THE USE OF EXPERT-SYSTEMS PROGRAMMING TECHNIQUES FOR THE DESIGN OF LEAD-LAG COMPENSATORS

John R. James, Electrical Engineering
United States Military Academy, West Point, NY 10996

Dean K. Frederick, Electrical, Computer, & Systems Engineering
Rensselaer Polytechnic Institute, Troy, NY 12181

James H. Taylor, Corporate Research and Development
General Electric Company, Schenectady, NY 12345

CONTROL '85
Cambridge University
July 1985

ABSTRACT. We discuss the application of expert systems programming techniques to the design of lead-lag compensators for a single-input, single-output, continuous-time plant. A design method based on first adjusting the high-frequency response with lead and constant-gain compensators followed by adjusting the low-frequency response with lag compensators has been developed. This design heuristic is presented and its ability to achieve specifications for a range of different plants is discussed. Inference engine capabilities and extensions to a conventional analysis and design package required to implement this automatic control system design method are discussed.

1. INTRODUCTION

While the analytical results of classical and modern control theory are stated with mathematical precision, the design expertise required to use a particular technique is normally in the form of heuristics (rules-of-thumb) and informal trial-and-error procedures. Also, although synthesis methods may appear to be straightforward, there are often subtle points in applying a particular procedure that are only fully appreciated after the user has attained a moderate to high level of experience. Thus, although a number of powerful software packages have been developed for computer-aided control engineering, their capabilities may be difficult for an inexperienced or part-time user to exploit due to the complexity of the programs and lack of guidance in their effective use. The result is that there is a pressing need to aid the less-than-expert applications engineer in ways that are difficult to accomplish through traditional programming methods.

We believe that expert-systems programming techniques offer a solution to such problems. We will support this opinion by describing an expert system that has been built to apply the powerful numerical routines available in the Cambridge Linear Analysis and Design Program (CLADP, Edmunds [1]) to complete an iterative design to meet specifications on closed-loop bandwidth, gain margin, and low-frequency gain (position- or velocity-error constant). This capability is a part of the overall process handled by our expert system, which we have designed to encompass a much larger part of the Computer-Aided Control Engineering (CACE) domain. Because we see this system as the beginning of a third-generation of CACE systems (Taylor and Frederick [3]), we call it CACE-III.

The use of expert systems to represent and apply expert knowledge in a particular domain is becoming increasingly widespread (cf. Hayes-Roth [2]). Our recent effort ([3], Taylor et al. [4,5]), and James et al. [6]) has been directed at constructing an expert system to aid a control engineer in exploiting available software to achieve an acceptable design in a broad sense that includes support for model development, specification development, control system design, and design validation. In this paper we discuss one element of this work, namely using the knowledge representation and inference mechanisms of expert systems to create a set of rules which will aid a control engineer in the design of lead-lag compensation for a single-input, single-output linear plant.

The knowledge representation scheme which we use (see [3]) is the production rule or "recognize-act" formalism. In this method the expertise is represented by a set of rules in the form of IF <premise> THEN <conclusion>. The premise is a conjunction of clauses which describe a possible situation, and the conclusion is a disjunction of actions which are to be performed if the situation occurs. Additional details on knowledge representation and inference mechanisms are contained in [2, 3, 5, and 6]. In this paper, we will not detail the rules themselves; rather, we will define the procedure in algorithmic terms so that the interested reader can apply it manually or embody the algorithm in an expert system.

In Section 2 we present the heuristic used for the design of lead-lag compensation, and Section 3 outlines the requirements for an inference engine to implement this heuristic. Section 4 deals with modifications and extensions to CLADP required to integrate it with the inference engine. We conclude with comments regarding the status and future of this effort.

2. A LEAD-LAG DESIGN HEURISTIC

We chose classical single-input/single-output frequency-domain design techniques for our initial study, with the expectation that well-established rules existed for its application. The goal was to have the expert system utilize CLADP to achieve specifications of gain margin, bandwidth, and low frequency gain by the addition of lead-lag compensation to a single-input, single-output linear plant. (In the CLADP nomenclature, this is called a precompensator, i.e., it is placed before the plant in the forward path, rather than after the plant or in the feedback path.) We discovered that numerous textbooks present in detail the results of adding lead or lag compensation (cf. D'azzo and Houpis [7]), and indicate that iteration is needed to achieve an acceptable design; however, a systematic approach for iterating the lead-lag compensator parameters to meet specifications is not given. The closest presentation meeting the requirement of providing a design *algorithm* is Thaler [8]; even this lacked the completeness required for rule-base development. In retrospect, we believe that the classical approach to control system design may be one of the more difficult to implement in an expert system, due to the visual aspects of the process (e.g., shaping Bode plots) and the amount of *ad hoc* adjustment required for success.

The rule base we developed automatically designs a compensator that achieves specifications for a variety of systems. The basic idea is to first adjust the high-frequency portion of the frequency response using either a constant gain or a combination of gain and lead compensators to meet the specifications for gain margin and closed-loop bandwidth. The low-frequency portion of the frequency response is then adjusted by adding lag compensators to meet the specification for position-error constant (type-0 systems) or velocity-error constant (type-1 systems).

A flow chart representation of the synthesis method is provided in Figure 2. Specific details of the algorithm are as follows:

A. Adjust the high-frequency portion of the frequency response. This is done in one of two ways, depending on the open-loop phase characteristic:

   1. If the minimum open-loop phase angle is greater than -180°, the gain margin is infinite so phase margin should be used to guide the design process. The secant algorithm is used to determine the gain that must be added to the forward path to meet the bandwidth specification, and a lead compensator may be added to provide the appropriate phase margin. The gain and lead are adjusted until the bandwidth specification is met.

   2. If the minimum open-loop phase angle is less than -180°, both the gain-margin specification and the closed-loop bandwidth specification must be met by adjusting the high-frequency portion of the frequency response. This is achieved as follows:

a. Add lead compensators to adjust the open-loop phase angle at the specified closed-loop bandwidth, ω_{sbw}. The transfer function used for each stage of lead is

$$G_{lead}(s) = K_{lead}\frac{\alpha_{lead}s + \omega_{ctr}\sqrt{\alpha_{lead}}}{s + \omega_{ctr}\sqrt{\alpha_{lead}}} \quad (1)$$

The center frequency of the lead, ω_{ctr}, is placed at ω_{sbw}. The amount of phase lead initially added is calculated to place the forward-path phase at ω_{sbw} in the range of -200° to -170° for a type-0 system and in the range of -170° to -140° for a type-1 system; the different ranges depending on system type were selected to account for the fact that type-1 systems tend to approach the origin closer to the negative real axis than do type-0 systems, so additional lead is required to maintain an appropriate distance from the critical point. The pole-zero ratio of the lead network (α_{lead}) that is needed to achieve the required amount of phase lead is calculated using the piecewise-linear approximation to the exact curve portrayed in Figure 3. Note that the product of the gain K_{lead} and this pole-zero ratio is the high-frequency gain of the lead compensator. Two or more stages of lead are used if necessary.

b. Adjust the compensator low-frequency gain(s) K_{lead} to meet the gain-margin specification.

c. Adjust the actual closed-loop bandwidth by making incremental changes in the values of α_{lead} to increase phase lead at ω_{sbw} by 10° when the bandwidth is too small and decrease phase lead by 10° when the bandwidth is too large, based on the approximate relation depicted in Figure 3.

d. Repeat steps (b) and (c) until the gain-margin and bandwidth specifications are within tolerance. When both conditions are satisfied, the high-frequency portion of the forward-path frequency response has been completed.

B. Add lag compensation to adjust the low-frequency gain. The lag transfer function is of the form

$$G_{lag}(s) = \frac{s + \omega_{zro}}{s + \omega_{zro}/\alpha_{lag}} \quad (2)$$

where ω_{zro} is the frequency associated with the zero of the transfer function and α_{lag} is the zero-pole ratio and is equal to the low-frequency gain. Again, more than one stage of lag may be used if necessary.

1. For a type-0 system, the zero of the lag is placed a decade below the magnitude crossover frequency ($\omega_{zro} = \omega_{mco}/10$) in order to make the high-frequency impact of the lag negligible. The pole of the lag is placed at $\omega_{zro}/\alpha_{lag}$, where α_{lag} is the amount of added low-frequency gain required to meet the low-frequency gain specification.
2. For type-1 systems, the zero of the lag compensator is placed an additional octave below the magnitude crossover frequency, making $\omega_{zro} = \omega_{mco}/20$. This modification corrects for the more pronounced effect that a lag compensator has upon magnitude-crossover conditions for type-1 systems compared with type-0 systems because of the increased low-frequency phase lag of the former.

The tolerances for declaring specifications to be met are: bandwidth: ±20%, gain margin: ±3dB, and low-frequency gain: ±3dB.

The rules for the initial adjustment of the open-loop phase angle are designed to add lead compensators to place the open-loop phase angle at the desired closed-loop bandwidth in the ranges given above. These ranges are based solely on experience. In most cases tried, especially where there are no lightly damped poles and/or zeroes, this algorithm has proven to be successful. For systems with very lightly-damped poles or zeroes, however, the algorithm does not initially place the leads at the appropriate frequencies, so the iteration scheme cannot meet the bandwidth and gain-margin specifications. The resonant case is known to be difficult, often requiring the design of compensators with complex zeroes [8].

The following example illustrates the ability of the algorithm to design compensation for a nontrivial plant. The expert system was given the transfer function

$$G(s) = \frac{500(s + 10)}{(s+2)\ (s+5)\ (s+3+j4)\ (s+3-j4)\ (s+20)} \quad (3)$$

with the design specifications of 9.0 rps bandwidth, 10.0 db gain margin, and 40.0 db low-frequency gain. Prior to beginning the design of compensation of the system, a Nyquist plot of the uncompensated system is displayed (Figure 4). As the design proceeds, the user is informed of the parameters of the lead, gain, and lag compensators being added. A Nyquist plot of the compensated system (Figure 5) is provided when the design is complete. The step response of the compensated system (Figure 6) is also provided if the user requests it. The complete closed-loop system is portrayed in Figure 7. The results achieved were 9.3 rps bandwidth, 8.5 db of gain margin, and 40.0 db of low-frequency gain. Each of these falls within the stated tolerance.

As indicated above, we use frequency-domain specifications to determine lead, lag and gain parameters in our control system design. However, if the system can be approximated by a second-order system with less-than-critical damping, an approximate relation exists between the bandwidth of the system and the rise time of the step response as well as between the maximum frequency-response amplitude and the percent overshoot to a step input [7]. There is also an approximate relation between the phase margin of the system and the percent overshoot of the step response (Dorf, [9]). An exact relation also exists between the low-frequency gain and the steady-state error to a step or ramp input. It is planned to provide the expert system with the ability to make use of these relations in order to support the user in entering specifications other than those used in the design rules.

3. INFERENCE-ENGINE CAPABILITIES

The choice of an inference engine with adequate means of knowledge representation and control of the inference process is crucial to the subsequent creation of the knowledge bases [2]. It has been our experience that the following capabilities define the minimum set needed to implement the logical flow of the control system design process:

a. arithmetic functions to support numerical operations within the inference engine,

b. use of logical variables which can be assigned either numerical or symbolic values, and

c. the ability to run and exchange information with external programs which execute the analysis and synthesis procedures via conventional means (e.g., CLADP).

Our implementation uses General Electric's DELPHI inference engine. DELPHI is written in Franz LISP and has the capabilities required. Specifically,

a. DELPHI has a PROG verb which provides access to the LISP prog feature and thus supports numerical calculations,

b. DELPHI also supports logical variables which can be assigned either numerical or symbolic values, and

c. DELPHI provides the capability to run external processes by creating a VAX/VMS subprocess which can be run through VMS mailboxes.

The LISP environment has also proven to be easily modified to create macro instructions for CLADP as well as to implement a variety of functions which support the exchange of numerical and symbolic data. This feature of the LISP environment has shortened the time which would otherwise have been required to build the expert system.

By applying the above capabilities, the expert system starts and runs CLADP (issuing the appropriate commands), assigns the values determined by CLADP routines to logical variables, and calculates new parameter values for the leads and lags. The results of these changes are then automatically analyzed using CLADP routines and displayed to the user. Thus, the expert system contains both the heuristic knowledge required to implement the lead-lag design algorithm and the capability to obtain the numerical results required to proceed. The protocol used to implement this two-way exchange of data is discussed in [6].

As a historical note, our initial implementation was made using an inference engine developed for General Electric's Diesel Electric Locomotive Troubleshooting Aid (DELTA, Johnson and Bonissone [10]). While DELTA supports both forward and backward chaining and has a variety of verbs and predicates, it does not provide for numerical computation, logical variables, or process running. This resulted in having to run CLADP from one computer terminal to determine the bandwidth, gain margin, low-frequency gain and other quantities required to implement the design heuristic, and using another terminal to run the expert system to enter these values as responses to queries. The user was thus in the position of acting as an interface between CLADP and the expert system. This limitation has been removed in the DELPHI implementation where the user interacts only with the expert system and all transactions with CLADP or other programs are handled automatically.

4. CLADP MODIFICATIONS AND EXTENSIONS

The implementation of the lead-lag design algorithm required that several changes be made to CLADP. These were of three types: new commands for obtaining necessary information, new commands for convenience and simplification of command sequences, and new input/output for communication with the expert system. The following essential new commands were added to CLADP to obtain data after a system frequency response has been calculated:

a. FREQ calculates the open- and closed-loop magnitudes and the open-loop phase angle at a specified frequency.

b. Given a specified open-loop magnitude, NMAG calculates the number of occurrences of the magnitude, the frequency of each occurrence, and the phase angle of each occurrence.

c. Given a specified closed-loop magnitude, CMAG calculates the number of occurrences of the magnitude, the frequency of each occurrence, and the phase angle of each occurrence.

d. Given a specified open-loop magnitude in decibels, DBMA calculates the number of occurrences of the magnitude, the frequency of each occurrence, and the phase angle of each occurrence.

e. Given a specified closed-loop magnitude in decibels, DBCM calculates the number of occurrences of the magnitude, the frequency of each occurrence, and the phase angle of each occurrence.

f. Given a specified phase, NPHA calculates the number of occurrences of the phase, the magnitude at each occurrence, in both decibels and absolute values, and the frequency of each occurrence.

g. GM calculates the -180° crossover frequency and the corresponding gain margin in both decibels and absolute value.

h. PM calculates the magnitude crossover frequency and the corresponding phase margin in degrees.

In addition, the following commands greatly facilitate adding compensator stages:

a. GAIN – creates a gain compensator having the gain specified either in decibels or as a real number,

b. LEAD – creates a lead compensator according to Equation (1), given the lead alpha (α_{lead}) and the lead center frequency (ω_{ctr}), and

c. LAG – creates a lag compensator according to Equation (2), given the lag alpha (α_{lag}) and the frequency of the zero (ω_{zro}).

Once it was determined that the above features were required to implement an automatic lead-lag compensator design in CACE-III, we realized that the commands listed above would be of considerable utility to the human user of CLADP as well. Their use can save the effort of generating plots and reading approximate data from the terminal display and defining compensator transfer functions in terms of the order and coefficients of the numerator and denominator polynomials.

In addition to the above extensions, we modified CLADP to read commands from a file and write the results in a form convenient for the inference engine to access and manipulate. The latter involved writing information to files in the form of three-tuples, which are the basic unit of information used by the expert system. For example, if CACE-III needs to know the closed-loop bandwidth with the present compensators, it issues the command DBCM -3.0. Then the results returned by CLADP are the three-tuples:

```
( 1' OCCURRENCE-OF-CL-MAG-DB . FOUND )
( SPECIFIED-MAGNITUDE OCCURS-AT-FREQ . 10.4 )
( AT-THIS-FREQ CL-MAG-DB-IS . -3.0 )
```

The same "signal-to-symbol" transformation could have been done by having DELPHI decipher the conventional CLADP output. However, it was easier and more effective to modify CLADP since we had access to the source code.

5. CONCLUSION

A number of control-design issues involved in developing an expert system to aid a control engineer in the design of lead-lag compensation for a single-input, single-output linear plant have been presented. Other issues such as the implemention of a partitioned rule-base architecture and the coordination of the symbolic manipulations of the inference engine with numeric calculations being performed by conventional software are detailed in [6]. The limitations of the design heuristic used to date have been discussed, and the results of applying the method to a type-0 plant having five poles and one zero have been given.

We are still seeking to improve the design heuristic. At present the two major deficiencies are a cycling of the design algorithm that occurs in some cases and the inability to handle resonant modes in a satisfactory manner. The cycling of the algorithm occurs while adjusting the high-frequency portion of the frequency response: Since the lead alphas are adjusted in increments of 10°, we occasionally find that an increase (decrease) of the lead alpha will cause a change in gain margin that requires a decrease (increase) in the lead alpha, etc. We expect that a reduction in the size of the increment made in the lead alpha, once cycling has been detected, will resolve the problem. The difficulty with the resonant modes is caused by the abrupt change in phase angle near the resonant frequency. We intend to expand the rule base so the expert system can add a compensator that has a pair of complex zeros to the left of the resonant poles and two real poles further to the left in the s-plane , i.e., a notch filter as is often done [8].

This presentation details an element of a substantially larger expert system, namely CACE-III [3-6]. The architecture for this expert system has been designed to provide support to the user in modeling, diagnosing, constraining, specifying, designing, and simulating a plant that can be either linear or nonlinear. The lead-lag design heuristic is currently the only working part of the design rule base. The larger expert system is based on using both CLADP and an extended version of SIMNON as the underlying conventional software (see Elmqvist [11] and Taylor [12] for the extensions). The latter serves to model and simulate the nonlinear plant, find equilibria, and determine linearized models to serve as the basis for controller design. Both of these packages are now integrated with the DELPHI inference engine, so we can expand the expertise of the expert system into other areas. These could include:

a. the design of 2-loop digital control systems with spectral separation (fast inner loop, slow outer loop),

b. the design of multiple-input, multiple-output control systems using the CLADP frequency-domain approaches [1],

c. the design of multiple-input, multiple-output control systems using the LQG/LTR method of Doyle and Stein [13], and

d. the analysis and design of nonlinear systems using the extended SIMNON [11 and 12] to first perform equilibrium finding and linearization, followed by the design methods outlined above.

The lead-lag compensator design rule base described here together with the material in companion references [3-6] document our effort to create a higher-level, more supportive environment for computer-aided control engineering. We have reached the point where the basic concept is concretely defined, the overall structure has been implemented, and enough capabilities have been incorporated to solve a useful class of problems and to illustrate the promise of such a system. However, we still have a substantial amount of work to be done in order to achieve a complete working system for the larger domain of activity outlined in the paragraph above and in [3-6].

ACKNOWLEDGMENTS. The research reported in this paper has been substantially aided by the contributions of others. Dr. Piero Bonissone has assisted from the beginning of the project, providing the initial inference engine and his experience with expert systems development. More recently, Dr. Melvin Simmons, Dr. Dale Gaucas, and Steven Kirk assisted in implementing the expert system using the DELPHI inference engine. Also, David Kassover and Neal Lassinger helped in making required modifications to CLADP. Their support is gratefully acknowledged.

REFERENCES

1. Edmunds, J.M., 1979, "Cambridge Linear Analysis and Design Program," *Proc. IFAC Symposium on CAD of Control Systems*, Zurich, Switzerland.

2. Hayes-Roth, F., Waterman, D.A., and Lenat, D.B., 1983, *Building Expert Systems*, Addison-Wesley, Reading, MA USA.

3. Taylor, J.H. and Frederick, D.K., 1984, "An Expert System Architecture for Computer-Aided Control Engineering," *IEEE Proceedings 72*, 1795-1805.

4. Taylor, J.H., Frederick, D.K., and MacFarlane, A.G.J., 1983, "A Second-Generation Software Plan for CACSD," *Abstr. IEEE Control System Society Symposium on CACSD*, Cambridge, MA USA.

5. Taylor, J.H., Frederick, D.K., and James, J.R., 1984, "An Expert System Scenerio for Computer-Aided Control Engineering," *Proc. American Control Conference*, San Diego, CA USA.

6. James, J.R., Taylor, J.H., and Frederick, D.K., 1985, "An Expert System Architecture for Coping With Complexity in Computer-Aided Control Engineering," to appear in *Proc. 3rd IFAC Symposium on CAD in Control and Engineering Systems*, Lyngby, Denmark.

7. D'Azzo, J.J. and Houpis, C.H., 1981, *Linear Control System Analysis and Design, Conventional and Modern*, Second Edition, McGraw-Hill, New York, NY USA.

8. Thaler, G.J., 1973, *Design of Feedback Systems*, Dowden, Hutchinson, and Ross, Stroudsburg, PA, USA.

9. Dorf, R.C., 1980, *Modern Control Systems*, Addison-Wesley, Reading, MA USA.

10. Johnson, H.E. and Bonissone, P.P., 1983, "Expert System for Diesel Electric Locomotive Repair," *Journal of FORTH Applications and Research 1*, 7-16.

11. Elmqvist, H., 1977, "SIMNON – An Interactive Simulation Program for Non-Linear Systems," *Proc. Simulation 77*, Montreux, France.

12. Taylor, J.H., 1982, "Environment and Methods for Computer-Aided Control Systems Design for Nonlinear Plants," *Proc. Second IFAC Symposium: CAD of Multivariable Technological Systems*, Purdue University, West Lafayette, IN USA, 361-367.

13. Doyle, J.C. and Stein, G., 1981, "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis," *IEEE Transactions on Automatic Control, AC-26*, 4-16.

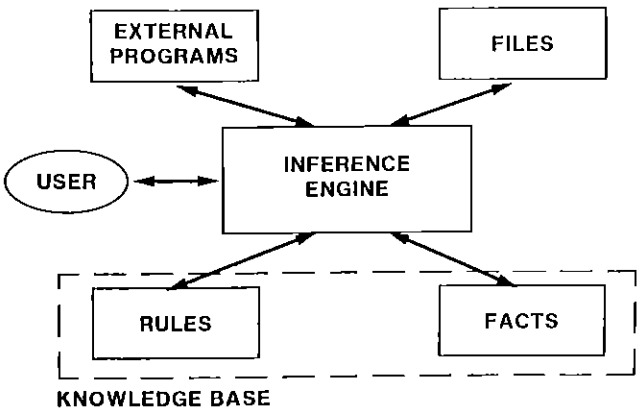

Figure 1. Elements of CACE-III

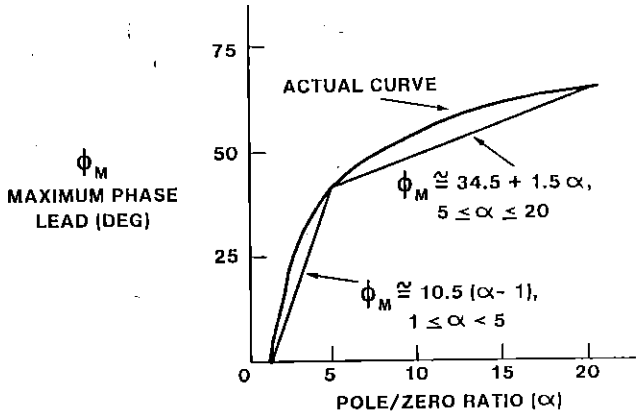

Figure 3. Approximation of Maximum Phase Lead Versus α Relation

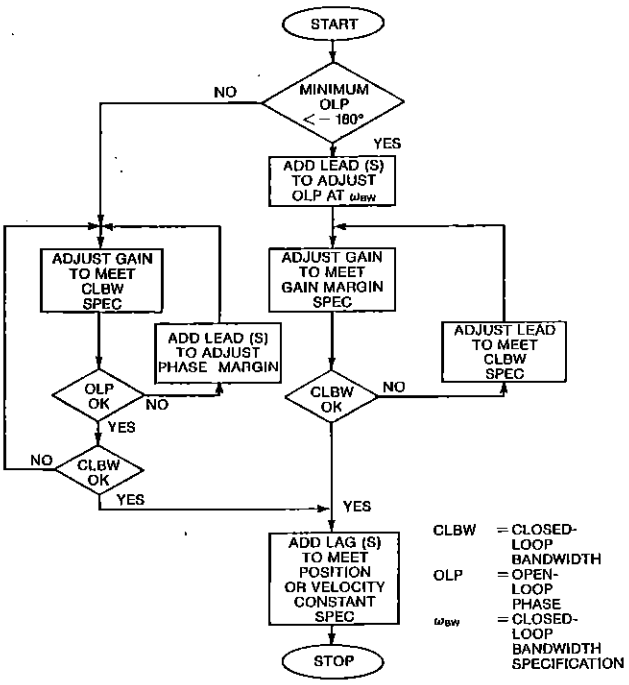

Figure 2. Lead-Lag Design Algorithm

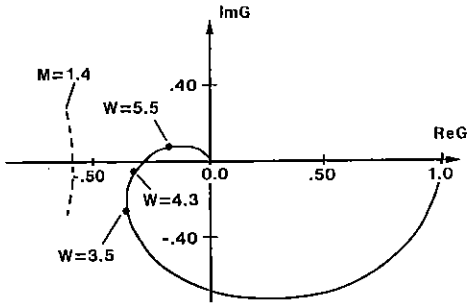

Figure 4. Nyquist Plot of Uncompensated Plant

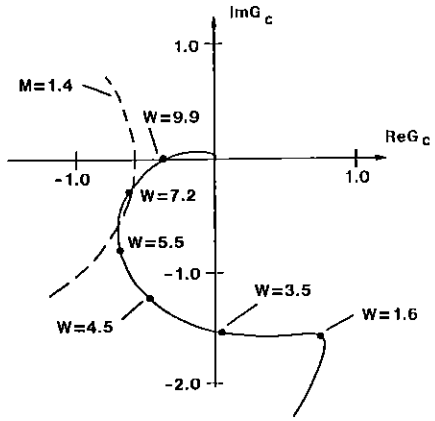

Figure 5. Nyquist Plot of Compensated Forward Path

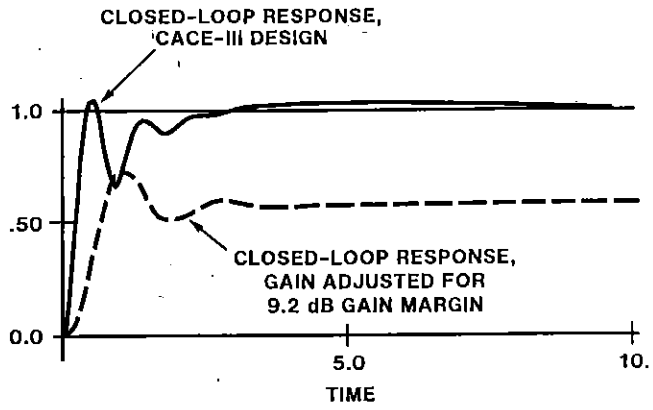

Figure 6. Closed-loop Time Responses

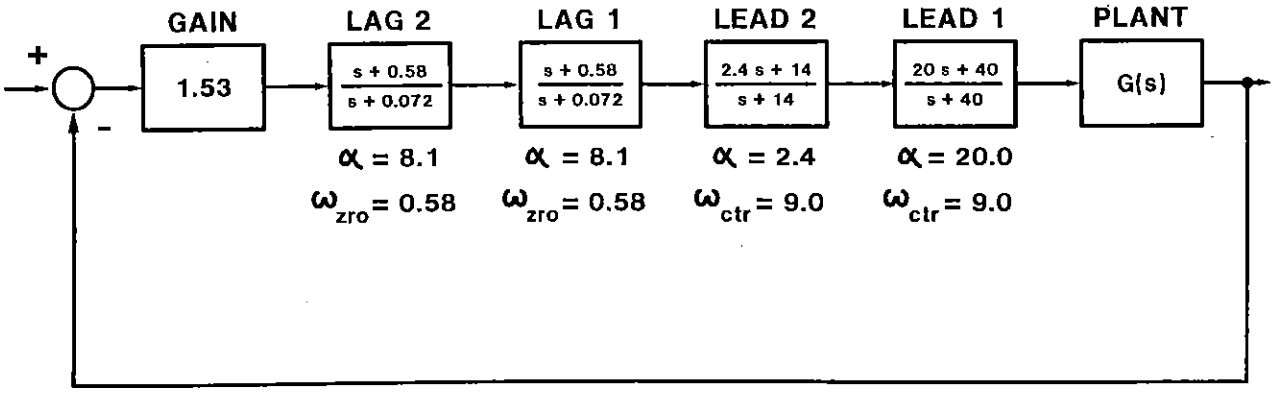

Figure 7. Block Diagram of Compensated System

$$G(s) = \frac{500(s+10)}{(s+2)\ (s+5)\ (s+3+j4)\ (s+3-j4)\ (s+20)}$$