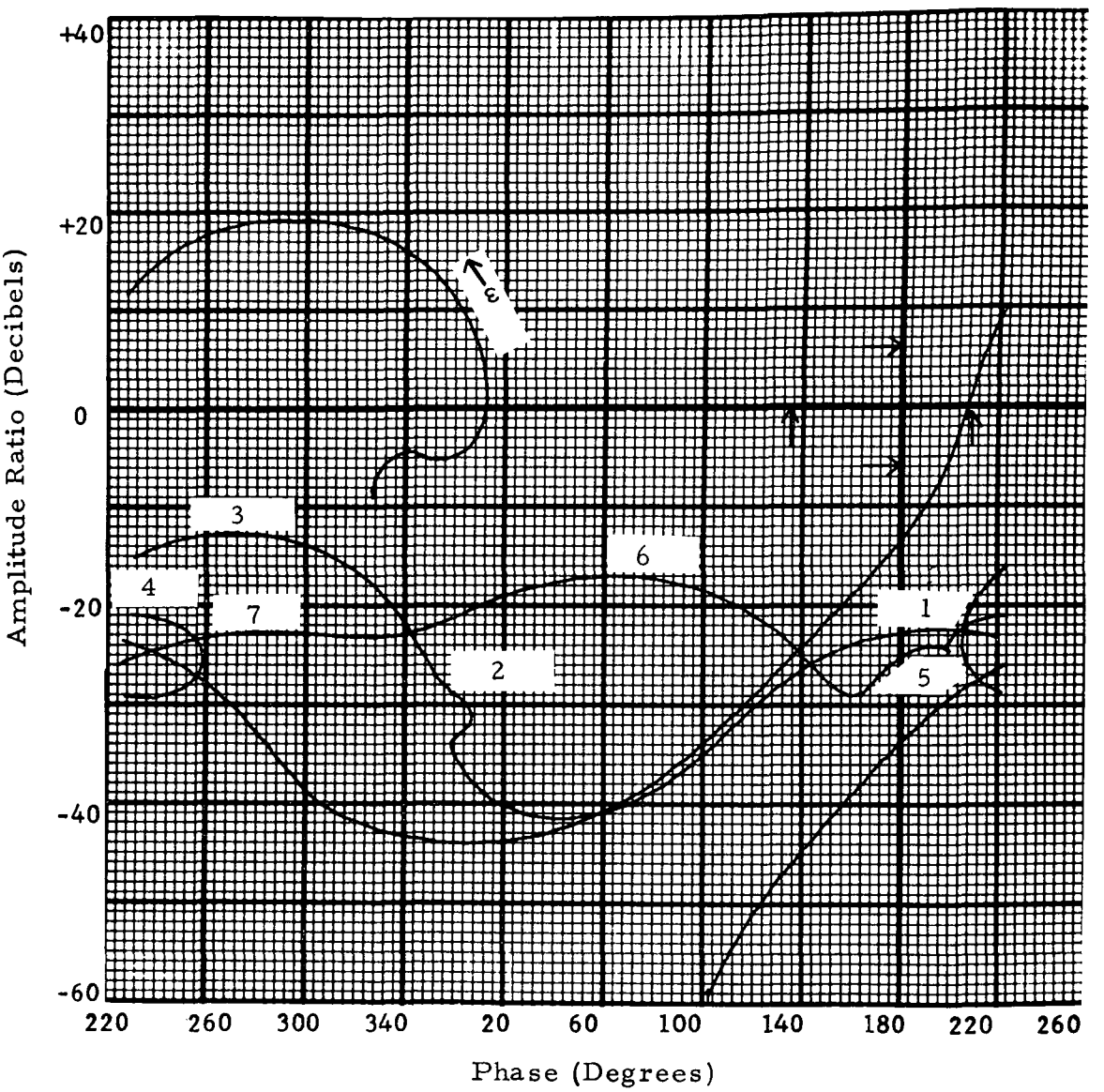


Figure 4.23. Example #7, Gain-Phase Frequency Response Plot Resulting From Seventh (Final) Iteration at Max-$\overline{q}$

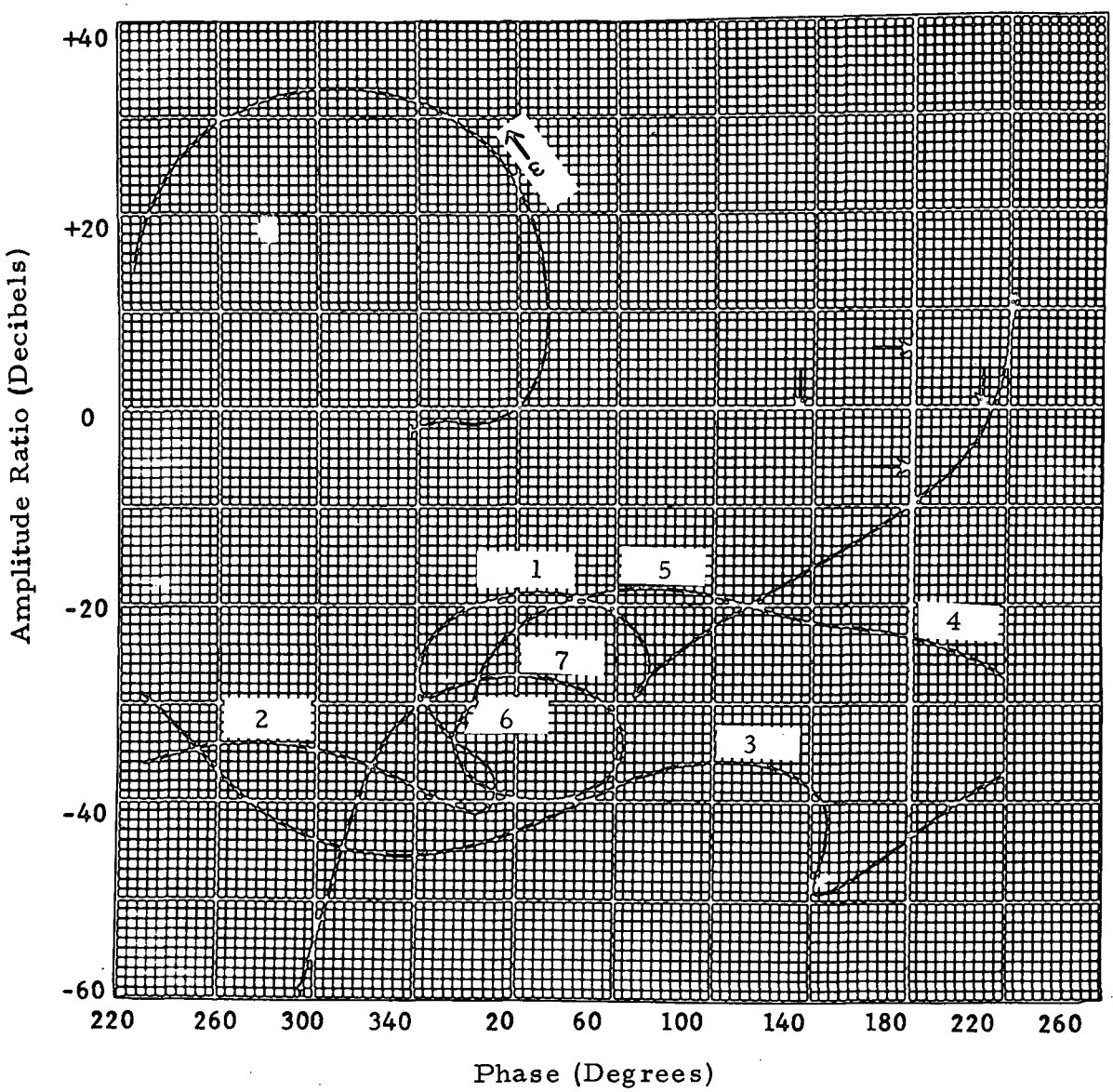


Figure 4.24. Example #7, Gain-Phase Frequency Response Plot Resulting From Seventh (Final) Iteration at Load Relief Switch-out

algorithm can be used. By observing the progress it is able to make from iteration to iteration, it can be used to design a minimum-complexity autopilot. For this example, the fact that COEBRA was able to satisfy all margin requirements in only one iteration, tends to indicate that some of the degrees of freedom in the autopilot could probably be eliminated.

4.8 Conclusions

The results presented in this chapter clearly demonstrate COEBRA's ability to successfully design autopilots for large flexible launch vehicles. Experience with the program shows that while it generally does not save computer time, it does save manpower and the time required to design an autopilot.

CHAPTER 5

SUMMARY, CONCLUSIONS AND SUGGESTIONS FOR FURTHER STUDY

5.1 Summary and Conclusions

As shown in Chapter 4, results from the COEBRA program clearly demonstrate that this algorithm successfully solves the problem of automating practical launch vehicle autopilot design and optimization. Perhaps the primary reason for the success of this algorithm is that its approach to design is much the same as the engineer's approach.

Via this algorithm, the COEBRA program satisfies the five basic design requirements that were given in Chapter 1. Referring to these requirements as they were listed in Chapter 1:

(1-a) The COEBRA program deals directly with stability margin requirements in the frequency domain. Referring to the discussion in Section 4.3 of Chapter 4, COEBRA is able to constrain the location of the so-called dominant rotational rigid-body roots. COEBRA puts an inequality constraint equation on each individual stability margin and each pair of dominant closed loop roots at each of the time points being designed together. With the "optimize margins" cost function, COEBRA not only meets the minimum margin requirements, but also seeks to optimize all stability margins. With a cost function separate from the constraint matrix, the cost function

in COEBRA can be formed from the margin objectives, while the constraint matrix can be formed from the margin requirements;

(1-b) With the "optimize load reduction" cost function (formed via a time domain transient response routine), COEBRA seeks to minimize structural bending moment loads (β, δ_ψ, and δ_ϕ due to winds) while meeting the minimum margin requirements;

(1-c) COEBRA can constrain the autopilot parameters to the so-called Drift Minimum condition [14, 17], thereby minimizing trajectory dispersions. In fact, COEBRA can design a Drift Minimum autopilot that has the maximum amount of load relief capability and that meets the minimum stability margin requirements;

(2) COEBRA designs with a user-selected autopilot configuration. From the outset, only practical controllers are considered since the user selects the number and types of feedback loops and the number of gains and filters. COEBRA optimizes the values of the parameters within this feedback structure and constrains the minimum and maximum allowed values on each parameter;

(3) COEBRA handles the problem of multiple time point design by forming the cost function and matrix of constraint equations from margins and wind responses at several time points or vehicle states. In this manner, all vehicle states are optimized simultaneously. Autopilot parameters can be shared between the vehicle states.

A novel feature of the COEBRA program and this design algo-

rithm is that multiple time points are handled by considering a separate airframe for each time point. It is obvious that these "separate airframes" can come from the same flight time. For example, the "first airframe" can be the nominal airframe at time t_1 while the "second airframe" can be the airframe at time t_1 with a tolerance on one or more of the vehicle parameters. In this way, COEBRA can treat both the nominal and the toleranced airframe together to yield a single autopilot that will handle both conditions;

(4) COEBRA can handle a very high order system (30th and greater with up to eight bending and slosh modes per time point). With a user selected feedback configuration, the complexity of the autopilot does not necessarily increase with an increase in the order of the fixed parts of the system. Things like sensor and actuator dynamics are included in a very straightforward manner and their inclusion only increases the required computations;

(5) COEBRA designs analog autopilots via the S-plane frequency response, and digital autopilots via the W-plane frequency response.

Examples in Chapter 4 show that this algorithm can handle both interior and exterior optima. The examples also show that the initial conditions on the controller parameters need not yield feasible solutions, i.e., solutions that meet the constraint requirements. In fact, the examples in Chapter 4 demonstrate that the initial condition

on the autopilot parameters need not even yield a stable system.

5.2 Projected Applications

Even though the class of problems this algorithm can handle has not been established, it would appear that it can handle a large variety of engineering-type problems.

For example, it would appear it can handle the problem of designing an airplane flight control system with the so-called flying qualities design criteria [2]. These criteria include: (1) the longitudinal plane requirements on phugoid stability, flight path stability, and short period response; (2) the lateral-directional flying qualities criteria on the responses of the dutch-roll mode, the spiral mode and the roll mode; and (3) miscellaneous requirements on capability to perform crosswind landings, coordinated turns, etc. These criteria could simply be added to the flexible-body stability margin design requirements that are already included in COEBRA.

Another problem that this algorithm could probably handle is the design of a reaction control system. This type of control system uses discrete control. This algorithm could be used to optimize phase plane switching logic like the so-called "near-minimum-fuel" switching logic developed by Carney and Conover [5]. Their phase plane logic was developed for a digital attitude control system that requires no rate gyros.

Another problem that this algorithm can surely handle would

be designing the autopilot for an interplanetary spacecraft like the Mariner [Kopf, 22]. In order to handle the Mariner autopilot design problem, COEBRA needs a transient response routine that could put requirements like rise time, overshoot, settling time and steady state error on the vehicle's attitude response due to guidance commands. In fact, without this transient response routine, the COEBRA program has difficulty in designing an autopilot for an aerodynamically stable launch vehicle like the present proposed Space Shuttle configuration. The reason for this is that since the vehicle is stable, COEBRA can reduce the attitude gain and as it does so, even though the closed loop rigid-body rotational roots meet a certain requirement, these rotational roots no longer dominate the drift root.

To conclude this discussion, a paper written by Robinson [28] is noted. In this paper, Robinson states that the COEBRA algorithm should prove fruitful in the optimal control of distributed parameter systems.

5.3 Suggestions for Further Study

An advantage of this algorithm is that additional design criteria can easily be added. For example, it is planned to include in the present COEBRA program, a routine for load relief optimization in the presence of stochastic winds. It is planned to augment the present approach that uses a deterministic load relief cost function, with an approach using a cost function that is formed via Wiener's Theorem

and the filtering property of power spectral density functions [Chang, 6]. Two other types of routines that could easily be added have already been discussed, namely, a transient response routine specifying the vehicle's attitude response, and a routine that would incorporate design criteria on airplane flying qualities.

This chapter concludes with a discussion of two additional extensions of this design algorithm (and the COEBRA program) that warrant further work.

The first is the extension of this algorithm to handle multiple input systems. An example would be the lateral-directional control system of an airplane where the two inputs are the rudder and the aileron. The present algorithm must design this type system in a series manner, by first optimizing one channel (e.g., yaw) with the other closed (e.g., roll), and then closing yaw and optimizing roll, etc., until the optimum "mixed" system is achieved. Extension to multiple input capability could be achieved using the concept of the Transfer Matrix [Ogata, 27] where all channels could be designed at the same time.

The second extension involves the blended use of this algorithm with linear quadratic-cost optimal control theory (i.e., the Regulator Problem). In other words, this algorithm and linear optimal control theory could be used together to design launch vehicle autopilots. The time invariant Matrix Ricatti equation could be solved to yield optimum

rigid-body or low frequency performance. The algorithm presented in this report (via the COEBRA program) could then be used to solve the flexible-body problem via filters and sensor locations. This would require judicious modelling of the rigid-body problem so that the system states would be viable feedback states (e.g., accelerometer feedback rather than angle of attack feedback, etc.). This blended approach to autopilot design could be used to take advantage of the desirable features of each design method while circumventing the disadvantages involved when using each design method separately. For quadatic-cost optimal control theory, the advantages referred to include its ability to directly treat (1) the multiple-input problem, (2) the time domain tradeoff between bending moment loads, trajectory dispersions, control deflections, etc., and (3) stochastic as well as deterministic forcing functions. The advantages to the COEBRA algorithm of course, include its ability to directly treat (1) frequency domain design criteria which is essential for the flexible-body problem, (2) a user-selected autopilot configuration, and (3) the multiple vehicle state problem.

BIBLIOGRAPHY

[1] Anderson, B. and J. Moore, Linear Optimal Control, Englewood Cliffs, New Jersey, Prentice-Hall, 1971.

[2] Anonymous, Military Specification - Flying Qualities of Piloted Airplanes, MIL-F-8785B(ASG), August 1969.

[3] Blackburn, T. R. and D. R. Vaughan, "Application of Linear Optimal Control and Filtering Theory to the Saturn V Launch Vehicle," Proceedings of the 19th Congress of the International Astronautical Federation, October 1968.

[4] Burris, P. M. and M. A. Bender, "Aircraft Load Alleviation and Mode Stabilization (LAMS)," AFFDL-TR-68-161, November 1969.

[5] Carney, R. and T. Conover, "Digital Attitude Control System," Trans. IEEE on Aerospace and Navigational Electronics, March 1963.

[6] Chang, S. S. L., Synthesis of Optimum Control Systems, New York, N. Y., McGraw-Hill, 1961.

[7] Coffee, T. C., "The Automatic Frequency-Domain Synthesis of Multiloop Control Systems," Proceedings AIAA Aerospace Computer System Conference, September 1969.

[8] Currie, M. G., "Experience With Modern Control Theory on Flight Control Design," McDonnell Douglas Astronautics Company, Western Division, WD1315, February 1970.

[9] Dantzig, G. B., "Maximization of a Linear Function of Variables Subject to Linear Inequalities," Activity Analysis of Production and Allocation, New York, N. Y., Wiley, 1951.

[10] Davidon, W. C., "Variable Metric Method for Minimization," A.E.C. Research and Development, ANL-5990, December 1959.

[11] DiStefano, J. J., et al., "Schaum's Outline of Theory and Problems of Feedback and Control Systems," Schaum's Outline Series, New York, N. Y., McGraw-Hill, 1967.

[12] Edinger, L. D., et al., "Design of a Load Relief Control System," NASA CR-61169, April 1967.

[13] Fletcher, R. and M. Powell, "A Rapidly Convergent Descent Method for Minimization," Computer Journal, 6, 2 (1963).

[14] Greensite, A. L., "Analysis and Design of Space Vehicle Flight Control Systems, Vol. VII, Attitude Control During Launch," NASA CR-826, July 1967.

[15] Harris, R. D., "Analysis and Design of Space Vehicle Flight Control Systems, Vol. XIV, Load Relief," NASA CR-833, August 1967.

[16] Hendricks, T. and H. D'Angelo, "An Optimal Fixed Control Structure Design with Minimal Sensitivity for a Large Elastic Booster," Proceedings 5th Annual Allerton Conference on Circuit and Systems Theory, Urbana, Illinois, October 1967.

[17] Hoelker, R. F., "Theory of Artificial Stabilization of Missiles and Space Vehicles with Exposition of Four Control Principles," George C. Marshall Space Flight Center, NASA TN D-555, June 1961.

[18] Hofmann, L. G., "Topics on Practical Application of Optimal Control to Single and Multiple Control-Point Flight Control Problems," AFFDL-TR-70-52, February 1971.

[19] Hooke, R. and T. Jeeves, "Direct Search Solution of Numerical and Statistical Problems," JACM, Vol. 8, No. 2, April 1961.

[20] Kalman, R. E., "When is a Linear Control System Optimal?," Journal of Basic Engineering, March 1964.

[21] Klingman, W. R. and D. M. Himmelblau, "Nonlinear Programming with the Aid of a Multiple-Gradient Summation Technique," JACM, Vol. 11, No. 4 (October 1964).

[22] Kopf, E. H., "A Mariner Orbiter Autopilot Design," Jet Propulsion Laboratory, Pasadena, California, NASA Report No. 32-1349, January 1969.

[23] Kuo, B. C., Analysis and Synthesis of Sampled-Data Control Systems, Englewood Cliffs, New Jersey, Prentice-Hall, 1963.

[24] Lorenzetti, R. C. and G. L. Nelson, "Direct Lift Control For the LAMS B-52," AFFDL-TR-68-134, October 1968.

[25] _____, "Computerized Design of Optimal Direct Lift Controller," J. Aircraft, Vol. 6, No. 2, March-April 1969.

[26] _____, "Direct Lift Control for Approach and Landing," J. Aircraft, Vol. 6, No. 3, May-June 1969.

[27] Ogata, K., State Space Analysis of Control Systems, Englewood Cliffs, New Jersey, Prentice-Hall, 1967.

[28] Robinson, A. C., "A Survey of Optimal Control of Distributed-Parameter Systems," Wright-Patterson Air Force Base, ARL 69-0177, November 1969.

[29] Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints," J. Soc. Indust. Appl. Math, Vol. 8, No. 1, March 1960.

[30] _____, "The Gradient Projection Method for Nonlinear Programming. Part II. Nonlinear Constraints," J. Soc. Indust. Appl. Math, Vol. 9, No. 4, December 1961.

[31] Rynaski, E. G., et al., "Optimal Control of a Flexible Launch Vehicle," Cornell Aeronautical Laboratory, NASA CR-80772, July 1966.

[32] _____, "Sensitivity Considerations in the Optimal Control of a Flexible Launch Vehicle," Cornell Aeronautical Laboratory, NASA CR-89568, June 1967.

[33] Shah, B. V., et al., "The Method of Parallel Tangents (PARTAN) For Finding an Optimum," Office of Naval Research, NR-042-207 (No. 2), 1961.

[34] Stapleford, R. L., et al., "A Practical Optimization Design Procedure for Stability Augmentation Systems," AFFDL-TR-70-11, October 1970.

[35] Stear, E. B. and C. P. Lefkowitz, "Automated Design of Space Booster Control Systems," Proceedings Joint Conference on Mathematical and Computer Aids to Design, Anaheim, California, 1969.

[36] Stein, G. and A. H. Henke, "A Design Procedure and Handling-Quality Criteria for Lateral-Directional Flight Control Systems," AFFDL-TR-70-152, May 1971.

[37] Vandierendonck, A. J., "Design Method for Fully Augmented Systems for Variable Flight Conditions," AFFDL-TR-71-152, January 1972.

[38] Whitbeck, R. F., "A Frequency Domain Approach to Linear Optimal Control," J. Aircraft, Vol. 5, No. 4, July-August 1968.

[39] Wilde, D. J. and C. S. Beightler, Foundations of Optimization, Englewood Cliffs, New Jersey, Prentice-Hall, 1967.