Use of expert-systems programming techniques for the design of lead-lag compensators

J.R. James, PhD
D.K. Frederick, PhD
J.H. Taylor, PhD

Indexing terms. Poles and zeros, Phasing and phase control, Modelling, Computer simulation

Abstract: The authors discuss the application of expert systems programming techniques to the design of lead-lag compensators for a linear, 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. The heuristic has been implemented using the production rule knowledge representation; examples of production rules are given and their use is discussed. Inference-engine capabilities and extensions to a conventional analysis and design package, required to implement this automatic control system design method, are also outlined.

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. Similarly, although synthesis methods may appear to be straightforward, there are often subtle points in applying a particular procedure that are fully appreciated only 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, owing 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 [1]) to complete an iterative design that meets specifications on closed-loop bandwidth, gain margin and low-frequency gain (position or velocity error constant) or closed-loop bandwidth, phase margin and low-frequency gain. The 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 [2], we call it CACE-III

The use of expert systems to represent and apply expert knowledge in a particular domain is increasingly widespread (cf. Hayes-Roth [3]). Our recent effort [2, 4–7] 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. The expert system consists of an inference engine that applies the data in a knowledge base, to aid the user in running external programs to achieve design specifications (Fig. 1)

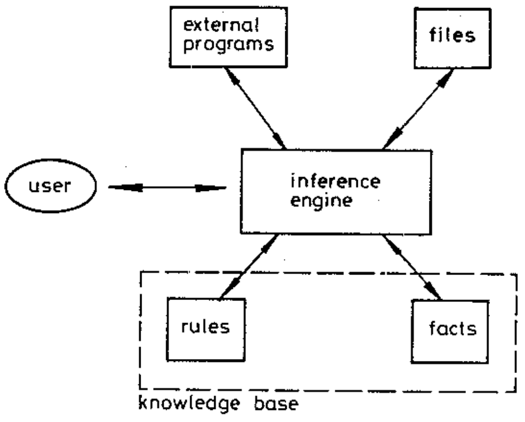

Fig. 1 *Elements of CACE-III*

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 (SISO) linear plant. We will present an overview of one application of this rule base. More detailed presentations of this example and other applications of the expert system to

Paper 5290D (C8), first received 2nd December 1985 and in revised form 13th January 1987

Dr. James is with the Electrical Engineering Department, United States Military Academy, West Point, NY 10996, USA
Dr Frederick is with the Electrical, Computer & Systems Engineering Department, Rensselaer Polytechnic Institute, Troy, NY 12181, USA
Dr. Taylor is with the General Electric Company, Corporate Research & Development, Schenectady, NY 12345, USA

compensate a sampled-date system and nonlinear system can be found in Reference 7.

The knowledge representation scheme which we use is the production rule or *recognise-act* formalism (see Reference 2). 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 a knowledge representation and inference mechanisms are contained in References 2, 3, 5, 6 and 7 and Bonissone and Johnson [8]. Examples of production rules used in CACE-III and a discussion of the use of production rules to mimic a train of thought are provided in Section 4

In Section 2 we present the lead-lag compensator design heuristic for SISO linear continuous time systems, and, in Section 3, we provide an example of using the heuristic to compensate a fifth-order type-zero system. Section 4 contains examples of the production rules used in CACE-III and provides some thoughts on the benefits of using production rules to represent and apply knowledge of this type. Section 5 outlines the requirements for an inference engine to implement the lead-lag design heuristic. Section 6 deals with modifications and extensions made to CLADP to integrate it with the inference engine. We conclude in Section 7 with comments regarding the status and future of this effort.

2 Lead-lag design heuristic

We chose classical SISO frequency-domain design techniques for our initial study, with the expectation that well established rules exist for its application. The goal was to have the expert system utilise CLADP to achieve specifications of gain margin, bandwidth and low-frequency gain by the addition of lead-lag compensation to a SISO 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 numerious textbooks present in detail the results of adding lead or lag compensation (cf. D'Azzo and Houpis [9]), 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 given by Thaler [10]; 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 or Nichols 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, or phase 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)

2.1 *The design heuristic*

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

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

(i) If the minimum open-loop phase angle is greater than $-180°$, the gain margin is infinite, so phase margin

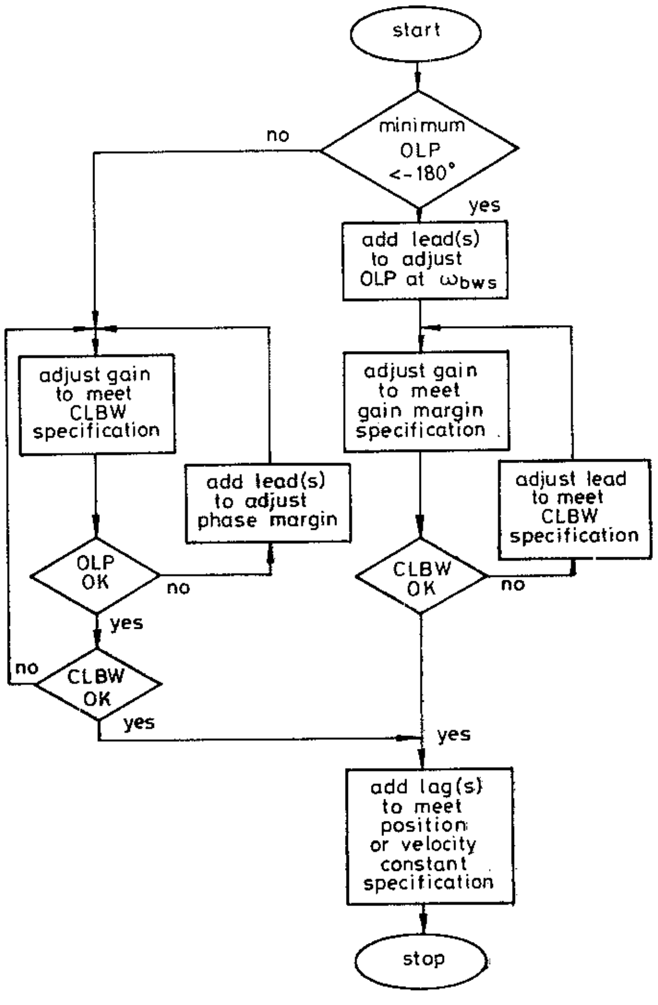
start
minimum OLP <-180°
no
yes
add lead(s) to adjust OLP at ω_{bws}
adjust gain to meet CLBW specification
adjust gain to meet gain margin specification
add lead(s) to adjust phase margin
adjust lead to meet CLBW specification
OLP OK
no
yes
CLBW OK
no
CLBW OK
no
yes
yes
add lag(s) to meet position or velocity constant specification
stop

Fig. 2 *Lead-lag design heuristic*
CLBW = closed-loop bandwidth
OLP = open-loop phase
ω_{bws} = closed-loop bandwidth specification

should be used to guide the design process. The high-frequency portion of the frequency response is adjusted to meet phase-margin and bandwidth specifications (ϕ_M, ω_{bws}) as follows:

(*a*) Estimate the compensated magnitude crossover frequency (ω_{ecco}, the frequency at which the compensated system magnitude is to be unity or 0 dB) needed to meet the specified closed-loop bandwidth by determining the actual uncompensated open-loop magnitude crossover frequency ω_{uco} and actual uncompensated closed-loop bandwidth with unity feedback ω_{ubw}, and making a logarithmic extrapolation: $\omega_{ecco} = \omega_{uco}\,\omega_{bws}/\omega_{ubw}$. For type-zero systems, a constant-gain precompensator with gain greater than 1.0 may have to be added to have the open-loop gain exceed a magnitude of 1.0, because the low-frequency magnitude approaches a constant. Consideration of the Nichols plot indicates that this approach should work well, but we have not tested it for a variety of systems. In any event, whatever value of the initial estimate of the compensated crossover frequency is arrived at, that value is subsequently adjusted (see below) until a compensated open-loop magnitude crossover frequency is achieved which causes the actual closed-loop bandwidth to be close enough to the specified value.

(*b*) A single stage of lead compensation is defined by a lead gain K_{lead}, a pole/zero ratio α_{lead} and a lead centre frequency ω_{ctr}, as in eqn. 1. The maximum permitted value of α_{lead} is 20, which corresponds to a maximum added phase of 64.8°; one or two leads may be added, and so the total added phase that can be inserted by the expert system is just under 130°. The lead compensator parameters are calculated so that the gain of the compensated system at the estimated compensated crossover frequency ω_{ecco} is unity, and enough phase is added at that frequency such that the phase margin specification ϕ_M is met. The method to achieve this is discussed by Sinha [11].

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

(*c*) Iterate the value of ω_{ctr} in increments of 10% of the estimated compensated magnitude crossover frequency ω_{ecco}, increasing ω_{ctr} to increase the closed-loop bandwidth and decreasing ω_{ctr} to decrease the bandwidth.

(*d*) When the phase margin and closed-loop bandwidth specifications are met, then proceed with low-frequency adjustment using lag compensators.

(ii) If the minimum open-loop phase angle is less than $-180°$, both gain-margin and closed-loop bandwidth specifications (G_M, ω_{bws}) will 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_{bws}. The centre frequency of the lead ω_{ctr} is placed at ω_{bws}. The amount of phase lead initially added is calculated to place the forward-path phase at ω_{bws} 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 Nyquist plots for type-1 systems tend to approach the origin closer to the negative real axis than type-0 systems, and so additional lead is required to maintain an appropriate distance from the critical point. The pole/zero ratio of the lead network α_{lead} which is needed to achieve the required amount of phase lead is calculated using a piecewise-linear approximation to the expression

$$\phi_M = \sin^{-1} \frac{\alpha_{lead} - 1}{\alpha_{lead} + 1} \tag{2}$$

(see, for example, Reference 12, Eqn 10.11. Note that the product of the gain K_{lead} and this pole/zero ratio is the high-frequency gain of the lead compensator. One or two stages of lead are used as 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_{bws} by 10°, when the bandwidth is too small, and decrease phase lead by 10° when the bandwidth is too large.

(*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.

Typical uncompensated and compensated Nichols plots for type-1 and type-0 systems are given in Figs. 3 and 4.

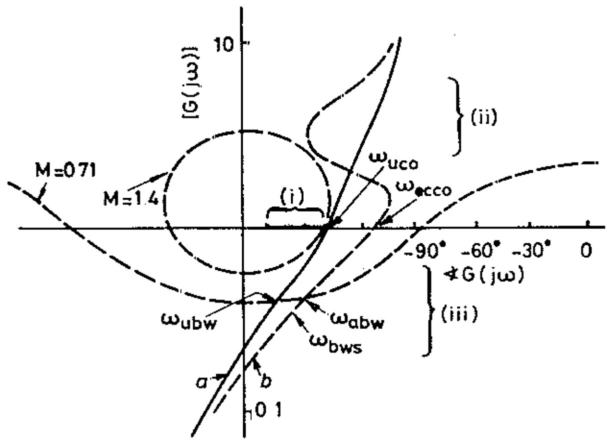

Fig. 3 *Uncompensated and compensated type-1 Nichols loci*
ω_{bws} = bandwidth specification
ω_{ecco} = estimated crossover frequency (compensated)
ω_{ubw} = uncompensated bandwidth
ω_{uco} = uncompensated crossover frequency
ω_{abw} = actual bandwidth (compensated)
a Uncompensated Nichols locus
b Compensated Nichols locus
(i) Adjust open-loop phase at ω_{bws} to lie in this range
(ii) Low-frequency adjustment placed well below ω_{ecco}
(iii) High-frequency adjustment based on the specified closed-loop bandwidth ω_{bws}

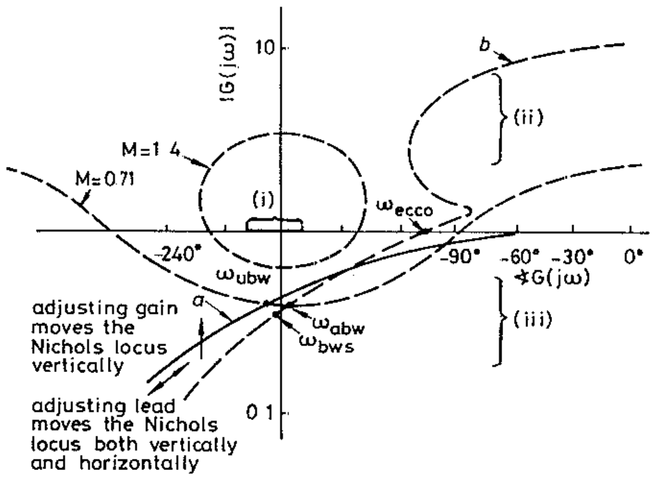

Fig. 4 *Uncompensated and compensated type-0 Nichols loci*
ω_{bws} = bandwidth specification
ω_{ecco} = estimated crossover frequency (compensated)
ω_{ubw} = uncompensated bandwidth
ω_{uco} = uncompensated crossover frequency
ω_{abw} = actual bandwidth (compensated)
a Uncompensated Nichols locus
b Compensated Nichols locus
(i) Adjust open-loop phase at ω_{bws} to lie in this range
(ii) Low-frequency adjustment placed well below ω_{ecco}
(iii) High-frequency adjustment based on the specified closed-loop bandwidth ω_{bws}

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

$$G_{lag}(s) = \frac{s + \omega_{zlg}}{s + \omega_{zlg}/\alpha_{lag}} \tag{3}$$

where ω_{zlg} is the frequency associated with the zero of $G_{lag}(s)$, and α_{lag} is the zero/pole ratio and is equal to the low-frequency gain. A maximum of two lags can be added by the expert system:

(i) For a type-0 system, the zero of the lag is placed a decade below the estimated compensated magnitude-crossover frequency ($\omega_{zlg} = \omega_{ecco}/10$) to make the high-frequency impact of the lag negligible. The pole of the lag is determined by the amount of added low-frequency gain required to meet the specification, which is α_{lag}

(ii) For type-1 systems, the zero of the lag compensator is placed an additional octave below the compensated magnitude-crossover frequency, making $\omega_{zlg} = \omega_{ecco}/20$. This modification corrects for the more pronounced effect that a lag compensator has on magnitude-crossover conditions for type-1 systems compared with type-0 systems owing to the increased low-frequency phase lag of a type-1 system.

(iii) To see why the lag for a type-1 system must be placed at a lower frequency, consider the Nichols charts for a compensated type-1 system (Fig. 3) and a compensated type-0 system (Fig. 4). The phase of the type-1 system tends to $-90°$ as the frequency tends to 0 rad/s The phase of the type-0 system, however, tends to 0° as ω tends to zero. Thus, as the frequency increases from a low value, the Nichols locus for a typical compensated type-1 system (Fig 3) tends to move closer to the $-180°$ axis than the locus for a compensated type-0 system (Fig. 4) The result is a more pronounced effect of the lags on the type-1 system locus, in the vicinity of the intersection of the Nichols locus with the -3 dB closed-loop contour ($M = 0.71$), which governs the closed-loop bandwidth

(iv) Fig. 3 and 4 also show the reason why the open-loop phase angle at the desired closed-loop bandwidth ω_{bws} is adjusted to a different range for type-0 systems than for type-1 systems. It is seen that, for type-1 systems, the Nichols locus tends to cross the $M = 0.71$ contour at a greater value of phase (due to the increased slope of the type-1 locus) than does the Nichols locus for type-0 systems Thus, the open-loop phase at the desired bandwidth frequency must be adjusted to lie in a more positive range of values (-140 to $-170°$ instead of -170 to $-200°$) to cause the locus of type-1 systems to cross the $M = 0.71$ contour at the desired frequency.

The tolerances for declaring specifications to be met are: bandwidth $\pm 20\%$, gain margin ± 3 dB, and low-frequency gain or velocity constant ± 3 dB. The rules for the initial adjustment of the open-loop phase angle are structured to cause lead precompensators to be added, to place the open-loop phase angle at the desired closed-loop bandwidth in ranges that approximate the bandwidth specification Additional rules make subsequent adjustments to the values of α_{lead} until the bandwidth and gain-margin specifications are within tolerances

2.2 Discussion of the heuristic

As discussed, the placement of the lead centre frequency and the amount of phase lead added are calculated using a heuristic derived from a study of the Nichols locus An alternative scheme to place the lead centre frequencies ω_{ctr} as a multiple of the closed-loop bandwidth, with 15 dB of gain margin, has been tried for several ranges of values for a third-order, type-1 system, and it worked well for the example to be given below. It appears that this approach would work for a wider variety of plants than the one currently used in the expert system. Also, an alternative to calculating the initial amount of phase lead to add, based on placing the phase at the desired closed-loop frequency in a range of values, would be to apply Sinha's method for a nominal phase margin. Regardless of the approach used to pick an initial adjustment, the subsequent adjustment of the compensation to meet specifications would proceed as described in this paper.

In most cases tried, especially where there are lightly damped poles and/or zeros, this algorithm has been successful. For systems with very lightly damped poles or zeros, however, the algorithm may not initially place the leads at the appropriate frequencies, and so the iteration scheme cannot converge to meet the bandwidth and gain-margin specifications. The resonant-mode case is known to be difficult, often requiring the design of compensators with complex zeros [10].

As indicated above, we use frequency-domain specifications to determine lead, lag and gain parameters in our control system design. 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 percentage overshoot to a step input [9]. There is also an approximate relationship between the phase margin of the system and the percentage overshoot of the step response [12]. An exact relation exists between the low-frequency gain and the steady-state error to a step or ramp input. These types of relations can provide the basis for supporting other combinations of specifications, in both the time and frequency domain.

CACE-III currently supports the entry of a wide variety of performance specifications, in both domains. However, rules which can make use of the above relations to translate other specifications into those used in the design heuristic have not been implemented. Another extension that would be required to support this capability would be rules to check that the original specifications had indeed been met when the derived frequency-domain specifications were satisfied. A truly flexible specification system that would allow the user to input an arbitrary combination of specifications, and then check for consistency and completeness and translate them into the specifications needed for our design heuristic, would probably be quite a challenge.

There are currently approximately 300 rules in CACE-III, and running a complete design problem such as illustrated in the example below requires about 30 min 'clock time' (not CPU time) on a lightly loaded VAX 11/785 About half of this time is required to 'compile' the rules as we switch among the rule bases (e g the supervisor, specification and design rules); this time could be eliminated if DELPHI supported storing and accessing compiled versions of the rules. ('Compiled' refers to loading the rules into the data structure used by the inference engine)

3 Example application

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

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

with the design specifications of 9.0 rad/s 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 (Fig. 5) As the design proceeds, the user is informed of the parameters of the lead, gain and lag precompensators being added. A Nyquist plot of the compensated system (Fig. 6) is provided when the design is complete. The step response of the compensated system (Fig. 7) is also provided if the user requests it. The complete closed-loop system is portrayed in Fig. 8. The results achieved were 9.3 rad/s bandwidth, 8.5 dB of gain margin, and 40.0 dB of low-frequency gain. Each of these falls within the stated tolerance.

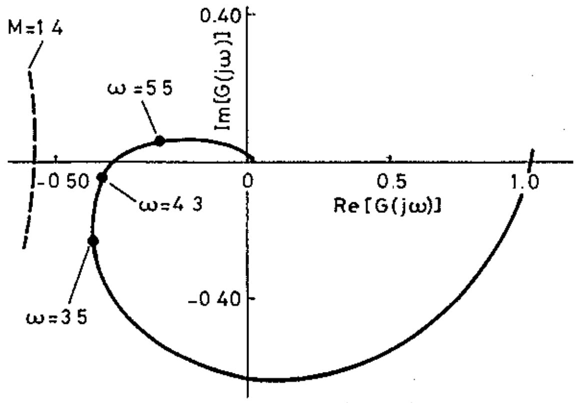

Fig. 5 *Nyquist plot of uncompensated plant*

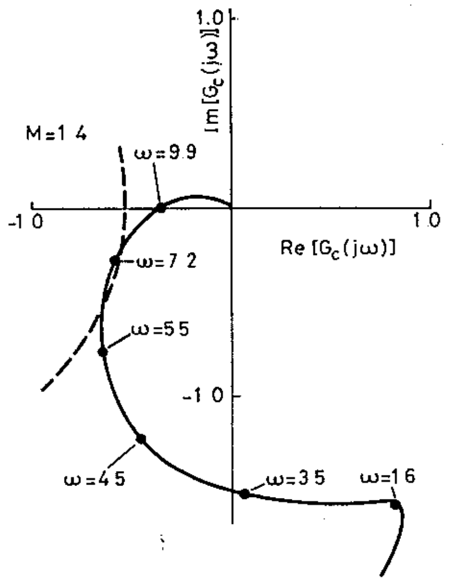

Fig. 6 *Nyquist plot of compensated forward path*

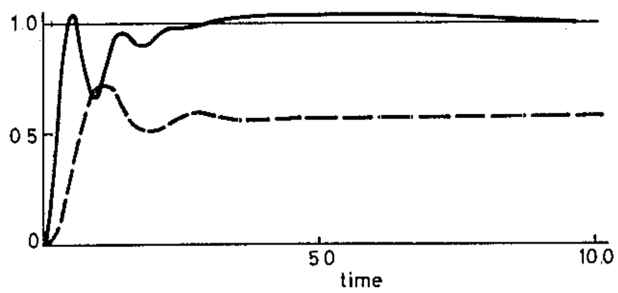

Fig. 7 *Closed-loop system step responses*
——— closed-loop response. CACE-III design
– – – – closed-loop response, gain adjusted for 9.2 dB gain margin

4 Why an expert system?

Considering the flow chart in Fig. 2, a question which may arise is: Why not implement the logical flow indicated in this Figure in a conventional structured programming approach using C, Pascal, Fortran, or some other high-level language? This could have been done, but we believe that such an implementation would suffer from the problems of *complexity*, and *difficulty of development and application*, mentioned in the introduction. We will make these issues more tangible by illustrating how we have used the expert systems approach to deal with them.

The management of complexity is very important, both for the user and the developer. This is accomplished in a rule-based system by separating the knowledge from the execution of reasoning or inference. For instance, the rule that is used to recognise that the high-frequency adjustment is successfully completed is given in Table 1. The three clauses in the premise of the rule (those elements between WHEN and THEN) must all be satisfied for the rule to be applied ('fired'). When these conditions are satisfied, the three actions in the conclusion portion (those elements after THEN) are performed. The second action of the conclusion causes the fact:

(LOW-FREQ ADJUSTMENT . REQUESTED)

to be written into the set of facts, causing the premise of the rule which will begin the execution of the lag precompensator design heuristic to be satisfied. All this is transparent, without a need to be famililiar with the overall program.

An important side-benefit of this formulation is the ability to query the system, to determine the flow of logic. For example, at the point corresponding to the rule fired by the fact (LOW-FREQ ADJUSTMENT . REQUESTED), the user can ask *why* and the rule will be displayed, together with the assignment of values to variables which were used to select the rule. A second query of *why* will cause the rule of Table 1 to be displayed, and so forth. In this manner, the developer or user can check the logic embodied in the set of rules. In addition, the *trace* function can be set to display each inference as it is made. In contrast, if the flow chart of Fig. 2 had been implemented in a conventional program, the user would not have been able to query the system to check on the logical flow of the design heuristic. Thus, the focus is on enabling the user to understand what is happening as the heuristic is applied, instead of implementing the heuristic in the fewest number of lines of code. It should be noted that extensive *help* systems have been constructed for other applications of the expert system approach (cf. Reference 8). A similar capability could be included here to

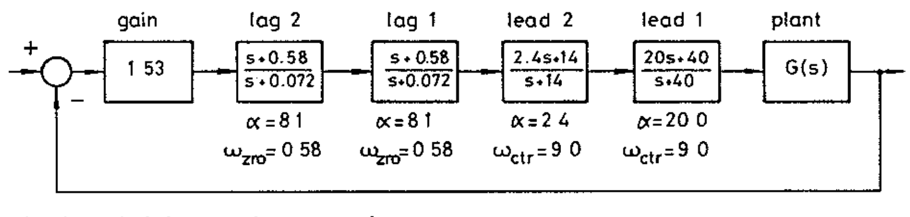

Fig. 8 *Block diagram of compensated system*

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

Table 1: Rule to detect successful high-frequency adjustment

```
(Rule_360 ('You now need to adjust the low-frequency response')
               (WHEN
                          (TRUE (PHASE-AT-OMEGA-BW VALUE . ADJUSTED)
                            'Phase at desired closed-loop bw is adjusted')
                          (TRUE (EXCESS-GAIN-MARGIN VALUE . -3< . < +3DB)
                            'Gain margin is OK')
                          (TRUE (EXCESS-CLOSED-LOOP-BW VALUE  -del< . < +delRPS)
                            'Closed-loop bandwidth is OK'))
               (THEN
                          (CLEAR (EXCESS-CLOSED-LOOP-BW VALUE  -del< . < +delRPS)
                            1.0
                            'Prevent looping on this rule')
                          (WRITE (LOW-FREQ ADJUSTMENT . REQUESTED)
                            1 0
                            'Now adjust the low-frequency Nichol s locus )
                          (WRITE (CLOSED-LOOP-BW VALUE . ADJUSTED)
                            1.0
                            'Closed-loop bandwidth has been adjusted')))
```

further clarify for the user what actions were being taken and why.

Another major benefit of the expert system approach is *extensibility*. The rule-based system functions by either the forward or backward interpreter establishing new facts (e.g. (GAIN-MARGIN DB-VALUE . 8.6)). These facts can be used to trigger complicated responses to the current situation, and are the basis for being able to *incrementally extend* the knowledge stored in the rules, without restructuring the top-level control of the program. For instance, the rule which recognises that the velocity constant specification is too large to be met with two lags, returns the user to the specification set of rules to decrease the specification (Table 2). This rule was added a year after the basic flow of Fig. 2 was implemented, but did not require any changes to other rules that had already been written. The only requirement is that the premise of the rule be satisfied.

Table 2: Rule to decrease the velocity-constant specification

```
(Rule_710 ('type-1, you need to decrease the spec for velocity const')
           (WHEN
                            (TRUE (PHASE-AT-OMEGA = BW VALUE . ADJUSTED)
                              'The phase at the desired BW has been adjusted')
                            (TRUE (EXCESS-GAIN-MARGIN VALUE . -3< . < +3DB)
                              'The gain margin is OK')
                            (TRUE (EXCESS-VELOCITY-CONSTANT VALUE   <-55DB)
                              'Cannot meet vel cons spec with two lags')
                            (TRUE (DESIGN-FACTS VALUES . ALL-ENTERED)
                              'Design facts are known'))
           (THEN
                            (CLEAR (DESIGN-FACTS VALUES . ALL-ENTERED)
                              1.0
                              'Prevent looping on this rule')
                            (ACKNOWLEDGE (DECREASE VC-SPEC  ACKNOWLEDGED)
                              1 0

                              'You cannot meet the specification with two lags.
                              We recommend that you decrease the specification
                              for velocity constant  We will return you to the'
                              specification development rule base. )
                            (CLEAR (EXCESS-VELOCITY-CONSTANT VALUE   <-55DB)
                              1.0
                              'Prevent firing this rule )
                            (PROG (MOVE-TO-SUPER DECREASING . VC-SPEC)
                              1.0
                               (prog ( )
                                (switch  super.rul) ))))
```

We believe that the above differences between an expert system and conventional programming techniques provide sufficient reason to prefer the use of production rules over a procedural programming approach. This is not related to the issue of the language used to implement the inference engine, which could be coded in any language. Prolog has found favour with many AI researchers, and inference engines have been written in Pascal, Fortran, Basic, and C. The Delta inference engine that was originally used to implement CACE-III was written in Forth Lisp, however, is designed for symbolic processing, and the Delphi inference engine currently used for CACE-III is written in VAXLisp, a variant of Common Lisp. It should be noted that VAXLisp supports direct linking of a shareable image (compiled and linked from any language supported by VAX/VMS) into the Lisp code, in addition to creation of VAX/VMS processes which can run in parallel with the Lisp program.

5 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 [3]. It has been our experience that the following capabilities define the mainimum set needed to implement a logical flow of the control system design process:

(*a*) arithmetic functions to support numerical operations and tests within the inference engine

(*b*) use of logical variables which can be assigned either numerical or symbolic values

(*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 proprietary Delphi expert system shell, which is written in VAXLisp and has the required capabilities. Delphi has a PROG verb which provides access to the Lisp prog feature and thus supports numerical calculations, and it provides the capability to run external processes by creating a VAX/VMS subprocess which can be run through VMS mailboxes. The Lisp environment was also easily modified to create macro instructions for CLADP and to implement a variety of functions which support exchange of numerical and symbolic data.

These capabilities are used as follows: the expert system starts, formulates the problem, and, when required, runs CLADP. To run CLADP, CACE-III issues the appropriate commands, reads the results and assigns the values determined by CLAPD routines to logical variables, so that it can calculate parameter values for the leads and lags. The results of this design activity are then automatically analysed 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 Reference 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) [8]. 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 insert precompensators and 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 the expert system, and all transactions with CLADP or other programs are handled automatically.

6 CLADP modifications and extensions

The application of the lead-lag design heuristic by the expert system required that a method be established to transform the symbolic determination that some information was needed from CLADP into a sequence of commands, which could be used by CLADP, and transform the data that CLADP produced into a symbolic form which could be used by the expert system. As noted in Section 5, the Lisp environment was used to create a set of macro instructions for CLADP which contain the sequence of commands appropriate to performing a characteristic locus plot, finding the open-loop phase angle at a desired frequency, finding the open-loop magnitude crossover frequency, gain margin, phase margin, and so forth. To facilitate this, certain commands and macros were added to CLADP. It would have been possible to create special Lisp functions to interpret the data produced by CLADP and create the information (facts or three-tuples) used by the inference engine to continue the inference process. However, because the CLADP source code was available and familiar to us, we decided to modify CLADP to produce the appropriate facts, in response to commands received from CACE-III.

The CLADP modifications 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. It should be noted that substantial work on the development of CLADP commands and macros, as well as early work on the development of the lead-lag design heuristic, was done by Lassinger [13]. The following new commands were added to the characteristic locus display (CLD) routine of CLADP to obtain data once 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, by interpolation (The CLADP command EVAL was not employed, because its use involves returning to the CLADP Supervisor, invoking another routine, and returning to the CLD routine.)

(*b*) NMAG determines the number of occurrences of a specified open-loop magnitude and supplies the frequency and phase at each occurrence; DBMA serves the same purpose for a magnitude specified in decibel form

(*c*) CMAG determines the number of occurrences of a specified closed-loop magnitude, and calculates the frequency and phase at each occurrence; DBCM is the decibel version of this command

(*d*) NPHA determines the number of occurrences of a specified phase, and calculates the magnitude (in both decibels and absolute value) and the frequency of each occurrence

(*e*) GM calculates the $-180°$ crossover frequency and the corresponding gain margin in both decibels and absolute value

(*f*) PM calculates the magnitude-crossover frequency and the corresponding phase margin in degrees

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

(i) GAIN creates a gain compensator having the gain specified either in decibels or as a real number

(ii) LEAD creates a lead compensator according to eqn. 1, given α_{lead} and ω_{ctr}

(iii) LAG creates a lag compensator according to eqn. 3, given α_{lag} and ω_{zlg}.

Once the above features had been added to implement the automatic lead-lag compensator design heuristic in CACE-III, it became apparent that these commands are 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 of defining compensator transfer functions in terms of the order and coefficients of the numerator and denominator polynomials. Note that these functions cannot be implemented directly as CLADP macros (batch files) because of the need to perform calculations to evaluate polynomial coefficients in terms of the parameters in eqns 1 and 3

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 units of information used by DELPHI (see Tables 1 and 2). As noted above, the same 'signal-to-symbol' transformation could have been done by writing Lisp functions to decipher the conventional CLADP output.

7 Conclusion

Many considerations involved in the development of an expert system to aid a control engineer in the design of lead-lag compensation for a single-input/single-output continuous-time linear plant have been presented. Other issues, such as the implementation of a partitioned rule-base architecture and the co-ordination of the symbolic manipulations of the inference engine, with numeric calculations being performed by conventional software, are described in Reference 6. The limitations of the design heuristic 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 only major deficiency is the inability to handle resonant modes in a satisfactory manner. The difficulty is caused by the rapid change in phase angle near the resonant frequency. We plan to expand the rule base so that CACE-III 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 [10].

This presentation details an element of a larger expert system [2, 4–7]. The architecture for this expert system has been designed to provide support to the user in modelling, 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 design rule base; we developed this with two goals in mind: to solve a problem that has importance in terms of current practice and to serve as an easily accessible model for expert-aided design in general. The larger expert system is based on using both CLADP and an extended version of SIMNON as the underlying conventional software (see Elmqvist [14] for SIMNON, and Taylor [15, 16] for the extensions). The latter serves to model and simulate the nonlinear plant, find equilibria and determine linearised models to be used 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 the design of:

(*a*) two-loop digital control systems with spectral separation (fast inner loop, slow outer loop)

(*b*) multiple-input/multiple-output control systems using frequency-domain approaches [1]

(*c*) multiple-input/multiple-output control systems using the LQR/LTR method of Doyle and Stein [17]

(*d*) the diagnosis of nonlinear systems [18].

The lead-lag compensator design rule base described here, together with the material in companion references [2, 4–7] document our effort to create a high-level, 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 to achieve a complete working system for the larger domain of activity outlined in the paragraph above and in Reference 2.

8 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 Gaucus 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, as well as in heuristic development. Their support is gratefully acknowledged.

9 References

1 EDMUNDS, J.M.: 'Cambridge linear analysis and design program'. *Proc. IFAC Symp. CAD Control Syst.*, Zurich, Switzerland, 1979
2 TAYLOR, J.H., and FREDERICK, D.K.: 'An expert system architecture for computer-aided control engineering', *Proc. IEEE*, 1984, **72**, pp. 1795–1805
3 HAYES-ROTH, F., WATERMAN, D.A., and LENAT, D.B.: 'Building expert systems' (Addison-Wesley, 1983)
4 TAYLOR, J.H., FREDERICK, D.K., and MACFARLANE, A.G.J.: 'A second-generation software plan for CACSD'. *Abstr. IEEE Control Syst. Soc. Symp. CACSD*, 1983, Cambridge, MA USA **17**
5 TAYLOR, J.H., FREDERICK, D.K., and JAMES, J.R.: 'An expert system scenario for computer-aided control engineering'. *Proc. Am. Control Conf.*, 1984, San Diego, CA, USA, pp. 120–128
6 JAMES, J.R., TAYLOR, J.H., and FREDERICK, D.K.: 'An expert system architecture for coping with complexity in computer-aided control engineering'. *Proc. 3rd IFAC Symp. CAD Control & Eng Syst.*, Lyngby, Denmark, 1985, pp. 47–52
7 JAMES, J.R.: 'Considerations concerning the construction of an expert system for control system design'. PhD Thesis, Rensselaer Polytechnic Institute, Troy, NY, USA, 1986
8 BONISSONE, P.P., and JOHNSON, H.E.: 'Expert system for diesel electric locomotive repair', *Hum. Syst. Manage.*, 1984, **4**, pp. 255–262
9 D'AZZO, J.J., and HOUPIS, C.H.: 'Linear control system analysis and design, conventional and modern' (McGraw-Hill, New York, 1981, 2nd edn.)
10 THALER, G.J.: 'Design of feedback systems' (Dowden, Hutchinson, and Ross, Stroudsburg, 1973)
11 SINHA, N.K.: 'Control systems' (Holt, Rinehart and Winston, New York, 1986)
12 DORF, R.C.: 'Modern control systems' (Addison-Wesley, 1980)
13 LASSINGER, J.N.: 'Tools and algorithms for computer-aided design of single-input, single-output control systems' Automation Systems Laboratory Report, General Electric Co., Corporate Research and Development, Schenectady, NY, USA, 1984
14 ELMQVIST, H.: 'SIMNON—An interactive simulation program for non-linear systems'. *Proc. Simulation 77*, Montreux, France, 1977
15 TAYLOR, J.H.: 'Environment and methods for computer-aided control systems design for nonlinear plants'. *Proc 2nd IFAC Symp. CAD Multivariable Technol. Syst.*, Purdue University, West Lafayette, IN, USA, 1982, pp. 361–367
16 TAYLOR, J.H.: 'Computer-aided control engineering environment for nonlinear systems analysis and design'. *Proc. 3rd IFAC Symp. CAD Control & Eng. Syst.*, Lyngby, Denmark, 1985
17 DOYLE, J.C., and STEIN, G.: 'Multivariable feedback design: concepts for a classical/modern synthesis', *IEEE Trans.*, 1981, **AC-26**, pp. 4–16
18 TAYLOR, J.H., JAMES, J.R., and FREDERICK, D.K.: 'Expert-aided control engineering environment for nonlinear systems'. *Proc. IFAC 10th World Congress*, Munich, FRG, 1987