Expert Control

Åström, Karl Johan; Anton, John J.; Årzén, Karl-Erik

Published in:
Automatica

DOI:
10.1016/0005-1098(86)90026-9

1986

Automatica, Vol. 22, No. 3, pp. 277–286, 1986
Printed in Great Britain.

0005–1098/86 $3.00 + 0.00
Pergamon Journals Ltd.

Expert Control*

K. J. ÅSTRÖM†, J. J. ANTON‡ and K.-E. ÅRZÉN†

A novel approach to on-line control is based on a collection of diagnosis estimation design and control algorithms being orchestrated by an expert system.

Key Words—PID control; relay logic; auto-tuning; adaptive control; heuristics; expert system.

Abstract—There has been substantial progress in the theory and practice of automatic control through application of mathematical analysis and numerics. Symbolic data processing has, however, so far only had marginal influence on control systems despite the fact that the actual engineering of control systems contains a substantial amount of heuristic logic. This paper shows how the logic may be replaced by an expert system. This leads to simplification of conventional systems and makes it possible to obtain control systems with new capabilities.

1. INTRODUCTION

THE PURPOSE of this paper is to illustrate some uses of expert system techniques in control systems. It is first observed that the actual implementation of control laws often incorporates a substantial amount of heuristic logic. This is true for simple regulators as well as for more sophisticated multivariable control loops. Expert system methodologies provide a systematic approach for dealing with heuristic logic. Selected basic elements of an expert system are presented. An example illustrates how expert system ideas can be incorporated into feedback laws. It is shown that this approach can be used to simplify the heuristic logic and also to provide new functions in a control system.

The approach taken in this paper is to improve the primary control function by introducing an expert system as a part of the primary feedback loop. There are other applications of expert systems in the control field. Moore *et al.* (1984) have proposed an expert system which supervises a conventional control system and aids the operator by performing intelligent alarm functions and performance analysis. Trankle and Markosian (1985) have also used an expert system combined with a control design package to reconfigure control laws for a flight control system.

* Received 6 December 1984; revised 27 July 1985; revised 6 November 1985. The original version of this paper was presented at the 9th IFAC World Congress on A Bridge Between Control Science and Technology which was held in Budapest, Hungary during July 1984. The published proceedings of this IFAC Meeting may be ordered from: Pergamon Books Limited, Headington Hill Hall, Oxford, OX3 0BW, England. This paper was recommended for publication in revised form by Editor A. Sage.

† Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

‡ Reasoning Systems, Inc., Palo Alto, California, U.S.A.

2. THE PRACTICE OF AUTOMATIC CONTROL

There has been a very significant development of control theory over the past thirty years. This has led to many new ideas and concepts, as well as increased insight and new design procedures. The inspiration has largely come from two sources: mathematics and digital computing. However, this vigorous theoretical development has so far only had a modest impact on the practice of automatic control.

The making of a control system can be composed of the following activities: modelling, identification, analysis, simulation, control law design and implementation. It is fair to say that developments over the past 30 years have had a drastic influence on identification, analysis and design. Implementation has changed in the sense that digital systems are now replacing analogue systems. The basic system structure and the algorithms have, however, remained the same.

Although major progress has been made in linear and non-linear systems theory, there are several instances where theory lags application needs. Typical examples are systems with selectors and anti-wind-up mechanisms. Such systems are frequently approached purely empirically.

Consider for example an ordinary PID regulator. Its linear behaviour can be described by the model:

$$u(t) = k\left[e(t) + \frac{1}{T_i}\int^t e(s)\,ds + T_d\frac{de(t)}{dt}\right], \quad (1)$$

where $u(t)$ is the control signal and $e(t)$ the error signal. PID control can be understood very well from this linear equation, suitable values of the parameters can be determined etc. To obtain a good PID regulator it is also necessary to consider operator interfaces, operational issues like switching smoothly between manual and automatic operation, transients due to parameter changes, the effects of non-linear actuators, wind-up of the integral term, maximum and minimum selectors etc. An operational industrial PID regulator consists of an implementation of (1) and heuristic logic that takes care of these issues (Fig. 1). Although these

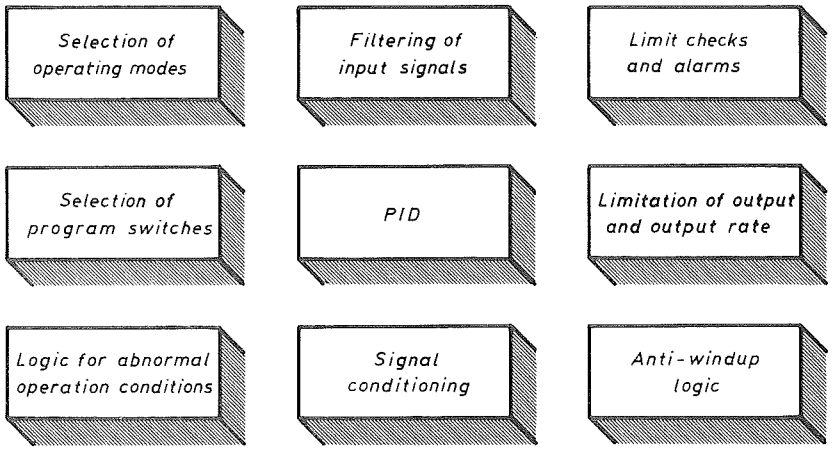

FIG. 1. Block diagram of an industrial PID regulator with heuristic logic.

heuristic factors are of extreme importance for good controller performance they have not attracted much interest from theoreticians. They are instead hidden in practical designs and rarely discussed in the control theory literature. One reason for this is that the theoretical analysis is quite difficult, another is that many researchers are unaware of these issues.

It can thus be concluded that practical solutions even to such mundane problems as PID control are not done by theory alone, but that heuristics play an important role. This shows up as logic that surrounds the implementation of the linear control law given by (1). The standard way of building industrial process control systems is to combine PID regulators with maximum and minimum selectors, logic and sequencing circuits. It is a very noticeable trend that DDC (Direct Digital Control) systems and PLC (Programmable Logic Controllers) systems are merging. The design of such combined systems also involves a lot of heuristics.

Heuristics are even more important in multivariable and self-tuning regulators. In these cases the fundamental control law is much more complicated than the control law given by (1). To obtain a well-functioning adaptive control system it is also necessary to provide it with a considerable amount of heuristic logic. This goes under different names such as supervision safety nets or safety jackets; see Clark (1981) and Isermann (1982). Experience has shown that it is quite time consuming to design and test this heuristic logic.

Although this paper concentrates on a simple control loop the authors believe that there are also applications for expert control at the higher levels of process control with many interacting components. At these higher levels it is possible to formulate the control problem in a dynamic programming framework. In such a formulation the "states" will be the steady state operating conditions between changes in the control laws applied to the lower level components. The states represent the operating conditions of the total plant. Decisions correspond to selections of operating conditions (control law type for a given loop, estimation etc.) carried out by lower level controls. There are also multiple plant-wide performance measures. Heuristics are important at both high and low levels. Dynamic programming formulations have not been widely used in practice due to computer requirements (Bellman, 1957). Some attempts have been made to use dynamic programming while making heuristic approximations to reduce computations and storage (Åström and Helmersson, 1983). Even so the methods have focused on low level system components, such as isolated loops.

3 HEURISTICS

In the previous section is was mentioned that heuristics plays a role in ordinary PID regulators and an even greater role in systems which combine PID regulators with sequencing and logic, and in adaptive regulators. Heuristics shows up in the form of selectors, *if-then-else* or *case* statements. This part of the code may be much larger than the code for the core control algorithm. The debugging, modification and testing of the control logic can therefore be very time consuming. From a purely pragmatic point of view of engineering efficiency it thus makes sense to look at more efficient ways of implementing the heuristic part of the code.

Once one accepts that control algorithms will contain heuristics one may also ask what a more extensive use of heuristics may contribute to control systems. As an illustration, consider typical adaptive algorithms such as model reference adaptive controllers or self-tuning regulators that are currently being used (Åström, 1983). These algorithms may be viewed as local gradient algorithms in the following sense. Starting from reasonably good *a priori* guesses of system order, sampling period

and parameters, the algorithms can adjust the regulator parameters to give a closed loop system with good performance. The algorithms are also capable of tracking a time-varying system provided that the parameters do not change too quickly. They will not work however, if the *a priori* guesses are too far off. Most digital adaptive algorithms will fail if the sampling period is too short. The present adaptive control algorithms thus have a limited range in which they will operate satisfactorily. Outside this range they may result in an unstable closed loop system. This is in fact the reason for introducing the safety jackets mentioned in Section 2.

There have recently been proposals for other types of tuning algorithms that have a wide range of operability, although possibly lacking the good local properties of the self-tuners (Åström and Hägglund, 1983, 1984). It seems appealing to explore the possibility of designing systems that combine a range of algorithms. This entails orchestration of different algorithms to achieve varying control objectives. Selection of different control structures is made in current control systems to a limited extent by hard-wired logic. If the scope is widened, e.g. by combining algorithms with different properties, this logic may grow unwieldy. The objective of expert control is to encode knowledge representations and decision capabilities to allow intelligent decisions and recommendations automatically rather than to preprogram logic which treats each case explicitly. Important analogues are found in existing expert system applications. It is thus useful to look there for tools that may aid in solving the control problem.

4 EXPERT SYSTEMS

Expert systems is a rapidly expanding area within the field of Artificial Intelligence (AI) (Winston, 1977; Nilsson, 1980; Barr and Feigenbaum, 1982; Davis, 1982; Davis and Lenat, 1982; Hayes–Roth *et al.*, 1983; Waterman and Hayes–Roth, 1983). One objective of AI is to develop computer-based models for problem solving. This is a distinctly different task from that of physical modelling via parameter identification. The AI based approaches attempt to model those aspects of a problem which are not naturally amenable to numerical representation or which can be more efficiently represented by heuristics. Expert systems seek to model the knowledge and procedures used by a human expert in solving problems within a well-defined domain. Important examples of expert systems are documented in Barr and Feigenbaum (1982), Davis (1982) and Nii *et al.* (1982).

Knowledge representation is a key issue in expert systems. Many different approaches have been attempted such as first order predicate calculus (logic), procedural representations, semantic networks, production systems (the AI versions of the if-then-else structure mentioned above) and frames, see e.g., IEEE (1983).

Production systems or rule based expert systems have some very desirable features for a process control application. A typical rulebased expert system has four principal components: (1) system data base; (2) rulebase; (3) inference engine and (4) user interface. These are illustrated in the block diagram in Figure 2.

The system data base

The system data base is the repository of facts, evidence, hypotheses and goals. For a process control example, the facts would include static data such as sensor measurement tolerances, operating thresholds, alarm level thresholds, constraints on operational sequencing, plant component configurations etc. Evidence includes dynamic data from sensors, instrument engineering reports and laboratory and test reports.

A practical observation for plant operations is that evidence as described above is typically diverse in type, often noisy, somewhat delayed, possibly incomplete and sometimes contradictory. An experienced process control engineer has techniques for dealing with these complications. He can develop hypotheses to supplement the current collection of facts. In an expert system, hypotheses are also generated and stored in the data base to cope with limitations in known facts or measured evidence. One important class of hypotheses in an expert control system will be the various state estimates made by parameter estimation algorithms. Including them under hypotheses acknowledges both that they are derived from evidence (data) and that their derivation is conditioned on other model assumptions.

Goals are other important entries in the data base. In an expert system they are usually both static and dynamic in nature. Static goals include the wide array of performance objectives as: maintain stable

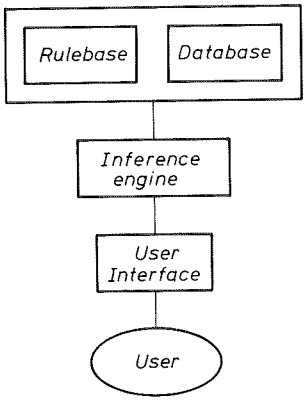

FIG. 2. A block diagram of a rulebased expert system.

operation, find optimal stationary operating points or determine if the current control law can be improved. Dynamic goals include those established on line, either by external command or from the program itself.

Rulebase

The rulebase contains the production rules which are typically described as: "if ⟨situation⟩ then ⟨action⟩". The "situation" represents facts, evidence, hypotheses and goals from the data base. In a conventional expert system the "action" can be to add a new entry to the data base or to modify other entries. In this case the action can also represent activation of a controller or an estimation algorithm. These actions are different from those found in conventional expert systems. The rules may be viewed as functions operating on the state. Since the data base is broader in concept than the usual notion of state in control theory, production rules are also richer in content than common transition functions. The rulebase is often structured in groups or knowledge sources that contain rules about the same subject. This simplifies the search. The knowledge sources are models for the essential problem elements. In a process control application they include the portion of the operator's skill to be automated, the control and estimation algorithms that may be applied, the appropriate characterization of these algorithms, judgemental knowledge on when to apply them and supervision and diagnosis of the system.

Inference engine

The purpose of the inference engine is to decide from the context (current data base of facts, evidence, hypotheses and goals) which production rules to select next. This can be done according to different strategies (Winston, 1977).

User interface

The user interface of a production system can be divided into two parts. The first part is the development support that the system gives. This contains tools such as rule editor and rule browser for development of the system knowledge base. The other part is the runtime–user interface. This contains explanation facilities that makes it possible to question how a certain fact was concluded, why a certain estimation algorithm is executing etc. It is also possible to trace the execution of the rules.

Planning

Expert control contains an element of planning. An on-line fault, a command to change production goals etc., each call for a sequence of steps to bring the process in line with requirements. Each step in this plan involves some action to adjust the process. The actions taken must not interfere with the preconditions of actions to follow in accordance with the control plan. With many actions available, and with many possible sequences, the development of a control plan may be viewed as a search through a large network for a path that reaches the currently established goals. Picture here an experienced human operator who, given time and enough information, knows how to bring the process in line with required operational conditions. This searching and planning in a complex environment is a fundamental activity in AI systems.

In classical algorithmic control there are well-defined and highly constrained notions of state and state transition. These are associated with physical parameters and operations. In expert control the goal is to deal with more ambiguous, less constrained, more qualitative notions, in addition to the available quantitative knowledge. Control strategy development in classical control can use tools like dynamic programming with an exhaustive search for a plan. Expert control deals with those process control probems where the sheer number of alternatives makes the search impractical. For instance the solution for finding a plan for 8 steps with 4 options at each step, should reasonably not deal with all $4^8 \approx 64\,000$ possibilities. Heuristics might be found, for example, to select appropriate steps at plan stages 2, 4 and 6, whereupon the search would be reduced to $4 \cdot 4^2 = 64$ options.

Important work on expert systems for planning has been carried out in the context of robot motion (Nilsson, 1980; Sacerdoti, 1977) and genetics experiment design (Stefik, 1981a,b). But the planning problem for expert control of a complex plant has elements which are not addressed in these works such as uncertainty in the state, the models, and the outcome of an applied action. (Nilsson (1980) does treat the last consideration in his program, STRIPS.) In some cases the stochastic dynamic programming problem formulation provides a framework for the requisite planning. But this algorithmic approach which amounts to global search, may not be feasible. Expert systems have been applied to similar complex tasks before with noteworthy success (Buchanan *et al.*, 1969). In the next section, although no specific planning formulation is put forth, some of the elements of the planning problem are described for the case of a single objective for a single loop.

5. AN ADAPTIVE REGULATOR

To illustrate the concepts a simple example in the steady state control of an industrial process will be investigated. It is shown that expert control can give added performance to such a mundane operation. The different functions that the system is required to perform are first given. These functions can be

implemented by a combination of different algorithms and rules. Functions which are related can be grouped into knowledge sources.

Operations

For simplicity only a single control loop is considered. Let the goal be to hold a process variable close to a given set point with reasonable control actions. If the dynamics of the process and the disturbances were known, a minimum variance regulator could be designed. Examples of such control strategies and the underlying theory are given in Åström (1970). If the process is described by the sampled model

$$Ay(t) = Bu(t) + Ce(t), \tag{2}$$

where u is the control variable, y the output, e white noise, and A, B and C are polynomials in the forward shift operator, then the optimal control law is

$$Ru(t) = -Sy(t), \tag{3}$$

where the polynomials R and S are given by

$$z^{d-1}CB = AR + BS \tag{4}$$

and d is the delay in the system, i.e. $d = \deg A - \deg B$. The operations necessary to obtain and maintain the minimum variance control law given by (3) in a safe way will now be discussed.

Minimum variance control is the main function of the system. Models (2) for the process and the disturbances and a sampling period are required to apply the function. With the simple minimum variance control law it is necessary to ensure that the preconditions for minimum variance control are satisfied. The most important condition is that the process dynamics are minimum phase, i.e. that the process zeros are inside the unit disc. A particular feature of the minimum variance control law is that the process zeros are cancelled. This may lead to ringing if the zeros are not sufficiently well damped. The trade-off between input and output variance is governed by the prediction horizon $d \cdot h$, where d is the delay in (2) and h is the sampling period. Notice that if the process is stable then the model will always have zeros with arbitrarily good damping if the sampling period is long enough (Åström *et al.*, 1983). To be able to detect ringing and to take appropriate actions it is useful to include a *ringing detector*. The ringing can be eliminated by increasing the value of the parameter d; see Åström and Wittenmark (1985).

There is a convenient way to find out if a process is under minimum variance control because the output is then moving average

$$y(t) = \lambda[e(t) + \ldots + f_{d-1}e(t - dh + h)], \tag{5}$$

where

$$F = \frac{R}{B} \tag{6}$$

and h is the sampling period. A *minimum variance supervisor* can thus be based on the calculation of the correlation function of the output.

If a sufficiently accurate process model for preselection of R and S from (3) and (4) is not available, a self-tuning regulator (STR) may be used (Åström and Wittenmark, 1973). Such a regulator may, under certain conditions, converge to the minimum variance regulator which could be designed if the process model were known. The simple self-tuner is in essence a parameter estimator. In the STR the model is reparameterized in terms of the regulator parameters. A *parameter estimator* will thus be included. The precondition for parameter estimation is experimental data obtained when the process is properly excited. To ensure a proper operation of the estimator an *excitation supervisor* will therefore be introduced. This would in essence determine the energy of the input signal in the useful frequency range.

The normal estimation algorithms will perform well if the parameters are constant or slowly varying. To cover situations where the parameters may change suddenly it is useful to include a *jump detector* which can be used to reinitialize the estimator; see Hägglund (1984).

If there is not enough excitation there are two options: either to stop updating the model parameters or to introduce perturbation signals (Åström, 1984). In those cases when perturbation signals are allowed the system will be provided with a *perturbation signal generator*. The generation of the perturbation signal requires some information about the frequency range of interest and about the allowable perturbation levels. This may be derived from the knowledge of the delay d and the sampling period h.

The STR also requires prior knowledge. In particular the following data is needed for the basic STR (Wittenmark and Åström, 1984): h, sampling period; d, delay in number of sampling periods; n_R, degree of the polynomial R; n_S, degree of the polynomial S; λ, forgetting factor; θ_0, initial estimate; p_0, initial covariance and high and low control limits.

The parameters d and h are crucial since the closed loop system may become unstable if they are underestimated. To detect this the system should therefore be provided with a *stability supervisor*. It is possible to determine if the integers n_R and n_S are

large enough simply by calculating the covariance functions $r_{yy}(\tau)$ and $r_{yu}(\tau)$ (Åström and Wittenmark, 1973). A *degree-supervisor* can thus be constructed and included in the system.

The importance of knowledge of the product $d \cdot h$ has been emphasized. A robust estimate of this quantity may be derived from the Ziegler–Nichols auto-tuner discussed in Åström (1982). This procedure gives the critical gain k_c and the critical period t_c. A *kc-tc estimator* with some supervision, as is discussed in Åström and Hägglund (1983), is thus also provided. A safe estimate of $d \cdot h$ with monotone step responses is actually $t_c/2$. When using the kc-tc estimator data will also be obtained, which is useful in estimating other parameters.

If the parameters k_c and t_c are known it is possible to determine a *back-up control* based on the Ziegler–Nichols methods for PID control. Such a regulator would provide control of the process even if the performance was far from optimal.

Other functions may also be provided. Assume that the process dynamics change with the operating condition of the process. Gain-scheduling may then be considered. For this purpose it is useful to have functions like *smooth-and-store regulator parameters*, *get regulator parameters* and a *test scheduling hypothesis*. The last function scans the parameter values stored in a table for a given process state and determines if the values are reasonably close. These tables themselves are produced during operations where conditions, parameters and outcomes are stored. The function that compiles these tables is called a *learning supervisor*. It is highly desirable to supervise the global behaviour of the system in normal operation. This is done by the *main monitor*. This computes means, variances and maximum deviations. From the information generated by this monitor questions arise such as: Is the system operating normally? Are the deviations the same as in the same period last year?

The functions discussed may be grouped into knowledge sources as follows:

Main control: minimum variance control; minimum variance supervisor; ringing detector; degree supervisor.

Back-up control: PID control; kc-tc estimator.

Estimation: parameter estimation; estimation supervisor; excitation supervisor; perturbation signal generator; jump detector.

Self-tuning: self tuning regulation.

Learning: get regulator parameters; smooth-and-store regulator parameters; test scheduling hypothesis.

Main monitor: stability supervisor; compute means and variances.

The data base

The system data base stores current process data to support technical audit and learning. The entries in the data base must accommodate these demands.

Data base planes. The data base hosts static and dynamic data consisting of facts, evidence, hypotheses and goals. These data types need not be maintained separately in an implementation. Rather it is usually more appropriate to divide the data base along planes that are the focus of the knowledge sources. One of the most important planes is the event list.

Event lists. One convenient and often used approach for dealing with a time-varying environment in an expert system is to make the processing "event-driven". Suitable actions are proposed under the direction of the supervisory control according to the nature of events entered on an event list. These events may be entered into the processing from an external source, or may result from internal processing by the knowledge sources on earlier events. Event types include: threshold crossings for process levels and rates, human operator command entries, entry of new hypothesis on process conditions, modification of an earlier hypothesis, request for control mode change, announcement of control mode change and requests by the human operator for information.

A few examples of event lists will now be given. Some lists are organized to provide data for the different knowledge sources. Data used for the main monitor are shown in Table 1. The entries are time, the mean values of the control and output signals, u and y, their standard deviations σ_u and σ_y, the results of a stability supervisor and the type of the regulator. The maximum and minimum deviations may also be entered. The major control modes are manual back-up, minimum variance and self-tuning. An entry is made in Table 1 when the mode changes or when the set point is changed.

It may be useful to add a few entries in the table such as max and min values or percentile values. The mean values entered in the table for event n are mean values between the events n and $n + 1$. From the data shown in Table 1 it is possible to make deductions like: What are the relations between the mean values of u and y? Do these relations change with time? Are there any relations between the standard deviations and the mean value of the

TABLE 1. MAIN MONITORING TABLE

#	Time	u	σ_u	y	σ_y	Stable	Regulator type

control signal? What are the patterns of the mode switches? Does the system go to tuning mode after large set point changes? What control modes are used most of the time? Are there drastic variations in performance with time and modes? The answers to these questions will allow inferences about the characteristics of the process.

The essential data used in the back-up mode is shown in Table 2. An entry in this table is made whenever the system goes into back-up mode or when a kc-tc tuning is made when the system is in back-up mode. The entries are the critical gain k_c, the critical period t_c and the PID parameters. The reason for entering both the critical point parameters and the PID parameters is to have options for the system to modify the PID design rules.

The main data generated during periods with fixed gain minimum variance control are given in Table 3. An entry into this table is made whenever the fixed gain minimum variance control is in operation. The entries are the parameters n_R and n_S, the prediction horizon d, the sampling period h and the regulator parameters.

From Tables 1 and 3 questions can be answered such as: What structures of minimum variance regulators are used? Do the structures have any relations to the operating conditions? Are there any patterns in the performances obtained? An entry is made into the parameter estimation table whenever a parameter estimation is performed. The entries are time, operating conditions, the perturbations, the sampling period, the prediction horizon, the complexity parameters and the regulator parameters.

Table 4 is related to the parameter estimation. Data are entered into this table when parameter estimation is performed either periodically or on demand when operating conditions are changed. The table can be used to explore parameter changes

TABLE 2. BACKUP CONTROL TABLE

#	Time	k_c	t_c	P	I	D

TABLE 3. MINIMUM VARIANCE CONTROL TABLE

#	Time	n_R	n_S	d	h	Parameters

TABLE 4. PARAMETER ESTIMATION TABLE

#	Time	OC	Perturb.	h	d	n_R	n_S	Parameters

and if the changes are related to operating conditions.

In practice the number of event types is large—a system of modest complexity may have in excess of 50. The flow of an event-driven system is produced by the event type together with the rules in the supervisory control element for the ordering of their processing. There are also other tables in the system data base such as a learning control table.

The hypothesis list. The hypothesis list is another important part of the data base. It is an organized collection of the derived working understandings of the process conditions. The organization often takes the form of levels of abstraction. The lower levels are typically concerned with immediate inferences from sensor data. For example, the hypothesis that the control errors are small is easily deduced from a list of the current means and variances, see Table 2. More complicated hypotheses may include making an estimate of the current level of stability of the process. Both numerical evaluations and rule of thumb heuristics can be used for this purpose.

As the level of abstraction increases it may be necessary to interact with the process engineer to blend his capability with that of the machine. In order to provide a mixed initiative capability, it is important for the engineer to have access to the rationale upon which system inferences are based. In an expert system this is provided by the attaching of rule numbers to internal events. In this way the data base supports the capability for a technical audit of the processing.

Another important function of the data base is to store processing history in a manner suitable for automatic learning. The idea is similar to learning from experience in chess or checkers (Samel, 1967; Michalski *et al.*, 1983). In the process control problem there are analogous questions. For example, it is currently difficult to determine analytical approaches for selecting thresholds for control mode switching.

6. A PROTOTYPE IMPLEMENTATION

A system employing the PID functions described in Section 5 has been implemented on a VAX 11/780 running under VMS. The expert system is written in LISP and the numerical algorithms in PASCAL. The LISP used is the UNIX dialect Franz Lisp (Foderaro *et al.*, 1981). The software package EUNICE (Kashtan, 1982), is used to create a UNIX environment under VMS. The expert system parts are based on the framework OPS4 (Forgy, 1979). A brief discussion of the system is given below. More details are given in Årzén (1986).

Concurrency is obtained by exploiting the real-time operating system facilities of VMS. The system has three parallel processes; the expert system, the

algorithms and the man–machine communication. These processes are implemented as VMS subprocesses. They communicate through VMS mailboxes. A mailbox is a queue where messages can be read or written. A message is associated with a line of text. This mechanism simplifies the communication between LISP processes since in LISP there is no difference between data and program. A message can thus be an arbitrary LISP expression. A process graph of the system is shown in Fig. 3.

The numerical calculations are implemented as a libarary of algorithms. The algorithms can be divided into three groups; control, identification and supervision algorithms. They are coded in a uniform way with separate parts for initialization, execution, parameter changes and abortion. This structure reflects the type of messages the expert system can send. Messages received by the expert system are typically results from identification algorithms or alarms from supervision algorithms. One design goal has been to make it easy to add new algorithms and to update old ones.

The expert system is written in LISP and OPS4, which is a production system framework that uses forward chaining. Forward chaining means that the program works from an initial state towards some goal by successively applying the rules. The framework OPS4 consists of three parts: the working memory, which is the data base; the production memory, which contains the rules and the inference engine, which controls the rule execution.

The syntax of a rule looks like

```
Rulename
    ("condition element" ...
→    "action" ...)
```

The "condition elements" are patterns which are matched against the contents of the working memory. The patterns may contain variables as well as constants. If all the condition elements in a rule are matched then the rule is satisfied and the actions can be performed. Actions can add new elements to the working memory, delete old elements or execute

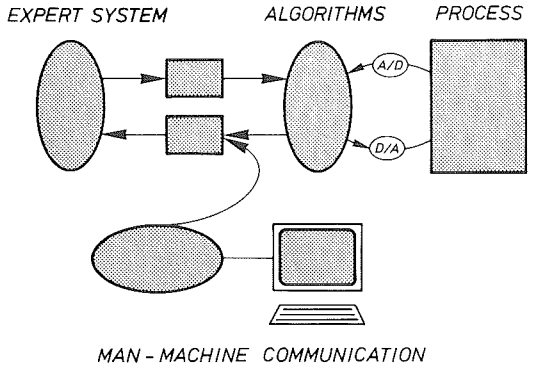

FIG. 3. Process structure. The ellipses represent processes and the rectangles represent mailboxes.

a user written LISP function etc.

The inference engine repeatedly finds all the rules that are matched. It selects one of them and performs its actions. When no more rules are found the system waits for incoming messages. A message is either a new element to be entered into the working memory or a LISP function that should be evaluated. New memory elements may cause new rules to match and thus start the rule execution again.

The man–machine communication process is written in LISP, which fits the purpose. An interactive environment is achieved automatically and the communication with the expert system is simplified. The available commands can be divided into two groups. The first group contains the usual commands available to a controller, e.g. commands for parameter changes or for switching the operating mode. The second group contains commands that have to do with the expert system. Examples are commands for inspecting the working memory, commands that start tracing the execution of rules etc. It is also possible to add, delete and edit rules on line.

This prototype implementation has been used for experiments with a PID controller with auto-tuning and gain scheduling. The tuning is based on the Ziegler–Nichols auto-tuner (Åström, 1982). The algorithms used are a PID algorithm, a relay algorithm, and a kc-tc estimator. The expert system has two tables, one for PID parameters measured at different set points and one for stationary values. An alarm level supervisor is used to ensure that the measured output does not exceed critical levels.

The rulebase for this first experiment consists of approximately 70 rules. This may seem large but can be explained in two ways. One reason is that this relatively small problem contains more logic than is apparent at first sight. All possible cases must be considered to make the rulebase consistent. The other, most important, reason is the poor expression power permitted by the rule syntax in OPS4. Some examples of rules are

```
Rule 5
  ((State is upstart)
   (Goal is PID-control)
   (PID parameters available)
→
   (⟨DELETE⟩ (State is upstart))
   (⟨ADD⟩ (State is PID-control) (Start PID)))

Rule 12
  ((State is = X)
   (Alarm has occurred)
→
   (⟨DELETE⟩ (State is = X))
   (⟨ADD⟩ (State is alarm) (Old state is = X)))
```

The symbol $= X$ denotes a variable that can match any constant in the working memory.

The approach taken in this work has been to take an existing expert system framework, OPS4, with its limitations and to incorporate it in a control system. The reason behind this was to be able to test the basic ideas behind expert control quickly. The experiments performed with this system have been promising. The expert control approach gave a much cleaner auto-tuner implementation than other implementations in PASCAL and PLM; see Åström (1983). The main reason for this was the clear separation of the algorithms and the logic. Another benefit was the ease in changing the logic in the expert system implementation. An example of this is that the addition of gain scheduling to the system required less than 10 new rules to be added. The approach is thus very valuable for prototyping.

The expert system framework OPS4 is not ideal for expert control. It has some advantages such as the possibility to extend the system with user written functions and its relatively high processing speed. The system has however no possibility for backward chaining or reasoning with uncertainty. There are no facilities for structuring the data base or the rules. The rule syntax is also somewhat awkward. The control problems also have a strong planning element which is not supported by OPS4. The most important drawback however with OPS4, which it shares with most of the other existing expert system frameworks, is that it is not designed for real-time operation. Real-time expert systems is still an area where much research has to be done before a true expert control system can be built.

7. CONCLUSIONS

It is straightforward to extract a more general pattern from the examples in Section 5. To solve a control problem using this approach a number of design approaches are first determined that may be appropriate for the problem. The design methods are analysed carefully to determine the conditions under which they perform satisfactorily and those when they do not. Next criteria are sought delineating these conditions. Finally an expert system is used to decide when and how to apply the different methods. This approach, which can be applied to a wide variety of problems, seems to offer interesting possibilities for combining analytical and heuristic approaches.

For simplicity, the use of AI techniques has been applied here to single loop control. This allows an uncluttered presentation of some of the elemental concepts that arise when AI and control technologies are merged. A heuristic component has been added to familiar estimation and control algorithms. A key point is that the incorporation of heuristics through AI structures results in systems that are far more flexible and transparent than systems based on selectors and safety jackets currently in use in standard hard-wired logic.

Experience from experiments with systems of this type has shown that the approach is useful in several respects. It is very effective as a test bench for defining the logic required for safe operation of potential control schemes even if this logic is later implemented differently. The experiments point toward the conclusion that powerful control laws can be obtained by combining conventional control algorithms with an expert system. The approach taken in this paper also emphasizes the need for new theoretical results. Design of a stability margin supervisor is a typical example.

Experience from building expert systems for real applications has shown that their power is most apparent when the problem considered is sufficiently complex. Process control problems are admittedly complex. Plant operators run systems with multiple loops, unpredictable material variations, etc. Over time, and with experience, operators generate rules of thumb that help them deal with this complexity. This paper has pointed out that an expert system can provide a framework for blending numerical algorithms with this detailed expertise of the plant operator.

REFERENCES

Årzén, K. E. (1986). Experiments with expert systems for process control. *Proc. 1st Int. Conf. Appl. Artif. Intell. Engng Practice*, Southampton, UK, 1986 (to appear).

Åström, K. J. (1970. *Introduction to Stochastic Control Theory*. Academic Press, New York.

Åström, K. J. (1982). A Ziegler–Nichols auto-tuner. Report TFRT-3167. Department of Automatic Control, Lund Institute of Technology.

Åström, K. J. (1983). Theory and applications of adaptive control. *Automatica*, **19**, 471–486.

Åström, K. J. (1983). Implementation of an auto-tuner using expert system ideas. Report CODEN: LUTFD2/TFRT-7256. Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Åström, K. J., P. Hagander and J. Sternby (1983). On the zeros of sampled systems. *Automatica*, **20**, 31–38 (1984).

Åström, K. J. and A. Helmersson (1983). Dual control of a low order system. In Landau, I. D. (Ed.) *Outils et Modèles Mathématiques pour l'Automatique l'Analyse de Systèmes et le Traitement du Signal*, Vol. 3, pp. 741–758. Editions du CNRS, Paris.

Åström, K. J. and T. Hägglund (1983). Automatic tuning of simple regulators. Preprints IFAC workshop on adaptive systems in control and signal processing, San Francisco, California.

Åström, K. J. and T. Hägglund (1984). Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, **20**, 645–651.

Åström, K. J. and B. Wittenmark (1973). On self-tuning regulators. *Automatica*, **9**, 185–199.

Åström, K. J. and B. Wittenmark (1985). The self-tuning regulators revisited. *Proc. 7th IFAC/IFORS Symp. Ident. Syst. Param. Est.*

Åström, K. J. (1984). Interactions between persistent excitation and unmodeled dynamics in adaptive control. *Proc. 23rd IEEE Conf. Decis. Control.*

Barr, A. and E. A. Feigenbaum (Eds.) (1982). *The Handbook of Artificial Intelligence*, Vol. 2. William Kaufmann, Los Altos, California.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, New Jersey.

Buchanan, B., G. Sutherland and E. A. Feigenbaum (1969). Heuristic DENDRAL. A program for generating exploratory hypothesis in organic chemistry. In Mettzer and Michie (Eds.), *Machine Intelligence*, Vol. 1, pp. 209–254. American Elsevier, New York.

Clarke, D. W. (1981) Implementation of adaptive controllers. In C. J. Harris and S. A. Billings, (Eds.), *Self-tuning and Adaptive Control*. Peter Perigrinus, UK.

Davis, R. (1982). Expert systems. Where are we? And where do we go from here? *AI Mag.*, Spring, 3–22.

Davis, R. and D. Lenat (1982). *Knowledge-Based Systems in Artificial Intelligence*. McGraw–Hill, New York.

Foderaro, J. K., K. L. Sklower and K. Layer (1983). *The Franz Lisp Manual*. University of California, Berkeley.

Forgy, C. I. (1979). OPS4 User's Manual. Technical Report CMU-CS-79-132, Department of Computer Science, Carnegie–Mellon University.

Hägglund, T. (1984). Adaptive control of systems subject to large parameter changes. *Proc. 9th IFAC Wld. Congr.*, Budapest, Hungary.

Hayes–Roth, F., D. Waterman and D. Lenat (1983). *Building Expert Systems*. Addison–Wesley, Reading, Massachusetts.

IEEE (1983). Knowledge representation. *Computer*, **16:10**.

Isermann, R. (1982). Parameter adaptive control algorithms—A tutorial. *Automatica*, **18**, 513–528.

Kashtan, D. L. (1982). EUNICE: A system for porting UNIX programs to VAX/VMS. Artificial Intelligence Center, SRI International, Menlo Park, California.

Michalski, R., J. Carbonell, and T. Mitchell (1983). *Machine Learning, An Artificial Intelligence Approach*. Tioga, Palo Alto, California.

Moore, R. L., L. B. Hawkinson, D. G. Knickerbocker and L. M. Churchman (1984). Expert systems applications in industry. *Proc. ISA Int. Conf.*, Houston, Texas.

Nii, H. P., E. A. Feigenbaum, J. J. Anton and A. J. Rockmore (1982). Signal-to-symbol transformation. HASP/SIAP case study. *AI Mag.*, Spring, 23–35.

Nilsson, N. (1980). *Principles of Artificial Intelligence*. Toga, Palo Alto, CA.

Sacerdoti, E. (1977). *The Structure of Plans and Behavior*. Elsevier, New York.

Samuel, A. L. (1967). Some studies in machine learning using the game of checkers II—recent progress. *IBM J. Res. Dev.*, **11**, 601–617.

Stefik, M. (1981a). Planning with constraints (MOLGEN: Part 1). *Artif. Intell.*, **16**, 111–139.

Stefik, M. (1981b). Planning and meta-planning (MOLGEN: Part 2), *Artif. Intell.*, **16**, 141–169.

Trankle, T. L. and L. Z. Markosian (1985). An expert system for control system design. *Proc. IEE Int. Conf. Control*, Cambridge, UK.

Waterman, D. A., and F. Hayes–Roth (Eds.) (1983). *Pattern-Directed Inference Systems*. Academic Press, New York.

Winston, P. H. (1977). *Artif. Intell.* Addison–Wesley, Reading, Massachusetts.

Wittenmark, B. and K. J. Åström (1984). Practical issues in the implementation of adaptive control. *Automatica*, **20**, 595–605.