NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

KNOWLEDGE BASED CONTROL SYSTEMS

By

RAVI LINGARKAR, B.Eng., M.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Doctor of Philosphy

McMaster University

ISBN 0-315-87999-8

Canada

KNOWLEDGE BASED CONTROL SYSTEMS

Abstract

There are many different approaches to knowledge based control, in this thesis, three different approaches have been taken for study. These are:

- Expert Control
- Fuzzy logic based Control
- Qualitative reasoning for control and diagnosis

In the first approach, the knowledge based systems acts as a supervisor to a traditional controller, with the ability to advise an operator about degradation in the performance or if needed tune/change the control algorithm used. This approach is most appropriate when the mathematical model of the system is known. Several new theoretical insights to the problem of expert control is given in this thesis. An implementation architecture, along with the lessons learned from using such architecture are also dicussed.

Alternatively, the knowledge based system may be placed in the control loop; i.e. the control laws may be embedded in rules that are a part of the knowledge based system. Fuzzy logic controllers fall within this category. Fuzzy logic controllers circumvent the problem of designing a controller based on a detail dynamic model by employing the approach of a human operator to an ill-defined system. A step by step procedure for developing such controllers is given in this thesis. This

DOCTOR OF PHILOSPHY (1992) McMASTER UNIVERSITY
(Electrical and Computer Engineering) Hamilton, Ontario

TITLE: Knowledge Based Control Systems

AUTHOR: Ravi Lingarkar, B.Eng., M.Sc.

SUPERVISOR: Professor N. K. Sinha

NUMBER OF PAGES: xvi, 204

Acknowledgements

Thanks to all the friends and scientists at McMaster who made this work possible. While tradition requires me to take responsibility for any remaining flaws, honesty compels me to share credit with them for whatever insight and clarity this thesis manifests.

I am deeply indebted to my supervisor, Naresh Sinha, for his guidance, support and encouragement. Many of the key ideas in this thesis arose during group meetings held by him every week. This work would not have been possible without his help, and thanks for giving so much.

Thanks to Dr. M. A. Elbestawi for getting involved in my work, for his many suggestions, and for allowing me to use his facilities. Thanks for believing in me.

Thanks to Dr. D. W. Capson for also getting involved in my work, and providing me with many valuable suggestions. Thanks to Dr. D. L. Parnas and Dr. J. Zucker for carefully reading through my thesis and suggesting many improvements.

Thanks to Dr. L. Liu and Dr. G. Bone for their help in implementing some of the work discussed in this thesis, and for many discussions on robotics. Thanks to V.

procedure is applied to solve two problems of increasing complexity in the areas of robotic deburring and mobile robotics. Implementation and test results are also given.

The first two knowledge based control techniques have some disadvantages. This is mainly due to the manner in which knowledge is represented, each fact or rule stands on its own as an underivable 'axiom' making it impossible to question the validity or basis of it. Qualitative control system, instead is based on deep knowledge extracted from the understanding of the physical interaction between the different components of the plant. This knowledge is used to construct a qualitative model that is used in control operations. In this thesis, an architecture for qualitative control is given and tested on a simple system via simulations. Furthermore, a strategy for expanding qualitative control to provide process diagnostics is also proposed.

P. Burhanpurkar for providing me with much information in the area of mobile robotics, and for allowing me to use his facilities. Thanks also to Dr. D. Heping, Dr. Y. Han, P. Tam, and R. Ho for many discussions on robotics and control systems.

Thanks to my colleagues at SPAR Aerospace Limited, for pointing to me, errors in my thesis. Thanks to B. Iyer for discussing with me many advanced software engineering concepts. H. Fanous for making me aware of his work in the area of knowledge based systems. S. McClure for explaining to me various issues related to real time systems. H. Sakata for talking about my work. M. Zakaria for teaching me many short cuts and issues related to robotic tracking systems. B. Mack for giving me practical insights on the hardware and software of advanced robotic systems. T. Ng for allowing me to attend numerous presentations related to my thesis on company time.

Thanks to McMaster University and Ontario government for their generous scholarship which facilitated my research. Thanks to Cheryl Gies for waging many administrative battles on my behalf.

Thanks to my parents for their help, patience and encouragement. Last but not the least, thanks to my wife Leena for her unconditional support.

Contents

List of figures

List of Tables

Chapter 1

Introduction

1.0 System Control

A system may be defined as a set of components working together to achieve a common objective. It may be possible, theoretically to define a process precisely by a function relating inputs, outputs and time. Each subcomponent of the system can itself be given a functional description. A process may be self-regulating, i.e. the variables and conditions in the system maintain the values necessary to achieve the objective under normal conditions. In most cases some type of control is needed. The purpose of the control subsystem is to order events and regulate the value of the variables in the system to optimize the achievement of the system goals. Although in the past, system control was primarily involved in the regulation of variables, today system control also incorporates higher level supervision and planning. Computers are generally used to implement these higher level functions. The software of such systems is normally embedded in some large system like ship, aircraft, missile, spacecraft, manufacturing or processing plant, or transportation system. Control may be distributed in these

systems with computers providing just a part of it.

Many of the difficulties that arise in building control systems stem from the fact that system control involves satisfying temporal relations. The objective is to maintain some property at some specified value over time or to bring about some sequence of state changes over time. The sequence of changes required may generally depend on satisfying prerequisite conditions in the process or environment and on time. In general, the operation of the process control system can be expressed as a function **F** relating the systems inputs **I**, outputs **O**, and time **T** (see figure 1.1). At any instant, an unique set of relationship exists between inputs and outputs in which each output value is related to the past and present values of the inputs and time. These relations involve fundamental mechanical, aerodynamic, chemical, thermal, or other laws embodied in the construction of the system. The system is constructed with components whose interaction implements **F**, which usually includes a control component.

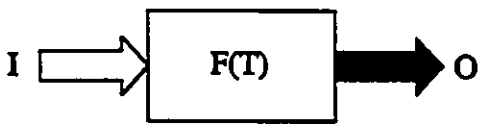


Figure 1.1. System representation.

In addition to the basic objective of implementing **F**, these type of systems also may have constraints on their operating conditions. Constraints define the range of conditions in which the system may operate while achieving its objective. Constraints may arise from several sources, which include:

- quality considerations,
- physical limitations of the components in the system (for example, actuator saturation),
- process characteristics (for example, limiting process variables to minimize production of by-products), and
- safety (avoiding hazardous states).

These constraints limit the set of acceptable designs. The designer also needs a set of criteria to choose the best acceptable design and to resolve trade-offs between conflicting requirements. In an ideal situation the designer should have a list of constraints along with a prioritized list of criteria that must be optimized.

System Components. Control engineers often model a closed-loop control system to implement the function **F** as four basic components: the system, the controller, the actuators, and the sensors. Figure 1.1 shows this control loop.

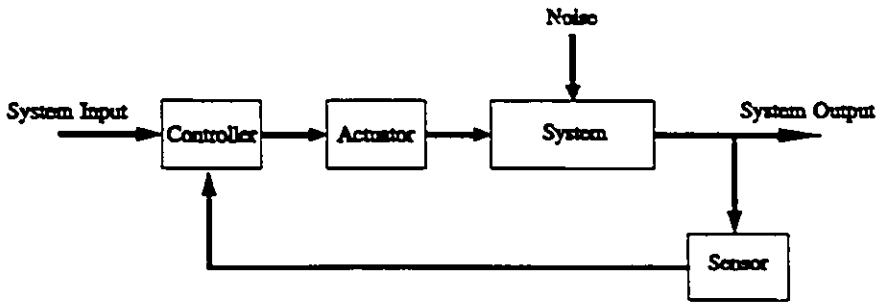


Figure 1.2. A typical control system.

2.0 Real-time knowledge based control systems

Real-time computer systems have become an integral part of our daily lifes. Real-time systems are those systems which are predictably fast enough for use by the processes being serviced. They must be able to guarantee a response after a fixed interval of time, depending upon the domain [LAF88]. These real-time computer systems are being used in a growing number of applications, starting from small, simple controllers found in common household appliances to large and complex systems used, for example, in industry.

In the past, many real-time systems have been developed using fast and efficient algorithmic approaches. Such systems where able to capture and analyze data in

a time-constrained environment. Many of these use numerical techniques and traditional software programmes. However, they are not in a position to apply knowledge and experience-based problem solving techniques, in a manner done by humans or operators. There are many problems that are not properly defined and the constraints involved are unstructured, thereby making them not conducive to mathematical modelling. As a result, algorithmic approaches fail. In the real-time domain, these algorithmic approaches have other deficiencies; when large amount of data is simultaneously available, relevant data may be overlooked and the response may be inconsistent and slow. This problem becomes even worse as the number of functions controlled, the rate at which the functions must be controlled, and the number of criteria that must be considered before a decision can be made are increased. To solve the above problems, in recent years there has been a flurry of activity in area of knowledge-based systems (KBS).

Due to the aforementioned problems with algorithmic approaches, KBS are being developed specifically in the real-time domain. A human expert normally operates in a narrow and specified domain. This knowledge is a combination of a theoretical understanding of the problem and a collection of heuristic problem-solving rules that has been, through experience, shown to be effective in solving a problem [LUG89]. A KBS is built by capturing the human experts knowledge in a machine readable form. KBS is then applied to real world problems with the

aim to provide decision support at a level comparable to a human expert. The KBS consists of two main parts, the inference engine and the domain dependent knowledge. It also provides some justification for its conclusions and reasoning.

There are many limitations with traditional KBS, for example, they cannot reason from first principles and learn from past experience. Despite these limitations KBS are proving to be of value in a number of applications. A few of these applications include control and diagnosis based on observable symptoms, monitoring a system by comparing the observed behaviour of a system to its expected behaviour, and debugging and repairing.

Application of KBS in the real-time domain has not been very successful. A real-time KBS is data driven and is different from a traditional KBS. In a traditional KBS the premise conditions of rules do not change with time during the entire transaction, and the reasoning is directed by the operator. For a real-time KBS, the reasoning is triggered by the input data collected from processors or sensors distributed in the underlying system it controls [YEH88]. Since the data vary with time, the system also needs to reason in a time variant manner. This also implies that the expert system would have to operate continuously in recurring cycles, its operation normally triggered by the changes in data as the time passes.

The requirement to deal with rapidly changing, time-dependent nature of real-time events requires the deveioping of fundamentally different techniques than those used in traditional KBS. This thesis is a study of application of KBS to the design of control systems, which are fundamentally of real-time nature.

2.1 Formulation of the real-time knowledge based system problem

A system is taken to be real-time if it is fast in the sense of problem relevant performance requirements, and uses temporally consistent data. A more formal definition of real-time [CAU88] [ORE85] is as follow:

If *response time* is the time the computer takes to recognise and respond to an external event, and if

T_α= time at which the system requests a response,

T_β = time at which the system response must be completed, and

t_r = response time then

a system is considered to be real-time, if for all $T_\beta > T_\alpha$, we have

$$t_r \leq T_\beta - T_\alpha$$

The above definition of real-time is very strict, however, in practice we can distinguish between *hard* and *soft* real-time systems [MOR88]. In a *hard* real time system, the system not responding within the strict time is fatal. In a *soft* real-time system, the system should respond at a minimum average rate. Although speed is important, there are several other issues that affect the performance of

real-time systems [DOD89], they include:

- The system should be consistent with itself and the environment as the data varies.
- The system is able to remain alert to incoming events.
- The system should be in a position to reset task priorities according to changes in the resource availability and /or demand and workload.

3.0 Issues related to real-time knowledge-based control systems

When knowledge-based systems are used in the area of control systems which are inherently real-time in nature, several important issues have to be taken into consideration. These issues include:

- Cyclicity of operation
- Asynchronous events and Non monotonic reasoning
- Time constraints for reasoning
- Temporal reasoning
- Focus of attention

3.1 Cyclicity of operation

Unlike expert systems, knowledge-based control systems have to be continuous

in operation. This means that its operation does not terminate once it has reached a conclusion using the current set of data. Instead, the system at the end of its predefined time interval should take in fresh data from the environment and restart its inference engine. This requirement for continuous operation implies that the system should not run lengthy secondary processes that may block the running of any of the different components of KBS. Such secondary processes may include garbage collection and archiving of sensor data.

3.2 Non-monotonic reasoning

Generally the activities within a KBS conform to a schedule, and this may not be so in the real-time domain. Events that need to be monitored may occur randomly, hence a mechanism should be in place that accepts unscheduled events. This entails that KBS should have the capability to save intermediate results or provides a synchronization mechanism between arriving asynchronous events and the rest of the system.

The non-monotonocity of an KB control system is due to the fact that the environment is constantly changing. Incoming sensor data as well as deduced facts do not remain static during the entire run of the program. Data may change to reflect the state of the environment, hence a mechanism should be provided to allow the assertion and retraction of inferred facts.

3.3 Reasoning under time constraints

It is extremely important that KB control systems meet their given *response times*. Different applications may have varying demands on the response times. There are several techniques that may be employed to assist in meeting the response times. These are discussed next:

3.3.1 World model simplification

As the complexity and data about the world model is decreased, the amount of processing time is decreased, which some times can be many orders of magnitude. This simplification of the world model should be balanced with the requirements that the model be accurate enough for the application. One form of simplification of the world model is through approximation of search, data and knowledge.

3.1.2 Integration of procedural components

KB control systems must be typically integrated with conventional real-time procedural software. The software will typically perform tasks such as data compression, signal processing, identification, application-specific input/output. The performance of the knowledge base can be enhanced by making sure that the procedural components are integrated efficiently.

3.1.3 Using Metaknowledge

Metaknowledge consists of information about the kind of knowledge stored in the KB. Using Metaknowledge for KB control systems is essential [RIC88] [MOO87], they help the inference engine to determine the applicability of the knowledge for the task at hand. One example of Metaknowledge is *meta rules*. Meta rules are used to partition the rule base. When one of these meta rules is fired, only a certain set of rules in the rule base will used during inferencing. This makes the inferencing process more efficient.

3.1.4 Hardware support

The most obvious technique for reducing the processing time is to use fast processors. Another technique is to use special purpose processors for symbolic and numerical computation. Parallel or distributed hardware can also help in reducing the processing time, but the overheads involved in coordinating the processors may quite soon exceed the advantage gained.

3.4 Temporal Reasoning

In some KB control systems it is necessary to reason not only about the present but also the past and the future. Inferencing in such systems should be performed on data and knowledge which are time dependent. Temporal representation give the system an ability to reference its own behaviour as facts and circumstances

change. Many types of temporal logics [ALE83a][ALE83b][ALE85] have been developed to handle temporal reasoning, and much of these techniques are still in their infancy.

3.5 Focus of attention

When a significant event occurs, it is important that a real-time knowledge based system be able to focus its resources on important goals. There are several methods for focusing attention, these methods include:

- Invoking a new set of rules that specializes in the current problem,
- changing the set of sensors the system is currently monitoring,
- changing the sampling rate or the filtering scheme of the data being analyzed, and
- activating another inference engine to analyze the event.

This ability to focus attention lets a system maintain a large body knowledge while applying only what is needed at any particular point in time.

4.0 Approaches to knowledge based control

There are many different approaches to knowledge based control, in this thesis, three different approaches have been taken for study. These are:

- Expert Control
- Fuzzy logic based Control
- Qualitative reasoning for control and diagnosis

In the first approach, the knowledge based systems acts as a supervisor to a traditional controller, with the ability to advise an operator about degradation in the performance or if needed tune/change the control algorithm used. This approach is most appropriate when the mathematical model of the system is known, and in many cases this approach provides the only possible architecture due to the slow response of knowledge based systems.

Alternatively, the knowledge based system may be placed in the control loop; i.e. the control laws may be embedded in rules that are a part of the knowledge based system. Fuzzy logic controllers fall within this category. Fuzzy logic controllers circumvent the problem of designing a controller based on a detail dynamic model by employing the approach of a human operator to an ill-defined system.

The aforementioned knowledge based control techniques have some disadvantages. This is mainly due to the manner in which knowledge is represented, each fact or rule stands on its own as an underivable 'axiom' making it impossible to question the validity or basis of it. Qualitative control system, instead is based on deep

knowledge extracted from the understanding of the physical interaction between the different components of the plant. This knowledge is used to construct a qualitative model that is used in control operations. When compared to rule based controllers, qualitative control systems are more robust to uncertainties in our knowledge about the system.

5.0 Contributions made in this thesis

1. An architecture for an expert control system is proposed. The proposed architecture has many advantages over previously developed architectures. One of them being that it uses a knowledge base that actively participates in the reasoning process.

2. For the first time a knowledge based controller is developed for the milling process, a very essential manufacturing process. The results from tests, and the lessons learned from this implementation are also given. Several enhancements to the proposed system that incorporate new technologies are also discussed.

3. When robots are used for deburring, significant amount of forces develop when the a robot interacts with the workpiece, which makes developing a good controller difficult. A fuzzy logic controller for the first time has

been developed for robotic deburring. The proposed fuzzy logic based controller is simulated, implemented and tested in real-time, on a robot with a independent end effector.

4. Obstacle avoidance in a wheel based mobile robot is achieved by varying the reference speed of the left and right wheel speed control systems. Developing a control system for this purpose is difficult, because it is not easy to model the robot. A new architecture that overcomes this difficulty of modelling the robot, based on fuzzy logic is proposed.

5. A method for developing qualitative reasoning control systems is proposed. The proposed scheme is tested on a sample system via simulations. Several problems with qualitative control systems have also been identified, and solutions have been proposed.

6. The idea of using qualitative reasoning is expanded to include diagnosis of process control systems. An architecture for the implementation of such a system is also given.

6.0 Organization of the thesis

There are three parts to this thesis. Each part discusses a particular approach to

developing knowledge based control systems, namely, expert control (chapters 2 &3), fuzzy logic based control (chapters 4, 5 and 6) and qualitative control (chapters 7 & 8).

Chapter 2 formulates the expert control problem, and in chapter 3, an architecture for an expert control system is proposed. The proposed architecture is illustrated via an example from the area of manufacturing. Test results, limitations, and enhancements to the approach are also given in chapter 3.

Chapter 4 discusses the theoretical aspects of fuzzy logic control systems. Chapters 5 and 6 discuss the development of fuzzy logic control systems for two different applications of increasing complexity, one for robotic deburring and the other for obstacle avoidance in a mobile robot. The architecture, implementation and test results are also given.

The application of qualitative reasoning for control and diagnosis is discussed in chapters 7 and 8. Chapter 7 discusses the theory behind qualitative control, and a control system is built for the system discussed in chapter 5. Chapter 8 discusses the use of qualitative reasoning for diagnosis of process control systems. An implementation architecture is also proposed in chapter 8.

The conclusions from undertaking the three different approaches to knowledge based control are given in chapter 9.

Chapter 2

Knowledge-based Supervision of Control Systems

1.0 Introduction

A common characteristic of large-scale, dynamic thermal and electrical systems is that components fail, loads are unpredictable, and reconfiguration is frequent. All these changes, affect both the appropriate model structure and the model parameters. The control of such systems require the knowledge of an accurate analytical model of the system. The model may also be used as the basis for future control decisions, resulting in a form of adaption.

Presently, there exist numerous techniques for parameters identification, given an initial *a priori* model [SIN83][GOO84] [LJU90]. All these techniques require a human to effectively identify and reformulate the boundary conditions and equations comprising the model form. However, a significant class of dynamic systems exists for which the model form, as well as the parameters change. These systems are generally controlled by using human operators for monitoring, troubleshooting and for recognizing model form changes. If the system is moderately complex, using human operators becomes difficult because of their

data processing limitations, unreliability and cost.

The approach for the identification and control problem taken in this thesis is to add to the computational methods, the techniques used by the artificial intelligence community. Hence, it is called knowledge-based control. Object-oriented methods [BOO91] are used to build a symbolic model of the physical system, representing the pieces of the physical structure in terms of assemblies, subassemblies, connections, and components as a network of frames [FIK85] with appropriate inheritance properties. Sets of rules operate over this knowledge base, making decisions to change the controller and/or control parameters. Data-driven programming techniques are used to attach procedures to locations in the knowledge-base. The architecture of such a system and its application to real-world systems is described in the next chapter.

2.0 Problem formulation

Any system can be represented by a set of differential equations, boundary conditions, and initial conditions given by:

$$\alpha_i [\ \theta_m,\ u(x,t),\ f(x,t)\ ]$$

$$\beta_j [\theta_m,\ u(x_b,t)\ ]$$

$$\gamma_k[\ u(x,t_o)\ ]$$

The dependent variables being $\mathbf{u}(x,t)$, x_b are the boundary locations, θ_m are the parameters, and the forcing functions are $\mathbf{f}(x,t)$. This general model formulation, representing a system by the tuple $[\alpha_i,\beta_j,\gamma_k]$ is applicable to wide a variety of both continuous and discrete (finite difference) systems. In most cases, for the purpose of control and identification, the form of the model $[\alpha_i,\beta_j,\gamma_k]$ is assumed constant. Deviations in the actual behavior of a dynamic plant from that expected are generally determined by computing the difference between the model-generated outputs and the actual plant. This can be written as:

$$u_a - u_m = e(\theta_m,t) \tag{2.1}$$

Where u_a is the output generated by the actual plant, and u_m being the output generated by the plant model. If the form of the model remains constant, system identification can be formulated as a parameter identification problem. In these cases, sets of $\mathbf{u}(x,t)$ are known for sets of input $\mathbf{f}$ and initial condition γ_k, but the parameter θ_m are not fully known. An error functional can be written in terms of the parameter shift error (2.1) by:

$$J = \int_t \int_x |e(\theta_m,t)|\,dx\,dt \tag{2.2}$$

Identification proceeds by minimizing this measure of error with respect to the

unknown parameters θ_m . This can be written as:

$$\frac{\delta J}{\delta \theta_m} = 0 \qquad (2.3)$$

Now assuming that θ_m varies continuously in the parameter space. Some iterative identification methods begin with an initial guess for θ_m, i.e. an *a priori* model.

3.0 Problems with numerical approaches

In many dynamic plants, the system model may change over time, moving within a model space. In this case, J must be minimized for both model and parameters. Since there may be an infinite set of discrete models, J cannot be minimized. If this is ignored, model form changes will lead to wrong parameters to be chosen, resulting in suboptimal or unstable control [SIN83]. Hence non-numerical approaches such as "knowledge-based" approaches have been proposed for online model identification. There are also other reasons for using knowledge-based approach. For example, equation 2.1, for individual errors can be written as:

$$u_{im} - u_{im} = e_i(t) \qquad (2.4)$$

These errors should be within a predetermined warning level. Violations of these warning levels often forms the basis of human operators' decisions to correct the

system. All this system dependent information can be best handled by a knowledge-based system. Parameter identification may need its own set of heuristics for proper operation. Take for example, the recursive least square (RLS) estimator [LJU83], the implementation of estimator requires the *a priori* specification of several critical parameters, including:

Δt	sampling period
k	process delay time
m, n, r	degree of the model polynomials
λ	forgetting factor values
$\theta(0)$	initial estimate of the parameter vector
P(0)	initial estimate of the covariance matrix
u_h, u_l	high and low control limits

The selection of these *a priori* estimator parameters and monitoring of the performance of the estimator depends on some heuristics which are partially system and situation dependent. Take for example, the case when a constant forgetting factor (λ) is used. When the system is close to steady state, old information is continually forgotten, while incoming data contains little new dynamic information. This leads to exponential growth in the covariance matrix, resulting in a situation where small errors produce large adjustments to the model parameter estimates. To overcome this problem, λ is set to 0.95 during startup

and 0.99 during periods of moderate process excitation. The above heuristic along with others noted by several other researchers [BOR76] [SAN85] [SEB86] [WIT84] [YDS85] are summarized below.

1. For systems without time delay, choose the sample rate such that there are at least three samples taken over the rise time of a sample step response.
2. For systems with significant time delay, choose the sample rate such that there are at least three samples taken over the span of the delay.
3. Choose the model order for the output, n to be in the range 2 to 4.
4. Choose the model order for the input, m to be n-1.
5. Set the forgetting factor, λ to 0.95 during start up.
6. Set λ to be 0.99 during periods of moderate excitation.
7. The covariance matrix P(0) is chosen to be αI where I is an identity matrix and α is chosen be large value.
8. The performance of the estimator can be determined by measuring the sum of the square of errors. A large error means a poor fit, and a steadily increasing error indicates estimator performance is degrading.

The idea behind expert control is the supervision of the control loop, and has been a subject of many papers in the last few years [ISE85]. The common idea of all supervision architectures comes from the incompleteness of the control laws which are more and more complicated. So, we understand the necessity to integrate

some known logical functionalities about the plant and the control law algorithm. These functionalities must ensure the validity of the mathematical theory which is behind the control law, and the physical description of the plant function along with its limitations.

In the next chapter, an architecture for supervision of adaptive systems proposed, used and thoroughly examined.

Chapter 3

Architecture For Supervision of Adaptive Systems

1.0 Introduction

In this chapter a new architecture for knowledge-based adaptive control systems is proposed. The usefulness of this architecture is shown via an example from the area of manufacturing. Its advantages and limitations over other approaches are also discussed.

This chapter is organized as follows. The process for which the adaptive controller is designed is described in Section 2. Representation of the knowledge and heuristics concerning the controller using frames is discussed in Section 3. The reasoning process of the adaptive controller is presented in Section 4. The hardware setup, simulations, and the results obtained from real-time experiments are presented in Section 5. Possible limitations and enhancements to the developed system are discussed in Section 6.

It is known that present adaptive control strategies work satisfactorily only within

a limited range [AST83]. To overcome this problem, researchers have suggested a third feedback level, termed the "supervision" level, to monitor range violations. To overcome the many problems associated with the design of an adaptive controller, we have taken a knowledge-based approach. In this approach, the knowledge-based system is a part of the primary feedback loop. It enhances the performance of the controller by allowing the designer to integrate the qualitative practical insights of experts together with other information available in more formal way.

Astrom et al. [AST86] have shown the advantages of using knowledge-based control by building a prototype adaptive regulator. While their scheme used a relational database to store the knowledge, we have used frames. Frames not only store knowledge, but also actively participate in the reasoning process [FIK85]. It is shown by choosing frames for knowledge representation, the "supervision" level is built into the knowledge representation scheme. Another benefit accrued by using an interactive knowledge-based system is that it provides a convenient environment to experiment with different control and system identification algorithms. It is also shown that combining declarative and procedural programming tools for designing knowledge-based controllers offers real advantages over traditional programming methods.

2.0 Computer-Controlled Machining

In this section the complexities associated with computer-controlled machining are described. In addition, the justifications for using a knowledge-based approach for designing a self-tuning controller for computer controlled machining is given.

Recent trends in machine tool design have transferred most of the cost of machining from the structure to the control system hardware, sensors, and software. In the majority of cases, the control methodology used in numerically controlled (NC) machining is open-loop positioning, which means that the tool positions and feedrates are precalculated and stored in the computer memory. Basically, in this approach feedback parameters are not used to compensate for any changes in the properties of the process. These changes can occur due to factors such as tool wear, random vibrations during cutting, wear of machine slides, etc. There are some well known problems associated with such control strategies . Some of them are:

When the depth of cut changes unpredictably during the machining operation, the NC programmer must use rather conservative feedrates to avoid tool breakage;

In the presence of significant tool/workpiece deflection (e.g., milling thin

webs) and other stochastic disturbances, significant dimensional errors occur in the finished workpiece again, conservative cutting conditions must be used to minimize these errors.

To significantly increase the utilization of modern NC machine tools (and hence decrease the machining costs), many researchers have suggested both adaptive and non-adaptive feedback control [DAN86] [MOH88] [TLU77] [ELB87]. In adaptive control systems, the machining parameters (e.g., feedrate, cutting speed) are used to compensate for unpredictable changes in the cutting process dynamics. Cutting force, torque, and power sensors are typically used to provide the feedback information.

End milling is an important machining operation, particularly in aerospace and automotive industries. Typical applications are, for example, the pocketing of airframe panels and the end milling of stamping dies in automotive manufacturing. Recently, there have been several attempts to apply parameter adaptive control algorithms to machine tools. Most of these applications are aimed at regulation of cutting forces during machining by adjusting the feedrate. Real-time control of the cutting force is very important, since it leads to increased tool life, better utilization of machines, and in some cases, better dimensional accuracy.

Figure 3.1 shows the block diagram of the machining process, where the effects of the tool deflections have also been included.

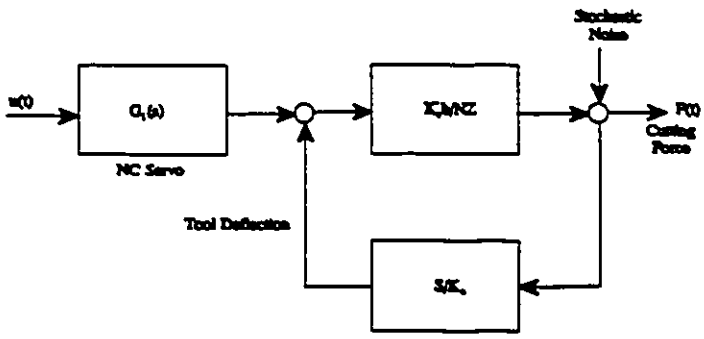


Figure 3.1 Dynamic model of the machining system.

It is evident that the cutting process model would vary, depending upon the depth of cut "b," spindle speed "N," number of teeth in the milling cutter "Z," cutting stiffness "K_c" and tool stiffness "K_s."

Self-Tuning Control in Peripheral Milling

The structure of the self-tuning control system for a CNC milling process is shown in Figure 3.2. The control objective is to hold the instantaneous resultant force acting on the cutter at a constant value F_r in spite of the variations in the depth of cut and other machining conditions.

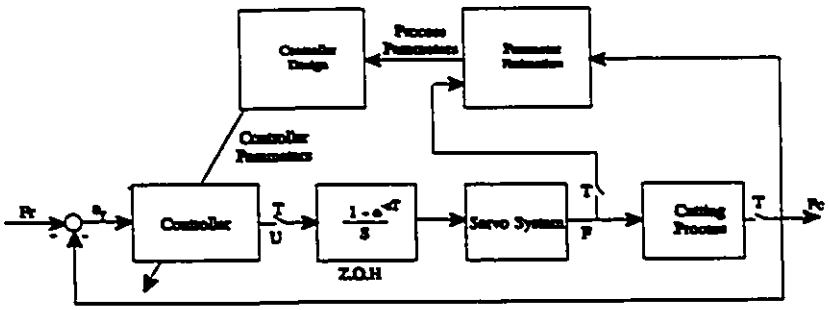


Figure 3.2. Structure of the self-tuning Controller.

The dynamics of the CNC servo loop (per axis) including the dc drive, is represented by a third-order system [WEL81] as:

$$\frac{f(s)}{U(s)} = \frac{B(s)}{A(s)} \tag{3.1}$$

where $B(s) = b_1 s + b_2$, and $A(s) = a_1 s^3 + a_2 s^2 + a_3 s + a_4$

and f(s) and U(s) are the Laplace transforms of the feedrate and commanded velocity, respectively.

The cutting force F_c in milling is related to the chip thickness by the equation

$$F_c = K'_c b h_{av}^m \tag{3.2}$$

where K'_c is a material dependent constant, b is the axial depth of cut, h^m_{av} is the average chip thickness, and m is an exponent (a typical value of m is 0.8). Including the effect of spindle/tool stiffness on the mechanics of the chip formation, the dynamics of the relationship between feedrate and cutting force is represented by first-order linear mode (i.e assuming that m=1) as follows [ELB87]:

$$F_c(k) = b_1 f(k-1) + a_1 F_c(k-1) + v(k) \tag{3.3}$$

where $b_1 = K_s(1-e^{-aT})/a$, $a_1 = e^{-aT}$, and $a = K_s NZ/(K'_c b)$. K_s is the spindle static stiffness (as would be measured at the bottom of the milling cutter), N is the spindle speed in revolutions per second, and Z is the number of teeth in the cutter, v(k) is the process noise.

The adaptive controller performs two main functions:

i) on-line identification of cutting process parameters, and

ii) control law calculation by using the estimated parameters.

The parameters of the cutting process, a_1 and b_1 as given in equation 3.3, are estimated using one of the many parameter estimation algorithms (recursive least square, recursive prediction error, etc.) [ELB87]. The parameters of the servo-loop are, on the other hand, constant and known *a priori*. Assuming a zero-order hold type of digital-to-analog converter, the discrete time representation of the servo-loop and the cutting process dynamics is of the form:

$$F_c(z^{-1}) = \frac{R(z^{-1})}{L(z^{-1})} U(z^{-1}) \tag{3.4}$$

with the polynomials

$$R(z^{-1}) = r_1 z^{-1} + r_2 z^{-2} + r_3 z^{-3} + r_4 z^{-4} \tag{3.5}$$

$$L(z^{-1}) = 1 + l_1 z^{-1} + l_2 z^{-2} + l_3 z^{-3} + l_4 z^{-4} \tag{3.6}$$

In real-time self-tuning action takes place with the coefficients of the polynomial $R(z^{-1})$ and $L(z^{-1})$ evaluated during each sampling interval. This is done by expressing the coefficients of polynomials in equation (3.5) and (3.6) as functions of parameters a_1 and b_1 in the cutting force model as given in equation 3.

When the feedrate is low, a nonlinear force equation, as suggested by Elbestawi and Sagherian [ELB87], is used. This nonlinear force equation is in the Hammerstein model form, and is expressed as a linear part

$$F_c(z^{-1}) = \frac{B^*(z^{-1})}{A(z^{-1})} X^*(z) \tag{3.7}$$

and the nonlinear part $X^*(z)$ is given by:

$$\begin{aligned} B^*(z^{-1}) &= b_1 z^{-1} \\ A(z^{-1}) &= 1 - a_1 z^{-1} \\ b^*_1 &= G_1 - 2G_2 + 3G_3\epsilon \\ b^*_2 &= G_2 - 3G_3\epsilon \\ b^*_3 &= G_3\epsilon \\ G_1 &= m \qquad G_2 = \frac{m(m-1)}{2!} \\ G_3 &= \frac{m(m-1)(m-2)}{3!} \end{aligned} \tag{3.8}$$

where a_1 and b_1 are as defined before and ε is the error term.

Self-tuning algorithm(s) converge, if the following conditions are satisfied:

- the process and noise model structures correctly correspond to the real

process,

- the estimated parameters lie within the stable range,
- a persistently exciting external signal acts on the closed loop system, and
- the desired control action can be attained by the actuators.

However, in practice, these conditions may be violated, and often performance problems were observed during operation. These problems include "bursts" in parameter estimation algorithm and oscillatory controlled system output. In many cases, the problems encountered in the machining experiments were typical of the adaptive scheme [ISE85]. Also, for machining processes, "timing is critical," which necessitates fast convergence of the parameter estimates. In addition, the control system must adequately deal with various cutting conditions (e.g., depth of cut, feedrates, work piece materials, cutter sizes, etc.). To avoid the problems faced in the aforesaid, the following actions should be taken.

Supervision of Parameter Estimation Algorithm

It is essential to monitor the performance of the RLS algorithm in real-time. These monitoring tasks are discussed next.

Filtering of the cutting force signal: The value of the forgetting factor in the parameter estimation algorithm was determined by off-line simulation of RLS

using data from nonadaptive cutting tests. Under the noisy condition of the metal cutting process, smoothing of F_c (cutting force) values is necessary to obtain better behaviour of the parameter estimates a_1 and b_1, which correspondingly brings about a stable control performance. This is achieved by using a moving average procedure. Averaging of F_c values is done over a fixed number of samples.

Resetting of the Parameter Estimator: When using the RLS method for parameter estimation, to start the recursive algorithm, the covariance matrix P(k) at time k = 0 is set to cI, where c >> 0. In milling applications a value of c =1000 was found to be adequate. During cutting, it is essential that the adaptive system must be able to react rather quickly to sudden changes in measured cutting force. The time required to react to a sudden increase in F_c can be substantially reduced by resetting the matrix P(k) to its initial value P(0) at each dangerous increase in depth of cut. This is recognized if the magnitude of $[F_c(k) - F_c(k-1)]$ exceeds a certain preset limit.

Monitoring the Values of Parameter Estimates a_1 and b_1: To ensure good control performance over the full range of operating conditions, supervision of the parameter estimates is very essential. For example, process parameter estimates may diverge when there is no significant change in system inputs and outputs. Hence switching the parameter estimator off is suitable when the estimates reach

the correct values. This would be recognized when the estimated force is equal to the measured value. Also, the parameter estimates $a_1 + b_2$ in the cutting process are constrained to be positive. When undesirable values of a_1 and b_1 are encountered in the transient stage, they are simply ignored.

Supervision of the controller performance: While conducting real-time tests, it was seen that the force response becomes rather oscillatory at low feedrates. The control performance at low feedrates is significantly improved when a nonlinear controller [ELB87] is used. Hence it is essential that the system uses a nonlinear controller when the feedrate falls below a certain level. This level was determined by cutting experiments. This level was determined by cutting experiments. The structure of the nonlinear controller is shown in figure 3.3.

From the preceding discussions it is seen that an intelligent means of control is required to handle the aforementioned complexities. In the sections that follow the design of the knowledge-based controller is presented.

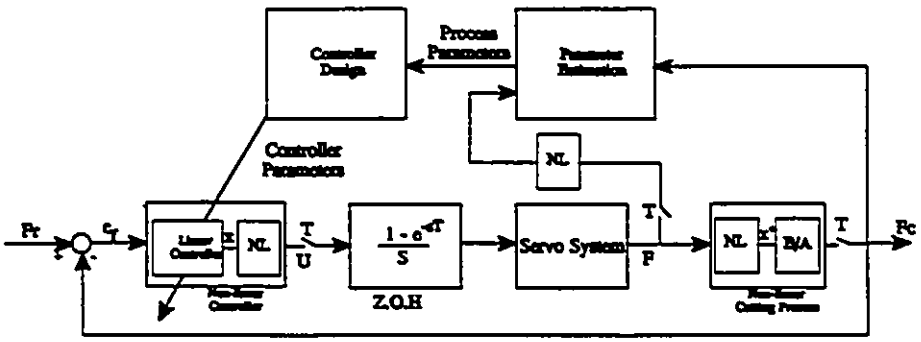


Figure 3.3. Structure of the non-linear controller.

3.0 Knowledge Representation

Knowledge is information about the world which allows an expert to make decisions. This knowledge needs to be represented and employed in a form that can be used for reasoning. The process of representing this knowledge formally is called knowledge representation. Just as data structures are used to store and deal with data, knowledge structures are used to store knowledge and reason with it. The type of knowledge representation that is appropriate in a given situation depends on what sort of knowledge is being represented and how it is to be applied. Many methods for knowledge representation have been proposed. Some examples are semantic networks, production rules, frames, conceptual dependency structures, and scripts. Of these, semantic networks, production rules and frames are more popular. Semantic networks are graph structures of different varieties, each emphasizing different kinds of relationships among the information represented in the network. The graph structures depict concepts (nodes) connected by links representing *semantic* relationships between the concepts. Production rules are expressed in the standard IF-THEN format and they are used to represent relationships between facts. Production rules consist of two parts: antecedant(s), which function as conditions, and a conclusion. Frames are a type of knowledge structure that function like tables or questionairres and use locations called slots to store attributes. Knowledge representation using frames is discussed in greater detail later in this section.

Semantic networks and production rules become unmanageable as the size of the knowledge base increases [PAT87]. In addition, it also becomes hard to interface the method of knowledge representation with the inference engine. In time-critical intelligent control applications, it is imperative that the knowledge representation scheme be efficient. Among the available knowledge structures, frames satisfy this criterion. The idea behind frames is that knowledge is stored in "chunks," or knowledge islands. This greatly reduces the computation required to access knowledge, i.e., once a particular chunk has been accessed, all knowledge relevant to that chunk is immediately available within that chunk.

Another reason for using the frame-based knowledge representation scheme is to exploit the structure in much of the knowledge being represented. This structure may arise due to the knowledge specific to the domain, and/or because of the structure imposed by us to effectively deal with large amounts of knowledge. Although it has been shown elsewhere [HAY83] that frame-based knowledge representation is equivalent to randomly ordered sequences of first order logic clauses, the readability and expressive power of frame-based representation is superior.

Representing knowledge through frames

Knowledge representation using frames was first suggested by Minsky [MIN75].

Frames provide a method of combining declarations and procedures within a single knowledge representation environment. The underlying principle behind frames is the packaging of both data and procedures into knowledge structures.

A frame has a unique name and its body is made of slots, which are used to describe the properties of the frame. Each slot may hold a value; if not, a default value is automatically assigned. These slots may have procedural attachments. which may be invoked when information is stored or retrieved from the frame. These procedure attachments are most often used to encode the heuristics associated with the controller.

Astrom [AST86] says that the code for incorporating the heuristics in the controller is much larger than the code for the core control algorithm, thereby making debugging, modification, and testing of the control logic difficult and time consuming. Encoding the heuristics as procedural attachments to the slots greatly reduces the aforementioned difficulties.

A typical frame that is used in the self-tuning controller for the CNC milling machine is shown in figure 3.4. The name of the frame is "CC-VARIABLES" and that of its parent frame name is "ROOT-FRAME." The frame contains two slots; the names of the slots are "FEEDRATE" and "C-FORCE." These slots

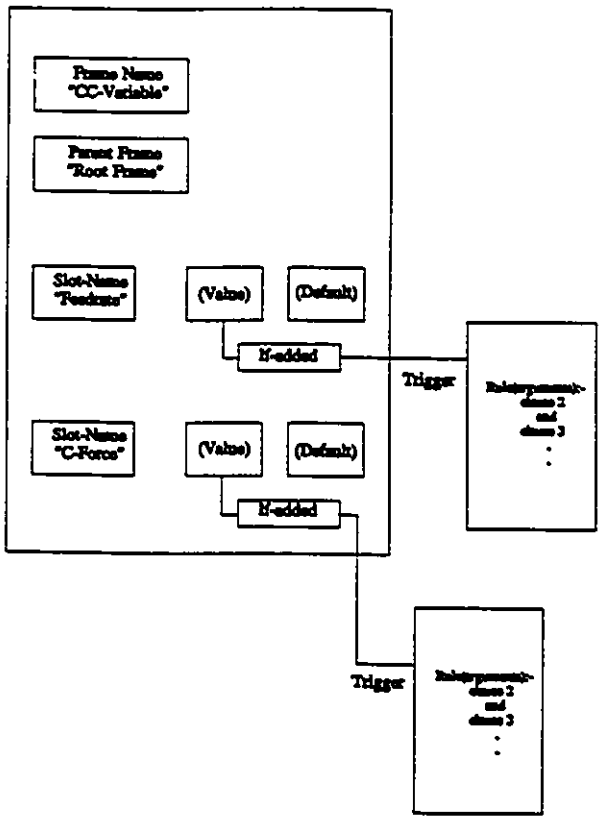


Figure 3.4. Sample frame used in the self-tuning controller.

holds values of the feedrate and the cutting force along with their default values. Each of these slots has attached predicates. These predicates behave as daemons, constantly monitoring the situation, and whenever a predefined situation arises, they "jump up" and carry out the appropriate function.

In other words, when a daemon will be activated is not defined by some other activating entity but by the daemon itself, in contrast to one procedure calling another procedure as found in conventional programs. For example, when the feedrate falls below a certain level the attached predicate associated with the slot "feedrate" is activated, and a switch from a linear to nonlinear controller is made (the reason for this switch was given in Section 2.0). Daemons are used not only for proper functioning of the adaptive controller (e.g. switching from linear to nonlinear controller), but also for maintaining the integrity of the controller (e.g. ignoring negative values of the parameter estimates).

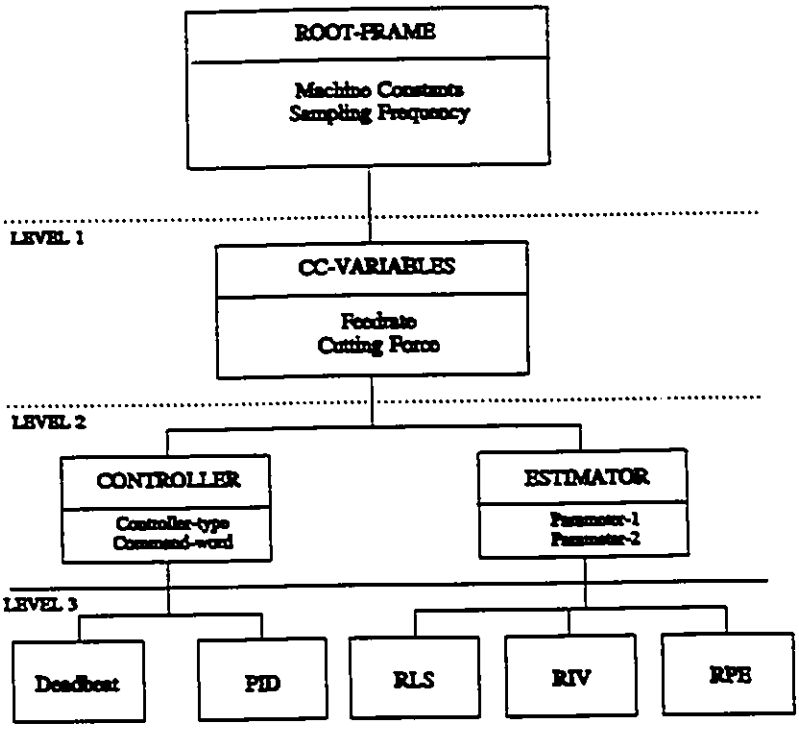


Figure 3.5. Hierarchical representation of frames.

When using frames as knowledge representation scheme, it is important to consider the manner in which they are arranged or connected to each other. Figure 3.5 shows how the frames are linked to each other in the system developed. The frames in the figure are arranged in a hierarchical manner. One way of looking at the relationship between the frames is with the concept of "refinement" [WAL88]. That is, this idea or object represented by frame becomes more general or more specific. It is usually advantageous to place more general frames at higher levels than specific frames in the hierarchy. Representing frames in a hierarchical manner allows for inheritance. Inheritance allows sharing of information among frames. That is frames at lower levels inherit the properties of frames at higher levels. Each frame has a parent, and it inherits the properties of its parent.

In figure 3.5, the frames "ROOT-FRAME" is the parent of the frame "CC-VARIABLES." The frame "ROOT-FRAME" holds slot values which are very general, like machine constants, sampling interval etc. The two frames "CC-VARIABLES" and "ROOT-FRAME" are related by the "similar" relationship because they represent entity sets whose members have sufficient properties in common to be regarded as similar, i.e., both the frames hold process variables. These slot values are inherited by the frames below the "ROOT-FRAME" in the hierarchy. The frames "CONTROLLER" and "ESTIMATOR" are related by the

sibling relationship to the frame "CC-VARIABLES" because the process variables stored above in hierarchy are required when calculating the "command word" and while estimating the parameters. Although this relationship does not seem clear cut, it avoids the redundancy of information. The frames "ESTIMATOR" and "CONTROLLER" are related to the frames below in the hierarchy by the "subset," or "isa," relationship. A "isa" relationship joins two concepts in which one concept is more general than the other. For example, "PID" is a "specialization" of "CONTROLLER," and "RLS" (recursive least square) is a "specialization" of "ESTIMATOR."

It is felt that frames should be used for the purpose of storing and retrieving data. Nondatabase actions and reasoning should be carried out with rules of logic. This separation between knowledge and reasoning provides conceptual clarity in knowledge representation, resulting in knowledge-based controllers that are easier to understand, maintain and work with [KAM88].

Knowledge-based systems are usually built with tools that support declarative programming as opposed to procedural programming. In procedural programming, the programmer instructs the computer by describing algorithms and data structures. In declarative programming, the programmer states specifications that the computation must satisfy. In the system developed, declarative language is

used for knowledge representation and reasoning, whereas procedural language is used for numerical algorithms.

The declarative language used is Prolog. It has many useful features, like built-in facilities for deductive retrieval through chronological backtracking and pattern matching via unification [KOS88]. The simulation of this case study was done in Arity Prolog, whereas the real-time implementation was done in Borland's Turbo Prolog. Although Turbo Prolog is less powerful (restricts the ability of the program to examine and modify itself) and significantly deviates from the standard implementations which follow Clocksin and Mellish [CLO87] closely, it was chosen because of its speed. Frames were implemented by exploiting the declarative nature of Prolog in a manner similar to that given in [ROW88] [WEI88] [JAY89]. Representing frames using Prolog is straightforward. Two predicates are required, one to create the frame and the other to create the slots within the frame whenever the frame is to be created. To create a new frame, the predicate "new-frame" is used. It requires two arguments: frame name and parent frame name. The predicate "new-slot" is used to create a new slot in a frame; it requires three arguments: frame name, slot name, and default slot value.

Knowledge Maintenance

Once the knowledge is represented through frames, they should be maintained,

i.e., slot values should be stored and retrieved whenever required. To get the slot value, the following information is supplied: the frame name and the slot name. The procedure to get the slot value is as follows. First, a search for the slot with the given slot name and frame is done. If it is found, a search for the slot is conducted in the parent frame, and so on further up the inheritance network. This operation is performed until the slot is found or until the root frame is reached.

To store slot values into the frame, a procedure similar to the preceding for getting slot values is used. The difference is that, if the slot has an attached predicate, it is executed first. Depending on the result of the execution, the value is stored or rejected. For example, in the frame "ESTIMATOR," when an attempt to store the value a_1 is made, the if-added predicate is attached to slot tests to see that the parameter is non-negative. Only if the test is successful will the parameter be stored; otherwise, it is ignored.

4.0 Reasoning Process

After a suitable means of knowledge representation is established, a suitable method to reason with this knowledge should be used. To do this, rules are used. These rules of logic use the knowledge stored in the frames.

In most applications, a means to couple numerical algorithms with the symbolic

processing environment should be provided. In our application, symbolic processing is done in Prolog and numeric algorithms are coded in the programming language C. To provide this shallow coupling [KIT86] between symbolic and numeric programs, an interface clause provided by the environment to access functions in C from Prolog is used. A C interface clause has the form

C_function_name(Arg1,Arg2,....,Arg*n*)

where C_function_name is the name of an interface function in C and Arg1, Arg2, ..., Argn are its arguments. These arguments may be constants or variables used within rules. Thus it can be seen that by interfacing C functions in the manner stated, we have a powerful programming style that provides both efficiency and ease of expression.

A sample rule in Prolog, to get the input and output of the cutting process, is given:

```
cutting_processIO:-
        C_getfeedrate(F),
        NewSlotValue(cc_variables,feedrate,F),
        C_getcuttingforce(C),
```

NewSlotValue(cc_variables,c_force,C).

In this rule, special C routines are called to get the feedrate and the cutting force of the process. These values are stored in the frame "CC-VARIABLES" in their respective slots as side effect when the subgoal "NewSlotValue" succeeds. When the feedrate is stored in the slot "CC_VARIABLES," the following attached predicate is executed:

```
if_added(cc_variables,feedrate,F):-
        F<=LOWER_LIMIT,!,
        NewSlotValue(controller,controller_type,non_linear).

if_added(cc_variables,feedrate,F):-
        F>LOWER_LIMIT,!,
        NewSlotValue(controller,controller_type,linear).
```

These attached predicates execute the switch between the linear and nonlinear when the feedrate falls below or above a certain value "LOWER_LIMIT."

The estimation of the parameters a_1 and b_1 can be carried out using one of the many parameter estimation algorithms. An example to illustrate the recursive

least-squares method is considered next. This algorithm is briefly recapitulated next [LJU83].

Consider a process expressed by a regression model:

$$y(t) = \theta^T \psi(t) \tag{3.9}$$

The unknown parameter vector θ can be determined by the equations:

$$\hat{\theta} = \hat{\theta}(t-1) + K(t)(y(t) - \psi^T(t)\hat{\theta}(t-1)) \tag{3.10}$$

$$K(t) = P(t-1)\psi(t)(I + \psi^T(t)P(t-1)\psi(t))^{-1} \tag{3.11}$$

$$P(t) = (I - K(t)\psi^T(t))P(t-1) \tag{3.12}$$

where θ represents the estimate of the unknown parameters, K(t) is the gain matrix, I is identity matrix, and P(t) is the covariance matrix.

In Prolog, the rule for estimation is written as

```
estimate_a1_b1(_):-
```

```
        get_slot_value(estimator,desired_force,Df),
        get_slot_value(estimator,c_force,Cf),
        Cf > Df - TOL,
        Cf < Df + TOL,!.

estimate_a1_b1(Est_type):-
        Est_type = rls,!,
        get_slot_value(rls,c_matrix,P),
        get_slot_value(rls,gain,K),
        get_slot_value(rls,p_estimate,THETA),
        C_rls(P,K,THETA,N_THETA),
        NewSlotValue(N_THETA).
```

The first rule switches off the estimator when the measured cutting force is close to the desired cutting force. The second rule is used to estimate the parameters using the recursive least-squares method. When the rule is executed, the covariance matrix P, the gain matrix K, and the previous estimates of parameters THETA stored in list form are retrieved from the frame "RLS." Although THETA is not stored in the frame "RLS," the inheritance mechanism is advantageously used to retrieve THETA from its parent frame "ESTIMATOR." The C program is called next to calculate the new parameter estimates as given

by equations (3.10)-(3.13). Then these new parameter estimates are stored in the frame "ESTIMATOR." The consistency of the parameter estimates is checked while storing.

The self-tuning controller can also be expressed as a rule and is written as:

```
Adaptive_controller(Est_type):-
        repeat,
        cutting_processIO,
        estimate_a1_b1(Est_type),
        cal_command_word,
        send_command_word.
```

When the rule is executed, the cutting process input and output are determined; the parameters a_1 and b_1 are estimated using the desired estimator "Est_type"; the coefficients in the numerator and denominator of the controller polynomial, which are expressed as functions a_1 and b_1, are changed; and the command word to be sent to the servo-mechanism is stored in a specified memory location. This process is repeated. An interrupt service routine which is activated every T_c seconds (controller sampling interval) takes the command word stored in memory and sends it to the servo-mechanism to adjust the feedrate.

The schematic diagram of the knowledge-based self tuning controller for the CNC milling machine is shown in the figure 3.6.

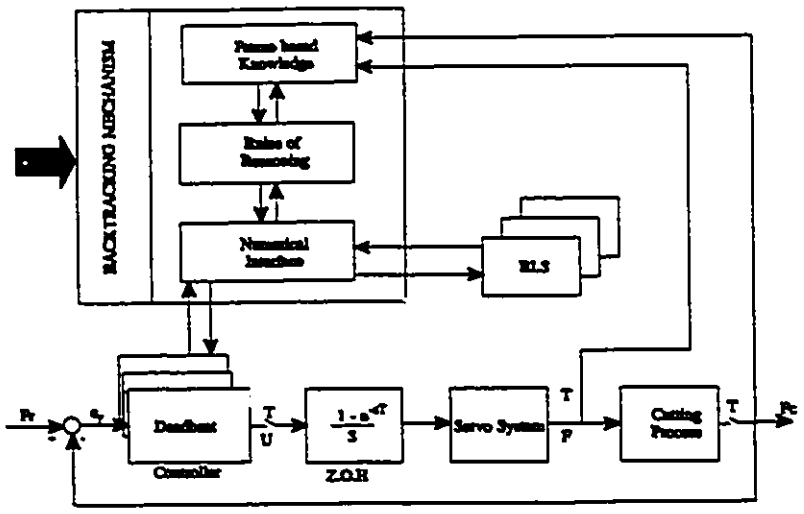


Figure 3.6. Block diagram of the knowledge-based adaptive controller.

5.0 Hardware Setup and Experimental Results

The controller was built for a vertical milling machine (Model TOSFA4, 5.5 kW spindle motor). The block diagram of the hardware setup is shown in Figure 3.7. The self-tuning controller was implemented on an Intel 386 based computer, which was interfaced to the NC controller (microprocessor) of the milling machine. A table dynamometer was used to measure the two in-plane components of the cutting force (F_x and F_y) by means of four piezoelectric charge transducers; two each in the x and y directions.

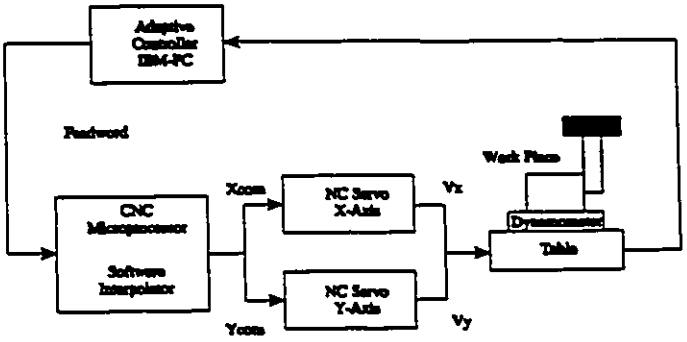


Figure 3.7. Real-time implementation of adaptive control using force measurement.

The force signals are sampled at t_s = 0.005 s. A resultant cutting force is calculated at each sampling instant as follows:

$$F_c = (F_x^2 + F_y^2)^{1/2} \tag{3.13}$$

The maximum value of the cutting force F_c during one spindle revolution (every 1/N s, where N is spindle speed in rev/sec) is then used by the controller, i.e., the controller sampling period T_c is set to 1/N. The tachometer signal (feedrate) is also sampled at t_s= 0.005 s, and an arithmetic mean was used for T_c.

Test Results

Figure 3.8 shows the simulated cutting force response using conventional (fixed parameter) PID controller, given by

$$u(t) = K_{ac}\int e_f + \beta\frac{de_f}{dt}dt + \gamma e_f \tag{3.14}$$

where K_{ac} is the gain in the PID algorithm, β is a constant,

$$e_f = \frac{(F_r - F_c)}{F_c} \tag{3.15}$$

and u(t) is the commanded feedrate. The result given in Figure 3.8 show the

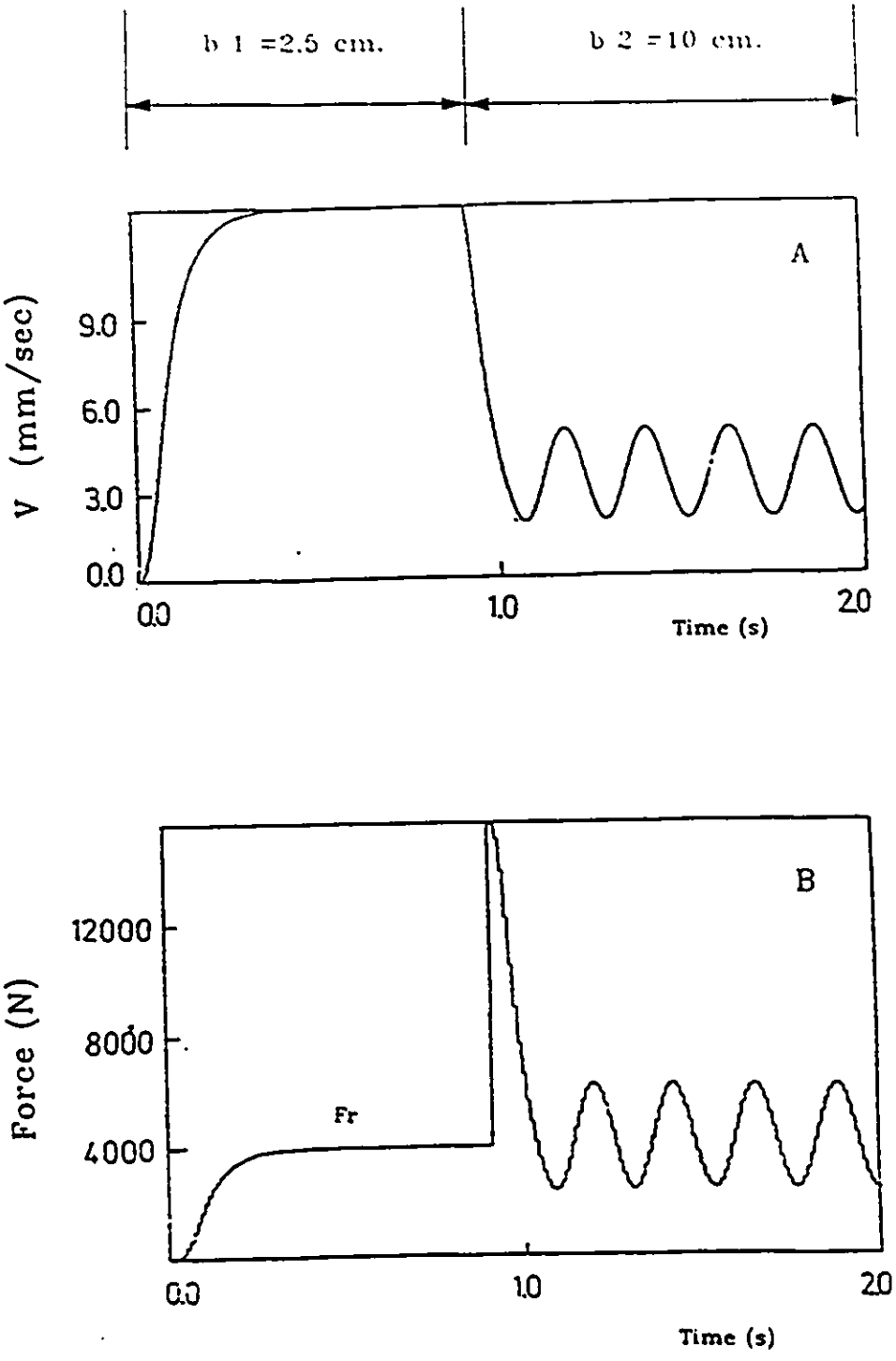


Figure 3.8. Experimental results with PID controller.
(A) Velocity, (B) Force transients.

response to a step change in the axial depth of cut from b_1 = 2.5mm to b_2 = 10mm. As seen while the system is stable in the first phase (b_1 = 2.5mm), it is clearly unstable in the second phase of the cut (b_2 = 10mm). It is possible, obviously, to stabilize the behaviour shown figure 3.8. by decreasing the value K_{ac}; however, this would be at the expense of the speed of response. Typical results of the real-time test results for the workpiece shown in figure 3.9 are shown in figure 3.10.

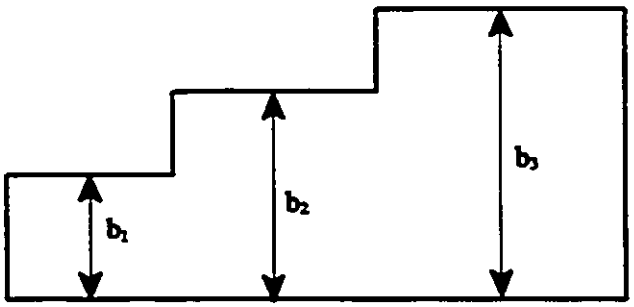


Figure 3.9. Profile of the depth of cut variation.

in figures 3.10(A) and 3.10(B), the responses of both the cutting force and the feedrate are given. The cut was performed in one direction (i.e., one-dimensional cutting), and the variations in the axial depth of cut are given on top of trace A. The spindle speed used in this experiment was 725 rev/min. All other cutting conditions are listed in the figure.

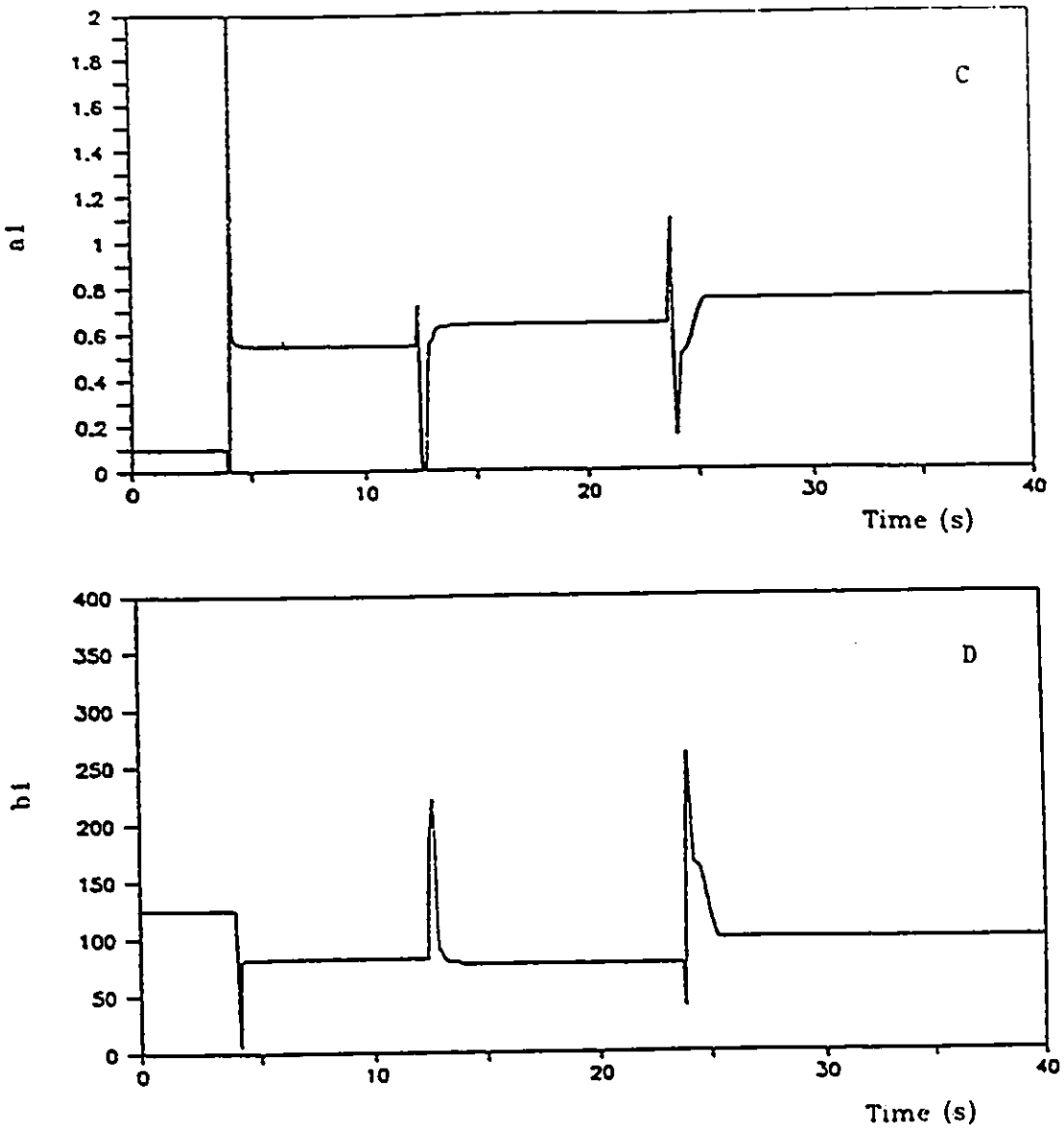
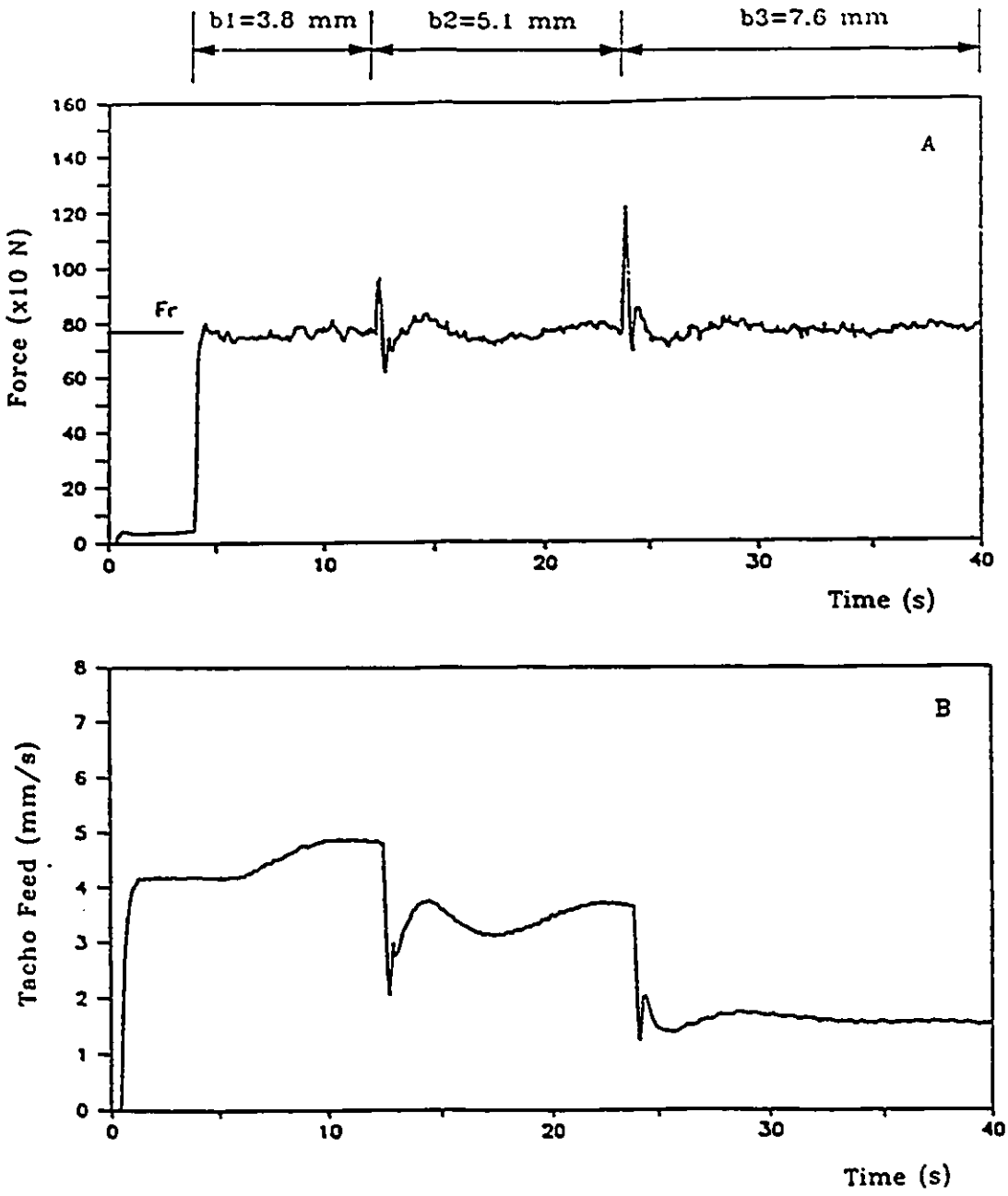


Figure 3.10. Experimental results with deadbeat controller. (A) Cutting force response. (B) Feedrate response (C) Evolution of parameter a1. (D) Evolution of parameter d1.


b1=3.8 mm
b2=5.1 mm
b3=7.6 mm
A
Fr
Force (x10 N)
Time (s)
B
Tacho Feed (mm/s)
Time (s)

Figures 3.10(C) and 3.10(D) show the estimated the a_1 and b_1 with respect to time. Except at time instants where the depth of cut has changed, the parameter estimates are smooth. This is because the estimator has been switched off when the measured cutting force is close to the desired cutting force. It is seen that the cutting force is rather well regulated at a reference force F_r of 870N.

6.0 Limitations and Enhancements to the System

Prolog, which has been used extensively for knowledge representation and reasoning, uses chronological back tracking and is therefore exponential in time. To curtail the search space of Prolog computations, "cuts" and "fails" are used, which dynamically prune the search tree. Cuts are considered equivalent to "goto's" in logic programming [BOB85] and they affect the declarative semantics of the program. This contradicts the claim of system conceptual clarity and system maintainability. However, by parsimoniously using harmful (red) cuts, the severity of the problem is reduced to a large extent. For example, the code given previously for switching between the linear and nonlinear controller uses "green" cuts. The same code can be written as:

```
if_added(cc_variables,feedrate,F):-
        F <= LOWER_LIMIT, !,
        NewSlotValue(controller,controller_type,non_linear).
```

```
if_added(cc_variables,feedrate,F):-
        NewSlotValue(controller,controller_type,linear).
```

The cuts in this piece of code are considered "red" because the same clauses without the cuts would be logically incorrect.

The most recent value of an attribute for an frame is stored in a slot. Rules of logic interact with a frame by inserting and retrieving values from their slots. In a real-time system, one often need to reason about the behaviour of the system over time. In the system developed in this chapter, reasoning over time is not possible. The system can be extended to allow reasoning over time, one approach is by making certain slots in the frame point to ring buffers. The ring buffers hold a series of values and their associated time tags.

The discrete-time controller for the CNC milling machine had fairly long sampling periods (114ms), and the number of on-line estimations of the parameters was small, thereby allowing the symbolic and numeric computations to be completed within one sampling period. For more complex controllers, where the sampling period is very small, processors with Von Neumann type architecture are inadequate. Lately, relatively inexpensive parallel processors (INMOS transputer) are becoming available. These can be plugged as add-on boards to

microcomputer. It is felt that by using parallel processors, one can design controllers with much more severe time constraints. Another approach to speed up computation is to use dedicated hardware to do symbolic computations, because in a knowledge-based controller, maximum time is spent in symbolic computation. At present, dedicated hardware is available to do symbolic computation (e.g. the LISP coprocessor by Texas Instruments).

An alternative approach to designing the knowledge-based controller is with the help of object oriented programming environment that allows persistence, specifically an C++ based system. Persistence allows objects to live past the life of the program. Such systems with the addition of security and concurrency control are called object oriented databases (OOD). Knowledge in such systems is represented as a set of persistent objects. For example, the class definition of a robot joint along with the controller parameters can be represented by:

```
class Joint {
private:        // private data members
        float Kp, Kd, Ki .... etc;
        .....

public:
        // Constructors and Destructors
        Joint();
        ~Joint();
        // Accessors and Mutators
        setKp(float Kp = <default value>);
```

```
    ..
    ..
};
```

Most of the arguments for not using traditional relational databases as knowledge bases have been overcome by OOD. Some of these problems include:

1. The complexity of the kinds of facts that can be recorded in a typical knowledge base is far greater than what a relational database (RDB) can handle. Facts stored in the RDB must conform to a rigid form, and exceptions are not allowed. In contrast, OOD systems usually cover a wide ranges of inhomogenous knowledge.
2. RDB explicitly avoid the meaning of the data they store, or any logical consequence of the data. They are usually only concerned with records, which are purely uninterrupted data structures. One important consequence of this difference is that there is no notion of entailment - and thus inference of the domain. OODs not only allow complex queries as in RDB to be answered, but also allow interpretation of the knowledge structures.
3. OOD can be used to represent generic descriptions in a way that allows new classes or instances to be recognized, or classified under them. RDBs

are simply flat representations of facts, and do not have conditions for recognizing new items as members of classes to which they have not been explicitly stated to belong.

4. The schema of an RDB that corresponds to concepts or classes in a knowledge base remains fixed. This is not so with a typical knowledge base, the class structure frequently changes. OOD can not only have more classes than instances but also can handle problems where the instances to be stored greatly outnumber the relations, usually by orders of magnitude.

At the time of writing, there were at least 36 different object oriented databases available in the market. One of them is freely available for research and educational institutions (SOS - Stone Object System).

In an industrial environment, control must achieved be in the presence of more constraints. For example, variables such as tool wear, tool deflection, and chatter vibration should be considered. Many of these are difficult to measure and must be inferred from variables that can be monitored (e.g cutting force). In the system developed it is envisaged that incorporating the aforementioned constraints will be easier when compared to a conventionally developed system.

7.0 Final Analysis

Supervision of the adaptive controller is necessary due to the incompleteness of the control law. Hence, for the proper functioning of the controller, it is necessary to know the physical limitations of the system being controlled along with the logical functionalities that validate the mathematical theory underlying the control law. In this chapter, we have used this knowledge to successful by build a frame-based supervisory adaptive controller for force regulation in a CNC milling machine. A frame-based knowledge representation scheme was used because it provides inheritance and data encapsulation, thereby allowing structured implementation and also enhanced maintainability of programs. Simulations and experimental obtained show that the performance of the conventional knowledge-based adaptive controller is as good as that of conventionally designed adaptive controller. The superiority of the knowledge-based approach lies in the easy incremental growth and maintainability it provides.

The limitation of the system is that it uses exponential in-time algorithms therefore making it unsuitable for applications with very severe time constraints. This limitation can be overcome by using parallel processors or dedicated hardware to do symbolic processing. The system can also be enhanced to handle other constraints such as tool wear, tool deflection, and chatter vibration.

Although knowledge in a knowledge based controller is application dependent, the fundamental structure of the controller is similar to that shown in Figure 3.6. Other knowledge-based controllers can easily be constructed on lines similar to those presented in this chapter. Before attempting to build a controller using this approach, one should consider the limitation of this approach, i.e., the implications of using exponential time algorithms in real-time applications [LAF88].

Chapter 4

Fuzzy Logic Based Control Systems

1.0 Introduction

When we communicate with computers, we must precisely specify all the information we want our computers to process. If we want a computer to store information about a person's car, for example, we would specify exact values for the car's colour, age, and weight. Precision, however does not characterise the we way we communicate with each other. When we talk about the person, we say things like "He is middle age, tall, slim and with a fair/dark complexion." In the case of control systems too, when communicating with the computer it is necessary to describe the process to be controlled precisely, but this is not the case in practice since the process model and parameters may not be known to a sufficient degree of accuracy. The question that arises is whether it is possible to communicate with computers the way we talk to each other. The answer is that it is possible if our software/hardware incorporates *fuzziness*, a concept that properties of things need not be all-or-none.

Fuzziness has a short but illustrious history. For thousands of years, philosophers

based systems of logic on the idea that things were always either true or false. In the 1920's the mathematician Lukasiewicz challenged this idea and proposed that a gradation exists between these two extremes. A decade later, the Physicist Black [BLA37] introduced *vagueness*, the notion that a set could contain elements that were partly in, and at the same time, partly out of the set. In the mid 1960's, Zadeh [ZAD65a] [ZAD65b] constructed a formal methodology for handling sets like the one proposed by Black, and he named these structures *fuzzy sets*. These seminal papers by Zadeh greatly influenced the pioneering research by Mamdani [MAM74] [MAM75] and his colleagues on the use of fuzzy logic to control systems. Fuzzy sets were later modelled as points in a hypercube by Kosko [KOS90]. Kosko's model makes it easier to answer questions about fuzzy sets by drawing appropriate pictures. Although, the idea of using fuzzy logic to design control systems is over two decades old, in recent years it has emerged as one of the most active areas of research. However, at present there still seems to be no consensus on a systematic procedure for designing fuzzy logic controllers (FLC). In this chapter several theoretical issues related to FLC design are discussed. In the chapters that follow the realization of FLC for two real-world problems of increasing complexity are discussed. These real-world problems include force regulation for robotic deburring [LIN88] [BON89b] [BON91], and obstacle avoidance for autonomous navigation in a mobile robot [BUR90].

2.0 Fuzzy Sets and Fuzzy Logic

In this section a summary of some of the concepts of fuzzy set theory and fuzzy logic that are relevant to the design of fuzzy logic controllers are described. These concepts are dealt in a more thorough fashion in [KLI88][ZIM85].

Fuzzy sets often correspond to subjective impressions of things we encounter in daily life. Adjectives (like TALL, HOT, HEAVY) are good candidates for fuzzy sets. In the realm of fuzzy sets, the most important idea is *grade of membership*, a number which describes the degree to which an element is in a set. The formal definitions of fuzzy set terminology is given below.

Fuzzy Set: Let X be the universe of discourse or the universal set, then a fuzzy set A in X is characterized by a membership function which is of the form

$$\mu_A : X \rightarrow [0,1] \tag{4.1}$$

where [0,1] denotes the interval of real numbers from 0 and 1, inclusive. Thus the fuzzy set A in X is defined as set of ordered pairs:

$$A = \{(a, \mu_A(a)) \mid a \in X \} \tag{4.2}$$

When X is continuous then the fuzzy set A is written as:

$$A=\int_X \mu_A(a)/a \tag{4.3}$$

In the case when X is discrete the fuzzy set A is given by:

$$A=\sum_{i=1}^{n} \mu_A(a_i)/a_i \tag{4.4}$$

The support set of a fuzzy set A is a crisp set of elements of X whose grade of membership is greater than zero. If the support set consists of one element whose grade of membership is 1, then it is called a fuzzy singleton. Specifically the element in X whose grade of membership is 0.5 is called the crossover point.

Operations on Fuzzy Sets:

Let A and B be two fuzzy sets in X with grade of membership functions μ_A and μ_B respectively. The set theoretic operations of *union, intersection* and *complement* for the fuzzy sets are defined via membership functions for all u ε X as follows:

$$\mu_{A\cup B} = \max\{\mu_A(u),\mu_B(u)\} \tag{4.5}$$

$$\mu_{A\cap B}(u) = \min\{\mu_A(u),\mu_B(u)\} \tag{4.6}$$

$$\mu_{A'}(u) = 1 - \mu_A(u) \tag{4.7}$$

The *cartesian product* of fuzzy sets $A_1,...A_n$ belonging to $X_1,...X_n$ is a fuzzy set with the membership function

$$\mu_{A_1x...A_n}(u_1,u_2,..u_n) = \min\{\mu_{A_1}(u_1),....\mu_{A_n}(u_n)\} \tag{4.8}$$

A n-ary *fuzzy relation* is a fuzzy set in $X_1x....X_n$ expressed by:

$$R_{X_1x...X_n} = \{((u_1,...,u_n),\mu_R(u_1,...,u_n))|(u_1,...,u_n)\varepsilon X_1x...xX_n\} \tag{4.9}$$

If R and S are fuzzy relations in UxV and VxW, the composition of R and S is a fuzzy relation given by

where * is an operator in the class of triangular norms.

$$R \circ S = \{[(u,w), sup_v(\mu_R(u,v) * \mu_S(v,w))], u \in U, v \in V, w \in W\} \qquad (4.10)$$

Visualization of Fuzzy Sets: Recently Bart Kosko [KOS90] has proposed a hypercube model that shows how to draw appropriate figures to illustrate fuzzy sets. These figures make it easy to answer questions about fuzzy sets. In the model, each element in a fuzzy set corresponds to an axis of a graph, and the grade of membership of each element corresponds to a point on the axis. The maximum value on each axis is 1 and the minimum value is 0, and points within the graph represent fuzzy sets. This method of representation for fuzzy sets can be used for sets containing any number of elements, in the case of two and three elements the boundaries of the graph define a square and a cube. For sets of more than three elements, the boundaries define a hypercube. To illustrate how Kosko's model can be used to answer certain questions about fuzzy sets, lets us consider an example of a fuzzy set with two elements. Figure 4.1 illustrates this fuzzy set consisting of elements X1 and X2. A is a fuzzy set whose elements have a grade of membership 1/4 and 3/4. It is represented by a point (1/4,3/4). The points on the corners of the graph represent nonfuzzy sets, as the coordinates of these points correspond to grades of membership for elements which are either completely in a set or completely out of a set. The distances among points in the figure reflect the features of fuzzy set theory.

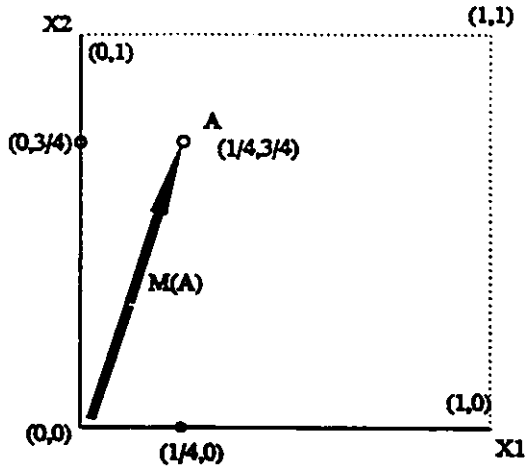


Figure 4.1. Size of the Fuzzy Set.

Size of the Fuzzy set: In classical set theory, the size of a set is often taken to be the number of elements in the set. In fuzzy set theory the size of a set is the sum of the grade of memberships of its elements. For the case shown in figure 4.1, the size of the fuzzy set A is the distance measured "city-block style" from the origin to the point A in the figure. This distance is 1/4+3/4, or 1. If one thinks of distance as "resemblance", a set's size shows how closely it resembles the empty set - the greater the size, the less the resemblance.

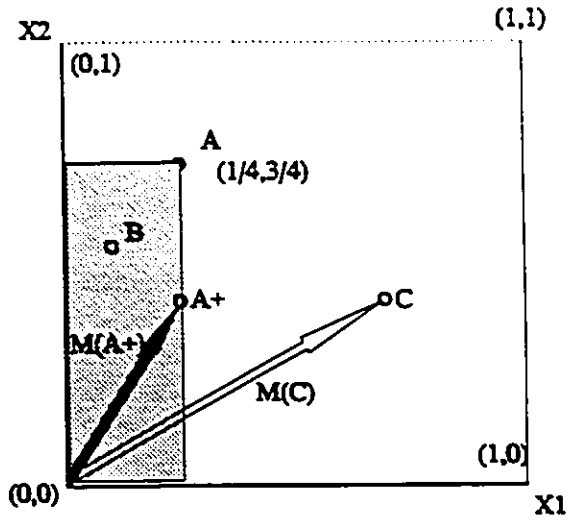


Figure 4.2. Representation of Fuzzy Subsets.

Fuzziness of a fuzzy set: The idea of resemblance helps define fuzziness. The fuzziness of a fuzzy set depends on its resemblance to the nonfuzzy set it resembles the most, and the nonfuzzy set it resembles the least. In figure 4.1, these resemblances can be determined by measuring the distance from the point A to the nearest corner (0,1) and the furthest corner (1,0). The ratio of these distance measures the fuzziness of A. In the above case it is (1/4+1/4) divided by (3/4+3/4), or 1/3. One can easily see that the measure of fuzziness is maximum when the point A is equidistant from all four nonfuzzy sets. The ratio of the distances in this case becomes one. Fuzziness is minimum at each corner.

Fuzziness of fuzzy subsets: In fuzzy set theory, B is a subset of A if all B's elements have grade of membership less than or equal to the grades of

membership of A's corresponding elements. Graphically, this means that B is within the shaded rectangle as shown in the figure 4.2. Kosko's model incorporates fuzziness into subsets. The model specifies that fuzzy set C can partially be a subset of A if its is not within A's rectangle, and it gives a way of determining the degree of *subsetness*. A+, the subset of A which most closely resembles C, is critical in this determination. According to this model, the extent to which C is a subset of A is the ratio of the size of A+ to the size of C (ratio M(A+) to M(C)). In the figure 4.2, A+ is the point on A's rectangle closest to C.

Fuzzy logic, approximate reasoning and the compositional rule of inference: The membership function for fuzzy sets provides a direct linkage to fuzzy logic. The degree of membership of x in A corresponds to the truth value of the statement: x is a member of A where A defines some propositional or predicate class. When $\mu_A(x) = 1$, the proposition A is completely true, and when $\mu_A(x) = 0$ it is completely false. Values between 0 and 1 assume corresponding values of truth or falsehood.

Generally in two valued logic, truth tables were useful in determining the truth value of a statement or a well formed formula. In most cases, this is not possible in fuzzy logic since there may be infinite number of truth values. To overcome this problem many researchers have suggested generalized modus ponens for fuzzy sets. They differ from the standard modus ponens in that, the statements which

are characterized by fuzzy sets are permitted and the conclusion need not be identical to the implicand in the implication. For example, let A, A1, B, B1 be statements characterized by fuzzy sets. Then one form of the generalized modus ponens read as:

Premise: x is A1

Implication: if x is A then y is B

Conclusion: y is B1

An example of this form of modus ponens is given as

Premise: The error is very large

Implication: If the error is large then the control_output is large

Conclusion: The control_output is very large

Although different forms of fuzzy inference have been proposed, we present only Zadeh's original compositional rule of inference [ZAD71]. Let X and Y be two universes and let A and B be fuzzy sets in X and X × Y respectively. Define fuzzy relations $R_A(x)$, $R_B(x,y)$, and $R_C(y)$ in X, X × Y, and Y respectively. Then the compositional rule of inference is the solution of the relational equation.

$$R_C(y) = R_A(x) O R_B(x,y) = \max_x \min\{\mu_A(x), \mu_B(x,y)\} \qquad (4.11)$$

where the symbol O signifies the composition of A and B.

Fuzzy logic seems to be a viable alternative to first order predicate logic used in AI, but many AI researchers are reluctant to accept fuzzy logic. This is so because fuzzy logic accommodates uncertainty by an approach to 'semantics' which is quite distinct from that used in conventional logic.

Linguistic Variables: They provide a link between natural language and representations which accommodate quantifications such as fuzzy proportions. We will next define a linguistic variable formally and show how it is related to fuzzy logic through an example. Informally, a linguistic variable is a variable that assumes a value consisting of words, sentences rather than numbers. For example HEIGHT might assume the values of very short, short, medium, tall, very tall. These values, which are themselves linguistic variables, may in turn, each be given meaning through a base universe. For example for each of the variables of HEIGHT, we associate a fuzzy set consisting of ordered tuples $\{(x/\mu_A(x)\}$ where $x \in X = [2,8]$ the universe (feet).

A formal and elegant definition of linguistic variable is one that is based on language theory concepts. It is defined as a quintuple

(x,T(x),X,G,M)

where

x is the name of the variable (HEIGHT in the above case),

T(x) is the terminal set of x (very short, short, tall, etc.),

X is the universe of discourse (range of heights)

G is the set of syntactic rules, the grammar which generates the values of x, and

M is the semantic rule which associates each value of x with its meaning M(x), a fuzzy subset of X.

The grammar G is further defined as the tuple (V_N,V_T,P,S) where V_N is the set of nonterminal symbols, V_T the set of terminal symbols from the alphabet of G, P is the set of rewrite (production) rules, and S is the start symbol. The language L(G) generated by the grammar G is the set of all strings w derived from S consisting of symbols in V_T. Thus, for example, using V_N = {A,B,C,D,E,S} and the following rules in, we generate the terminal string "not short and not very tall."

P:	S→A	A→A and B	C→D	D→short
	S→S or A	B→C	C→ very C	E→tall
	A→B	B→not C	C→E	

The string is generated through the application of the following rules:

S→A→A and B→A and not C→A and not very C→A and not very E→A and not very tall → B and not very tall → not C and not very tall → not D and not very tall → not short and not very tall

The semantic rule M gives meaning to the values of HEIGHT. For example, we might have M(tall) = $\{(x/\mu_{tall}(x)) \mid x \in [2,8]\}$ where $\mu_{tall}(x)$ is defined as function.

3.0 Fuzzy Logic Control

Befcre examining fuzzy logic control of systems further, it is important to know why such systems are required. To answer this question let us consider figure 4.3 which shows the classical model of process control.

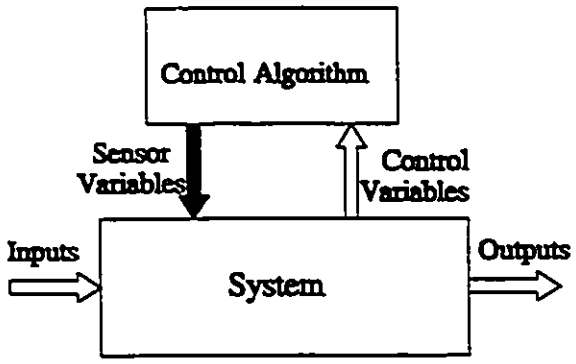


Figure 4.3. Classical approach to process control.

As shown in the figure, classical control theory is concerned with four basic components: the system to be controlled, a set of process parameters that can be observed (the sensor or the control system input variables), a second set of process parameters that can be directly controlled (called the control or control system output variables), and a control algorithm that transforms the sets of sensor variable observations into set of control variables settings. The control algorithm is derived from a model of the process to be controlled. When the model is accurate and the control algorithm evaluations are faithful, the performance and stability of the control algorithm can be guaranteed within known limits. Control is well known and understood, and is the control mechanism of choice when applicable. However, in many situations, classical control may not be applicable for several reasons:

- There may be no complete model of the process, or available models maybe too complex or make unacceptable assumptions.

- Control algorithms derived from classical control techniques do not respond very well to noise in sensor variables measurements.

- The stability and performance available from classical control algorithms may not be adequate for the requirements for the task at hand.

Fuzzy Logic based control systems: These were developed to overcome some of the aforementioned problems. The model for fuzzy logic control is shown in the figure 4.4.

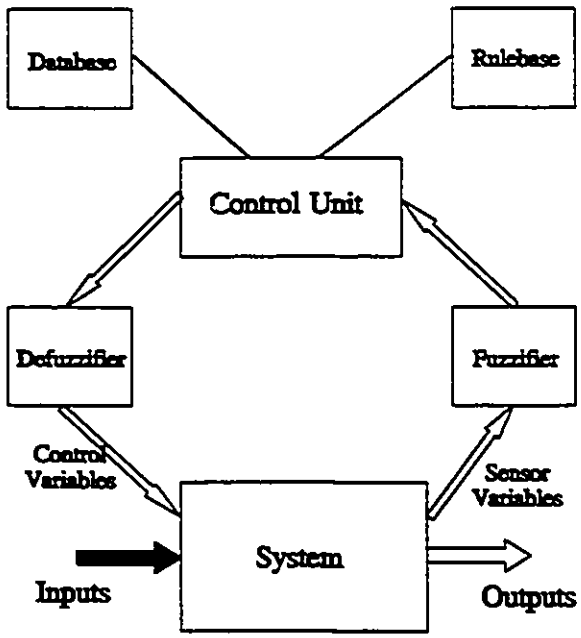


Figure 4.4. Setup of a typical Fuzzy Logic Controller.

The sensor readings, which are assumed to be nonfuzzy, must be converted to a fuzzy set form, and this occurs in a process called fuzzifier. The definitions of the fuzzy predicates are stored in a database associated with the controller. The linguistic control knowledge is stored as a set of fuzzy conditional statements in a rule-base and can be seen as specifying a fuzzy relational algorithm. The control unit combines the fuzzified sensor data with the database and the rule-base

to determine a fuzzy action specification. Finally, this is converted to specific nonfuzzy control action by the defuzzifier. One assumption behind the above model is that the conditional statements in the rulebase relate the sensor parameter and their rate of change to the control parameter or their increments.

4.0 Literature survey of applications of Fuzzy Logic

Since the introduction of fuzzy logic by Zadeh in 1965, fuzzy logic has been used successfully in a number of control applications. The first application of fuzzy set theory to the control of dynamic processes was reported by Mamdani and Assilian [MAM74]. They were concerned with the control of a small laboratory scale model of a steam engine and boiler combination. The control problem was to regulate the engine speed and the boiler pressure. Despite the nonlinearity, noise and the strong coupling in the plant, they managed to get acceptable control using a fuzzy logic controller. Kickert and Lemke [KIK76] applied fuzzy logic to design a controller for a laboratory scale warm water plant. The aim was to control the temperature of the water in one of the compartments by altering the flow rate through a heat exchanger contained in the tank. A secondary control task was to ensure fast response to step changes in the outlet water temperature set-point. The first experiment on an industrial plant with fuzzy logic controller was undertaken by Rutherford and Carter [RUT76]. They developed a controller for the permeability at the Cleveland sinter plant, and showed that the fuzzy logic controller worked slightly better than the PI controller.

Tong applied fuzzy logic to a pressurised tank containing liquid [TON76]. The problem was to regulate the total pressure and the level of liquid in the tank by altering the rates of flow of the liquid in the tank and the pressuring air. Good control was achieved despite the non-linearity and strong coupling. However it was no better than performance obtained by a controller designed using conventional techniques. Tong *et al* [TON79] also examined the behaviour of an experimental fuzzy logic control algorithm on an activated sludge waste water treatment process. They concluded that a fuzzy algorithm based on practical experience can be made to work on this difficult process. The success obtained by Mamdani and Assilian in the control of a steam engine led them to study temperature control of a stirred vessel which formed the batch reactor process, a nonlinear, time varying gain and delay process [MAM75]. The results obtained showed that, processes can be controlled effectively using heuristic rules based on fuzzy control, but to achieve good control the fuzzy rules must be correctly formulated, to take account of time delays when they occur. Around the same time, the techniques of fuzzy logic were applied by independent groups over a wide variety of processes. Ostergaard [OST76] applied it successfully to a heat exchanger; Van Amerongen [VAN77] applied fuzzy sets to model the steering behaviour of operators of the Durkee plant of the Oregon Portland cement company using fuzzy algorithms. In this and related work on cement kiln control, it was noted that human operators could successfully control their operation. Attempts to obtain adequate mathematical models of the kiln process have been

generally unsuccessful. Thus, conventional control techniques have not been widely successful for cement kiln control. In contrast, fuzzy control has been applied successfully, reducing product variability, including shift-to-shift characteristics differences between operators.

In the early eighties, much of the work done during the seventies was formalized, mainly in the direction of algorithm completeness, interaction, rule competivity, and controller stability [COZ81] [TOG83]. Recently, a number of new applications of fuzzy logic have come from Japan. Sugeno *et al* [SUG89] have proposed a fuzzy algorithm of a model car using oral instructions. The controller was designed using the operator's knowledge and experience. Ono *et al* [ONO89] have described an algorithm for refuse incineration. Experimental results substantiate the claim that the fuzzy control system is practical to built and beneficial. Fujimoto et al [FUJ89] have employed fuzzy logic for speech recognition. Yamakawa [YAM89] presented a high-speed fuzzy controller hardware system, which has fifteen control rule boards and one defuzzifier. The fuzzy controller facilitates approximate reasoning at one million inferences per second and can be used for various purposes by programming on each control rule board. This controller made an inverted pendulum stand on vehicle by using seven rules. The hardware is useable in many other applications which need swift approximate reasoning. Hirota [HIR89] developed a robot arm which was fuzzy controlled and was able to play two dimensional ping pong. Twenty five rules

were used and input to the rules were ambiguous instructions generated by the robot using imagery data from a camera. The point to which the robot tip should move was calculated based on fuzzy inference method. The system was controlled by a 16-bit personal computer and worked in real time.

One deployed interesting application is the building environmental controller by Mitsubishi. Most large office buildings have an environmental control system for heating and cooling. The Mitsubishi controller uses 25 rules for heating and 25 rules for cooling. The inputs to the system are eight bit accurate and include items such as room and wall temperature and rate of change of these temperatures. The system was developed in about 4 months, which included three months of tuning. The improvement of the fuzzy logic controller over PID controller was dramatic. The fuzzy system reduces heating and cooling times by a factor of five, and temperature stability by a factor of two; when doors are opened and closed the perceived disturbance in the room environment is reduced. The fuzzy system also requires fewer sensors and, most importantly, reduces the energy consumption of the heating and cooling system by 24%.

At the 1989 American control conference, one session was devoted to fuzzy systems where six papers were presented. One investigated the upper bounds for a multivariable fuzzy controller under Godel's implication and proposed a generalized multivariable structure [GUP89]. Another used FLC to control a

laboratory scale fluidized bed [KOF89]. Two papers considered improvements to SISO FLC implementations [CHU89] [MAN89]. One presented a comparative study of state feedback and FLC using computer simulation and hardware implementation of a cartpole balancing problem [BER89]. One used FLC to control the tip position of a single-link flexible manipulator.

Although, much work has been done in the area of fuzzy logic control systems, developing them for a specific application is an art. In the chapters that follow, fuzzy logic controllers for two different applications of increasing complexity are developed.

Chapter 5

Fuzzy Logic Controller for Robotic deburring

1.0 Introduction

Generally most machine parts need some form of deburring. Presently, it is estimated that $3.9 billion a year is spent on deburring, and most of it is done manually. Automating this process using a robot yields benefits in cost, part-assembly fit, and final product performance. The robotic deburring process begins with interpreting a CAD model of the machined part, fixturing and tool. Next, a path is generated based upon the workspace processing. The path is checked for collision in the approach phase and deburring phase. This path is transferred to workstation controller where it is used to direct the robot with real time adjustments based upon force feedback signals corresponding to the depth of cut.

There are several impediments to classical or modern control approaches to deburring. They are:

- the deburring process is random and uncertain,
- the concepts of optimal quality and optimal economy are related to 'operator skills',

- deburring operation is fairly fast, therefore does not allow sufficient time for a great number of calculations, and
- in-process measurement of part quality is not precise.

When robots are used for deburring, significant amount of forces develop when a robot interacts with the workpiece, which makes developing a good controller difficult. Many researchers [PAU82] [ZAL82] [TLU86] have proposed that an independent end effector could alleviate some of problems with conventional robots. In this method the robot is used only as a coarse positioning device, with fine motion control performed by an independently controlled actuator called the independent end effector mounted between the cutting tool and the robot's wrist. Some of the advantages using a independent end effector over space control systems are listed below [BON89b]:

1) The force measurements and the actuator motion are both performed in the task space so that no coordinate conversion is required.
2) Only the independent end effector is controlled and not the entire arm, allowing a fast and accurate response to be more easily achieved.

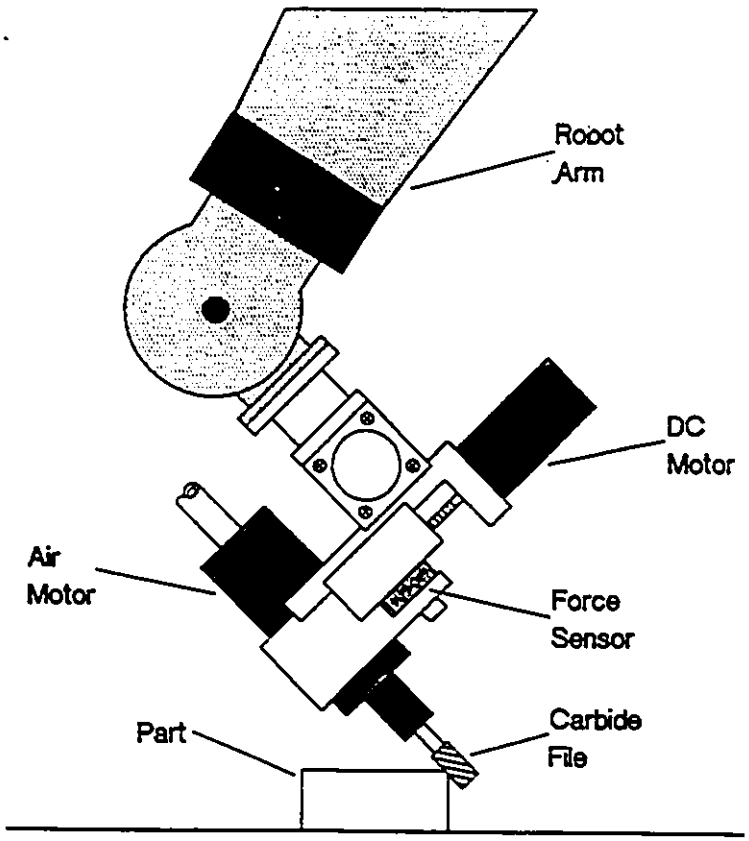


Figure 5.1. Independent end effector for robotic deburring.

3) Positioning errors occurring after the joint actuators due to link flexibility, joint flexibility and joint backlash are accounted for.

4) Only minor communication with the robot controller is required, making the method nearly independent of the robot host type.

The main disadvantage of the independent end effector is the need to develop the special end effector. The independent end effector developed for this purpose is shown in the figure 5.1.

In this chapter, the design of the fuzzy logic controller for force control [LIN88] will be discussed; a important part of the robotic deburring system. The objective of the control system is to achieve a desired depth of cut by maintaining a constant cutting force. The controller is simulated, implemented and tested in real time. The results from these tests are also given.

2.0 FLC for Force regulation

The structure of the fuzzy logic controller is shown in figure 5.2.

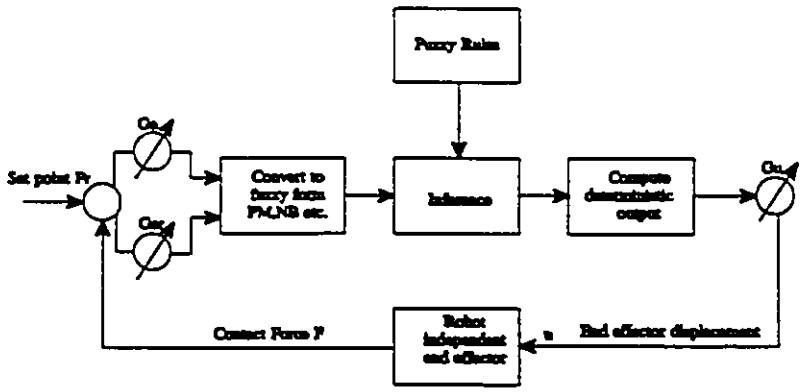


Figure 5.2. Schematic for the fuzzy logic controller for robotic deburring.

The fuzzy logic controller is a three dimensional algorithm which infers the controller output (end effector displacement) u, based upon the contact force error (e) and the force error rate (er). The error and the error rate are first quantized, then converted to a fuzzy form in order to compare them against the antecedents of fuzzy control rules. Several rules might apply, so their consequents are modified according to the degree of fulfillment and combined together to form a fuzzy set. This is defuzzified and a deterministic control output is generated, which is applied to the process.

The design and implementation procedure for the fuzzy logic controller described above is accomplished according to the following five steps.

Step 1. Identifying the process: Input and control variables are defined, which determine the states of the process to be observed and the control actions to be considered. For this application, the controller inputs are force error (e) and force error rate (er). The controller output is the change in end effector displacement (u). The input force error is scaled and quantized into fourteen levels, whereas the change in force error is scaled and quantized into thirteen levels. The output of the fuzzy logic controller is also quantized to fifteen levels. The scaling factors Ge, Ger and Gu (see figure 5.2) are chosen appropriately by determining the range of values the variables e, er and u can take.

Step 2. Defining membership functions: Membership functions are constructed that determine the way observations of the process variables are expressed as fuzzy sets. In this application fuzzy sets are formed to represent the discrete support universes consisting of 14 elements for the force error, 13 elements for force error rate and 15 elements for change in end effector displacement. Appropriate membership functions are assigned to each element of the support set using the definitions by Mamdani [MAM75], an approach employed in preference to a functional method [KIC76] because of manipulative simplicity. These yield

the fuzzy set values shown in table 5.1, 5.2, and 5.3. The variable identifiers used to describe the fuzzy values have the following meaning;

PB	Positive Big
PM	Positive Medium
PS	Positive Small
PO	Positive Zero
ZO	Zero
NO	Negative Zero
NS	Negative Small
NM	Negative Medium
NB	Negative Big

and are consistent with those reported in most of the application studies. The terms NO and PO are introduced to give finer tuning around the equilibrium state. Term PO defines a region slightly above zero and NO a region that is slightly below zero in the state space.

	-6	-5	-4	-3	-2	-1	-0	+0	1	2	3	4	5	6
PB	0	0	0	0	0	0	0	0	0	0	.1	.4	.8	1
PM	0	0	0	0	0	0	0	0	0	.2	.7	1	.7	.2
PS	0	0	0	0	0	0	0	.3	.8	1	.5	.1	0	0
PO	0	0	0	0	0	0	0	1	.6	.1	0	0	0	0
NO	0	0	0	0	.1	.6	1	0	0	0	0	0	0	0
NS	0	0	.1	.5	1	.8	.3	0	0	0	0	0	0	0
NM	.2	.7	1	.7	.2	0	0	0	0	0	0	0	0	0
NB	1	.8	.4	.1	0	0	0	0	0	0	0	0	0	0

Table 5.1. Fuzzy set definitions for force error (e).

	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
PB	0	0	0	0	0	0	0	0	0	.1	.4	.8	1
PM	0	0	0	0	0	0	0	0	.2	.7	1	.7	.2
PS	0	0	0	0	0	0	0	.9	1	.7	.2	0	0
NO	0	0	0	0	0	.5	1	.5	0	0	0	0	0
NS	0	0	.2	.7	1	.7	.2	0	0	0	0	0	0
NM	.2	.7	1	.7	.2	0	0	0	0	0	0	0	0
NB	1	.8	.4	.1	0	0	0	0	0	0	0	0	0

Table 5.2. Fuzzy set definitions for force error rate (er).

	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
PB	0	0	0	0	0	0	0	0	0	0	0	.1	.5	.8	1
PM	0	0	0	0	0	0	0	0	0	.2	.7	1	.7	.2	0
PS	0	0	0	0	0	0	0	.4	1	.8	.4	.1	0	0	0
NO	0	0	0	0	0	0	.2	1	.2	0	0	0	0	0	0
NS	0	0	0	.1	.4	.8	1	.4	0	0	0	0	0	0	0
NM	0	.2	.7	1	.7	.2	0	0	0	0	0	0	0	0	0
NM	1	.8	.4	.1	0	0	0	0	0	0	0	0	0	0	0

Table 5.3. Fuzzy set definitions for end-effector displacement (u).

Again, to ensure finer control around the equilibrium state, the fuzzy set values listed in table 5.1 have +0 and -0 in their discrete support set.

Step 3. Constructing the rule base: The fuzzy controller is a three dimensional algorithm which infers the controller output (u) based upon the output force error (e) and the output force error rate (er). Since eight fuzzy sets are defined on variable space for e, and 7 fuzzy sets are defined over the variable space for er; 8*7 = 56 rules required. These rules are shown in table 5.4. The inputs (IF) are along the axes of the matrix, and the outputs (THEN) are at the intersections.

ERROR	ERROR RATE						
	NB	NM	NS	ZO	PS	PM	PB
NB	PB	PB	PB	PB	PM	PM	PM
NM	PB	PB	PB	PB	PM	PM	PS
NS	PM	PS	PS	PS	PS	NS	NM
NO	PM	PM	PS	ZO	ZO	NS	NM
PO	PM	PS	ZO	ZO	NS	NM	NM
PS	PM	PS	NS	NS	NS	NS	NM
PM	NS	NM	NM	NB	NB	NB	NB
PB	NM	NM	NM	NB	NB	NB	NB

Table 5.4. Fuzzy logic linguistic algorithm.

Step 4. Fuzzy inference: The compositional rule of inference is used to infer the output from a fuzzy relation composed with non-fuzzy measurements (in this case e_o and er_o). Only those rules that are pertinent to the inputs have any effect upon the resulting fuzzy output set. A pertinent rule in this case is the one for which the error and error rate are defined as having a non-zero grade of membership at the measured value. For example, consider a typical rule which can be written as:

If error is P_e then if error rate is P_{er} then P_u

where (P_e,P_{er},P_u) are fuzzy sets defined on the finite discrete spaces (e,er,u) respectively. Then the fuzzy relation R of this is:

$$R = P_e * P_{er} * P_u \tag{5.1}$$

where * denotes the cartesian product. Next the composition of this rule with the non-fuzzy singleton set (Q_e,Q_{er}) of the measured and quantified error and error rates are formed. The fuzzy output set generated (Q_u) by this rule is given by the composition:

$$Q_u = (Q_e * Q_{er}) \circ R = (Q_e * Q_{er}) \circ (P_e * P_{er} * P_u) \tag{5.2}$$

where o represent composition. The membership function $\mu(.)$, given by

$$\mu_{Q_u}(u) = \max_{e,er} \left\{ \min \left[\mu_{Q_e}(e), \mu_{Q_{er}}(er), \mu_{P_1}(e), \mu_{P_2}, \mu_{P_u}(u) \right] \right\} \qquad (5.3)$$

However since Q_e is a non-fuzzy singleton, its membership value is zero for e except at e_o, where it is 1. This is also true for Q_{er}. Using this knowledge the above equation can be simplified and written as:

$$\begin{aligned} \mu_{Q_u}(u) &= \max_{e,er} \left\{ \min \left[\mu_{P_1}(e_o), \mu_{P_2}(er_o), \mu_{P_u}(u) \right] \right\} \\ &= \min \left[\mu_{P_1}(E_o), \mu_{P_2}(er_o), \mu_{P_u}(u) \right] \end{aligned} \qquad (5.4)$$

The result of this evaluation is a fuzzy set of grades of membership for all possible control actions. In order to produce a deterministic action, one of these values must be chosen. For this application, the control value with the largest grade of membership is chosen.

Step 5. Improving efficiency: The rule-based design is flexible, but not very efficient. To improve efficiency of execution or storage requirements of the controller, the rule may be 'compiled' to a straight forward look up table. The

algorithm for generating the look up table is given below:

read linguistic rule and fuzzy sets

while (any more input combinations) do

{

select the discrete support set (e_c, er_c)

while (any more outputs) do

{

select output support set P_u

while (more rules) do

{

select rule $P_e \rightarrow P_{er} \rightarrow P_u$

calculate the relation given equation 5.4

}

calculate the maximum value of relation

given by equation 5.4 and write the output fuzzy set

}

calculate the mean of the maxima for u_o and write to

the look up table

}

The decision table generated using the above algorithm for the robotic deburring case is shown in table 5.5.

The first column represents the quantization levels for the force error (e), whereas the first row represents the quantization levels pertaining to force error rate (er). The intersection of these two quantized inputs contains the quantized output. When this value is appropriately scaled, it represents the controller output u (end effector displacement). The look up table version of the fuzzy logic controller is shown in figure 5.3.

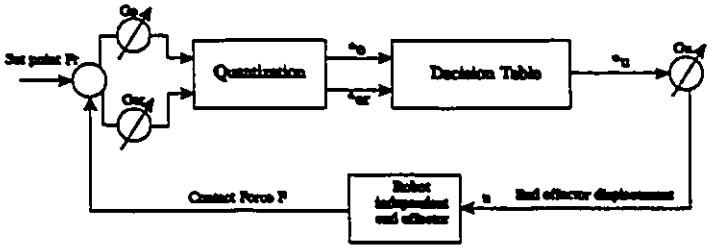


Figure 5.3. The look up table version of the fuzzy logic controller.

	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
-6	7	6	7	6	7	7	7	4	4	4	4	4	4
-5	6	6	6	6	6	6	6	4	4	4	4	4	4
-4	7	6	7	6	7	7	7	4	4	4	4	2	1
-3	6	6	6	6	6	6	6	4	4	4	4	3	2
-2	4	4	1	2	1	1	1	1	1	0	-1	-4	-4
-1	4	4	2	1	2	2	2	2	2	0	-1	-4	-4
-0	4	4	3	2	1	1	0	-1	0	-3	-1	-4	-4
0	4	4	3	2	0	1	0	-1	-1	-3	-1	-4	-4
1	4	4	2	0	-1	-1	-1	-1	-1	-1	-1	-4	-4
2	4	4	1	0	-1	-1	-1	-1	-1	-1	-1	-4	-4
3	-1	-3	-4	-4	-4	-4	-6	-6	-6	-6	-6	-6	-6
4	-1	-2	-4	-4	-4	-4	-7	-7	-7	-6	-7	-6	-7
5	-4	-4	-4	-4	-4	-4	-6	-6	-6	-6	-6	-6	-6
6	-4	-4	-4	-4	-4	-4	-7	-7	-7	-6	-7	-6	-7

Table 5.5. Fuzzy control law modelled as a lookup table.

3.0 Relationship between Fuzzy and PI controller

In a discrete PI controller, the output of the controller is given by:

$$u(k) = u(k-1) + \Delta u(k) \tag{5.5}$$

where Δu represents the incremental change in the controller output. Δu depends on the error e(k) (difference between the output and the set point at k) and Δe(k), where Δe(k) is given by:

$$\Delta e(k) = e(k) - e(k-1) \tag{5.6}$$

Now making the assumption that the control frontiers are linear and quantization levels for the control variables approach zero, we get figure 5.4. The slope of the control policy is given by tan(α).

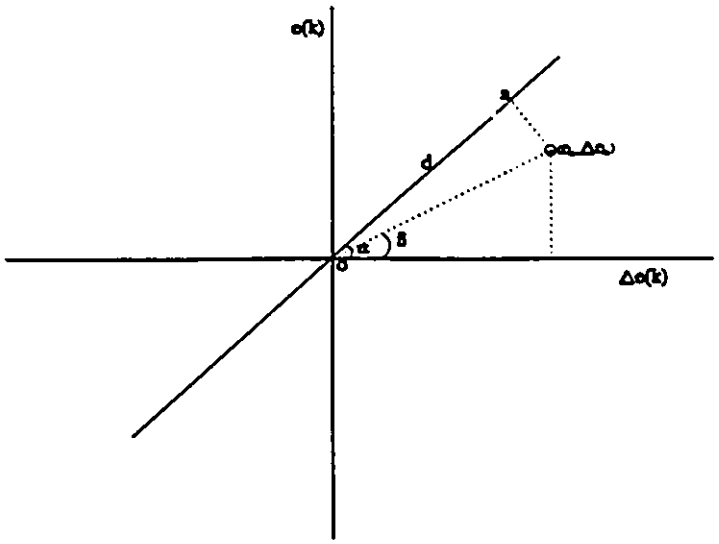


Figure 5.4. The control plane.

For any particular operating point the control output is proportional to d, given by:

$$\Delta u_o = Kd \tag{5.7}$$

Now d is given by:

$$d = \sqrt{e_o^2 + (\Delta e_o)^2}\,\cos(\alpha - \beta) \tag{5.8}$$

Expanding we have:

$$d = \sqrt{e_o^2 + (\Delta e_o)^2}\ (\cos(\alpha)\ \frac{e_o}{\sqrt{e_o^2 + (\Delta e_o)^2}} + \sin(\alpha)\ \frac{\Delta e_o}{\sqrt{e_o^2 + (\Delta e_o)^2}}) \quad (5.9)$$

Simplifying we have:

$$d = e_o\cos(\alpha) + \Delta e_o\sin(\alpha) \quad (5.10)$$

Now combining equation 5.10 and 5.7 we have:

$$\Delta u_o = Ke_o\cos(\alpha) + K\Delta e_o\sin(\alpha) \quad (5.11)$$

Equation 5.11 is simply a PI controller with a proportional gain and integral gain:

$$K_p = K\cos(\alpha) \quad (5.12)$$

$$K_I = K\sin(\alpha) \quad (5.13)$$

Hence it can be seen that the fuzzy logic controller developed for robotic deburring is a form of PI controller.

4.0 Simulation and real time test results

To investigate the characteristics of the fuzzy logic controller developed, some simulations of the robotic deburring control system were carried out. To simulate the FLC the robotic deburring system had to be modelled [BON89b]. The structure of the modelled system is shown in figure 5.5. The process was modelled as a controlled autoregressive integrating moving-average (CARIMA) model of the form:

$$A(z^{-1})F(t) = B(z^{-1})X(t-d) + \frac{a(t)}{\Delta} \tag{5.14}$$

$$\begin{aligned} A(z^{-1}) &= 1 + \sum_{i=1}^{n} a_i z^{-1} \\ B(z^{-1}) &= \sum_{i=0}^{m} b_i z^{-1} \\ \Delta &= 1 - z^{-1} \end{aligned} \tag{5.15}$$

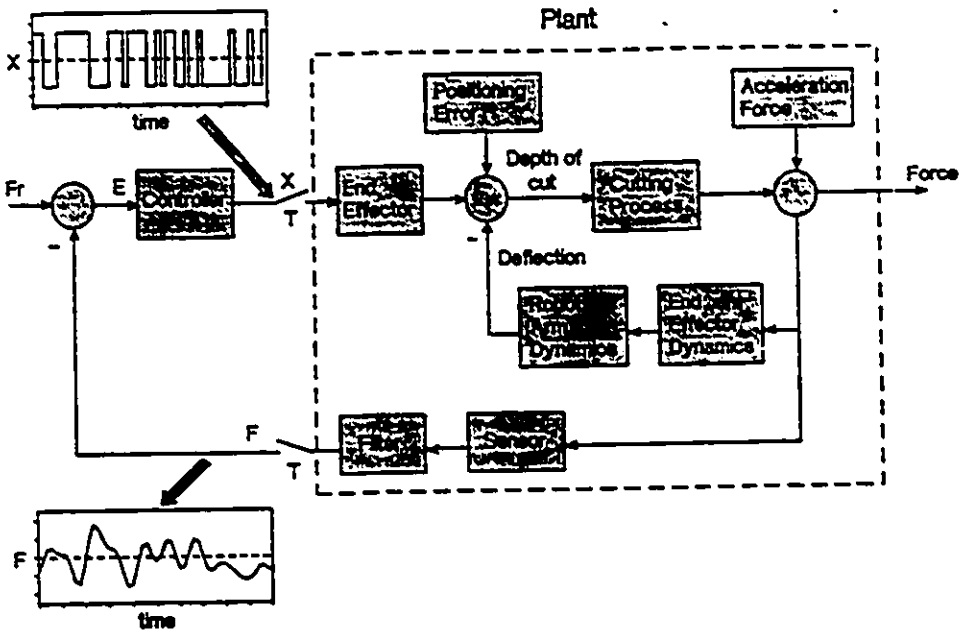


Figure 5.5. Structure of the robotic deburring system.

where: F(t) and X(t) are the measured force and position command at time t, z^{-1} is the backward shift operator, d is the dead time in sampling intervals and a(t) is the white noise sequence. The models were estimated by using maximum likelihood parameter estimation procedure. More details of this procedure for estimating the model of the robotic deburring process is given in [BON89b]. It was found that the process was nonlinear. For different amplitudes of the pseudorandom binary signal (PRBS) different models were obtained. The results of simulation are shown in figure 5.6. The contact force is maintained at 5 Newtons. It is seen from the response, that the contact force is well regulated. The real-time test results are shown in figure 5.7. The performance was as predicted by simulations. A comparative study, which uses Extended Horizon Control (EHC) [BON91] indicates relative advantages for each approach. Given an adequately structured model for the process, the EHC provides tighter control with faster initial transient responses.

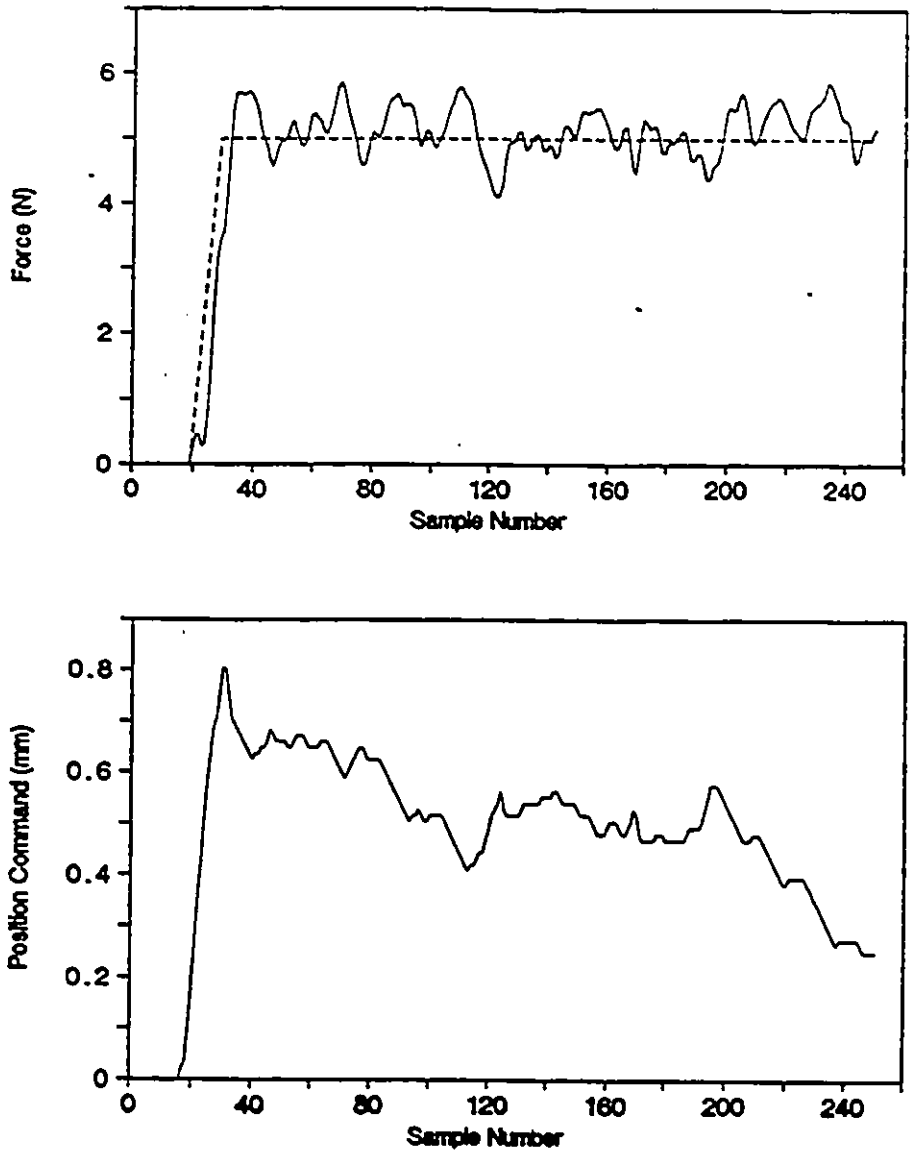


Figure 5.6. Fuzzy logic control simulation result.

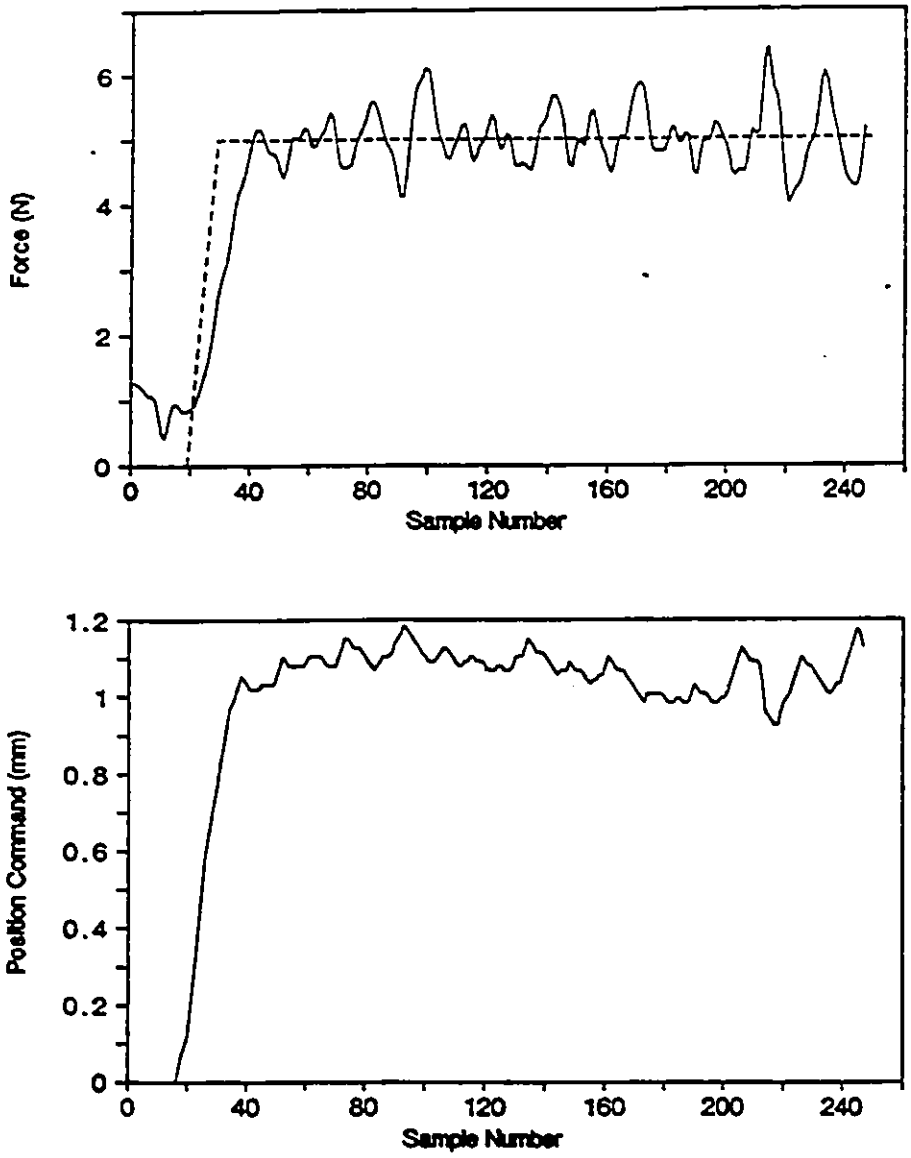


Figure 5.7. Fuzzy logic control real time experimental results.

However, this depends on a reasonably accurate initial estimate of the model. It must also have robust system identification protocol, which can accommodate both random fluctuations and rapid instantaneous disturbances. In contrast, the fuzzy logic controller gives slower initial transient response, although this is significantly improved if a reasonable set of starting rules are used, rather than a empty set. Small amplitude oscillations around the steady state levels are also a common feature.

Chapter 6

Fuzzy Logic controller for obstacle avoidance in an Autonomous Mobile Robot

1.0 Introduction

The term 'autonomous mobile robot' is applied to a vehicle that is capable of executing a pre-defined generalized task (such as patrolling the corridors of a building), without being pre-programmed for specific details of the task. This further implies that the robot should be capable of reacting to unpredictable dynamic events which may impede the successful execution of the task. To achieve this level of "human scale performance", the robot must self-define its operating domain and reliably locate objects within that domain regardless of size, shape, material composition, etc.. It must collect and process data so as to generate navigational ideas, optimal for the specific situation in which it finds itself at every moment. It must perform these rather "abstract" functions efficiently, quickly, economically and without human intervention either prior to or during the execution of the task.

At present, mobile robots being used for factory automation etc., travel along a path which has been laid out. The path taken by such mobile robots are known, and the passageways are usually kept clear. The most popular method used in such systems are marker guided systems. These systems are easy to implement but are also least flexible. In this chapter we discuss the implementation of a prototype fuzzy logic controller for obstacle avoidance by a mobile robot. This controller will allow the robot to move freely on the floor. The chapter is organized is organized as follows. The hardware and the software architecture of the robot is described in section 2. The components required for the fuzzy logic control system are described in section 3. Section 4 describes the fuzzy rules and reasoning used for obstacle avoidance by the mobile robot and the experimental results are given in section 5.

2.0 Description of the Mobile Robot

The mobile robot unit used is a three-wheeled vehicle with two direct drive front wheels (10 inch in diameter) providing vehicle propulsion and steering, while a swivel caster gives the necessary support at the rear. The drive wheels are driven by DC motors. The drive motors are independently controlled through pulse-width modulation (PWM). High resolution optical encoders are placed on the wheel periphery to provide odometric information. The encoders provide feedback

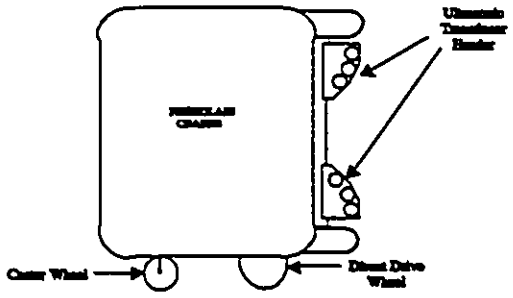


Figure 6.1. Side elevation of the mobile robot.

for every 0.1mm motion of the wheel, thus allowing very precise displacement and velocity information to be used for deadreckoning. The side elevation of the mobile unit is shown in figure 6.1.

An ultrasonic collision avoidance system consisting of two headers is placed in the front. The approximate dimensions of the robot are 85 x 47 x 60 cms and it weighs about 350 pounds. The robot is enclosed in a one piece fibreglass shell. The DC motors, stepper motor and the interface electronics are driven by nine 28 AH gel cell deep cycle batteries. These batteries are placed close to the robot's

centre of rotation to reduce the rotational inertia of the robot in the horizontal plane.

2.1 Collision Avoidance System

The hardware for the collision avoidance system consists of two ultrasonic transducer headers along with two major modules. These modules are made up of the ultrasonics controller with the sonar ranging modules, and the ultrasonics head positioning system. In general, it is better to have more ultrasonic transducers mounted on the robot, but this will increase the amount of circuitry (thereby increasing costs) and put additional time constraints on the onboard computer. Through empirical tests it was found that two transducer headers each having six ultrasonic transducers were sufficient to reliably detect objects in the path of the robot. Each transducer header consist of 2 banks of three ultrasonic transducers placed on each side of the header. This can be seen in figure 6.2.

The transducers in each bank are not positioned in the same manner. The top, middle and the bottom transducer are placed such that their normals are parallel, -25 degrees and -41 degrees to the horizontal as shown in figure 6.3.

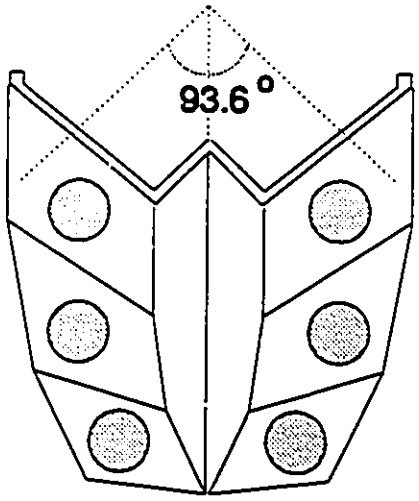


Figure 6.2. Ultrasonic transducer header.

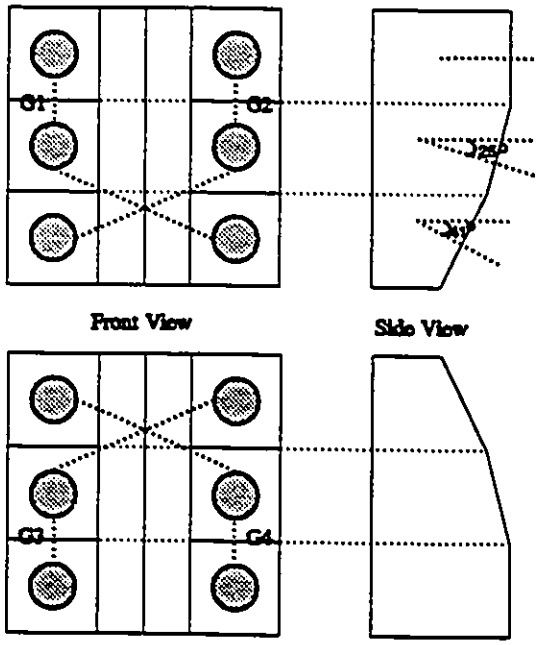


Figure 6.3. Front and side view of the top and bottom ultrasonic transducer header.

The angle between the two banks is 93.6 degrees. This angle was chosen because the stepper motor that controls the header, steps in increments of 7.2 degrees, and it is desirable to have one sonar firing from each bank at the same angle. There is a notch dividing the two banks of transducers, a highly sonar absorbent barrier is placed in the notch to prevent interference between the two banks. This interference occurs when the robot is moving parallel to a wall, the echoes from the transducers in banks firing at the wall are received by the transducers parallel to the wall. When this happens the robot concludes that there is a object in its path, when in reality it is not so. The header itself is made of insulator type material so that each transducer is electrically isolated. This prevents capacitive effects between the transducer and the header, if present could lead to wrong distance readings.

Generally one sonar firing and receiving circuit is required for each transducer. In the mobile robot used, frequency multiplexers are used to reduce the number of sonar firing and receiving circuits from twelve to four. The use of multiplexers increases responses time but this does not affect the performance of the mobile robot because it is typically a slow moving vehicle (1 m/s). All three transducers in each bank of the header do not fire at the same time, this is to prevent an echo being received by the wrong transducer. To minimize interference, the transducers in the top and the bottom transducer headers are grouped into 4 groups G1, G2,

G3, & G4, each containing 3 transducers. They are shown in figure 6.3 by a broken line connecting the transducers in each group. The firing sequence is G1, G3, G2 & G4, each group firing for a duration of 20 milliseconds. Groups G1 and G2 fire at 50 KHz, whereas group G3 and G4 fire at 60 KHz, the frequency multiplexing circuit takes care of distinguishing the echoes from each group.

The head positioning system contains a microstepper motor along with its drive circuit. The driver circuit is a precise motion control circuit used for positioning the ultrasonics transducer header. The essential difference between a conventional microstepper design and this one is that the microstepper applies a sinusoidal exciting current to the stepper coils. This allows for more precise positioning of the ultrasonics transducer header.

2.2 Onboard Computer System

The mobile robot is controlled by two microprocessors arranged in a master/slave configuration as shown in figure 6.4. The different tasks are distributed equally between the master and the slave processor. The slave processor handles ultrasonics control and distance measurement, stepper motor control for head positioning, power supervision and timing generation

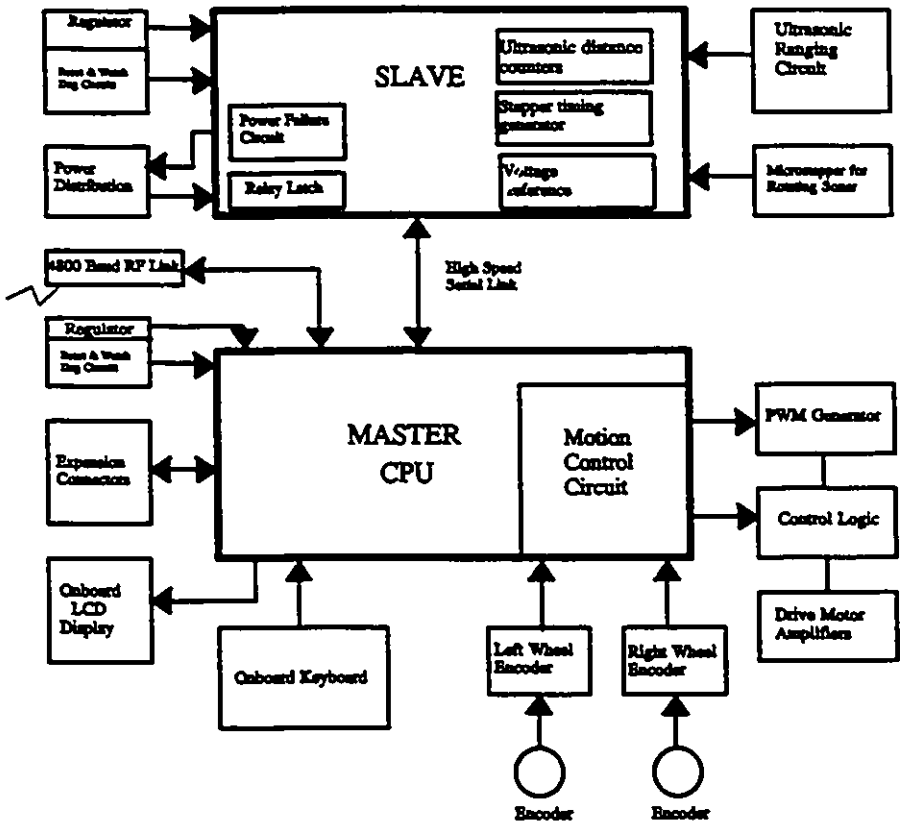


Figure 6.4. Block diagram of the onboard distributed computer system.

The master processor looks after motion control, user interface, expansion, and RS 232 link. The two processors communicate with each other through a high speed serial link. For the simplification of sensor data manipulation as discussed in the introduction and the functionality of the mobile robot, it was found that two microprocessors were enough, but if necessary, additional CPU's may be connected to the master CPU through the expansion connectors provided. The onboard computer communicates with the base station operator or a more powerful high-level planner through a 4800 baud radio frequency (RF) link.

2.3 Software

The mobile robot software consists of two parts, the surveillance navigation software, and the software for motion control and ultrasonic data processing. The navigation software is written in C and then cross compiled to object code. All other software is partly written in C and in assembler language to provide easy manipulation of register data in the CPU and interface chips. In addition, when rapid operation of interrupt routines is required, the interrupt service routines are written in assembler language to ensure that the code has optimum efficiency.

3.0 Fuzzy Logic Based Control System

The general architecture of the FLC system is shown in figure 6.5.

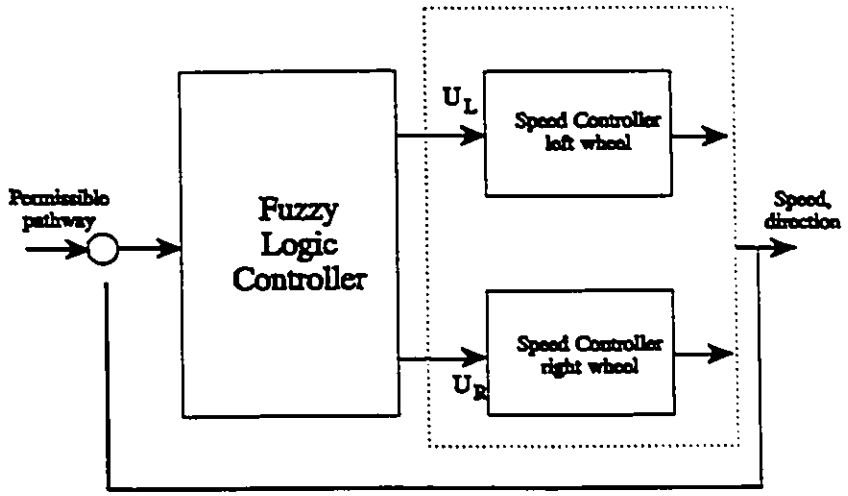


Figure 6.5. Architecture of the FL control system.

It is composed of the sensor system which determines the permissible pathway, the fuzzy logic controller, and two systems to propel the left and right wheel. The details of each system is discussed next.

3.1 Drive servo-control hardware

Closed-loop control of the mobile robot's direct drive wheels are based on the use

of Hewlett Packard HCTL-1000 general purpose motion control integrated circuit [FIS88]. The integrated circuit is a programmable microprocessor with algorithms to support four different modes:

- Position Control
- Proportional velocity control
- Point to point, trapezoidal profile velocity control
- Integral velocity control using linear acceleration profiles

The functions are specified by writing to the registers from a shared bus. The chip is capable of supporting both analog DC and pulse width modulation (PWM) motor torque control. PWM is used because of simplicity of the circuitry and efficiency considerations. Power MOSFET's connected in an "H" bridge configuration are used for bidirectional control from a single polarity power supply consisting of storage batteries.

An incremental optical encoder provides position feedback from each wheel to its HCTL-100 controller. This is shown in figure 6.6.

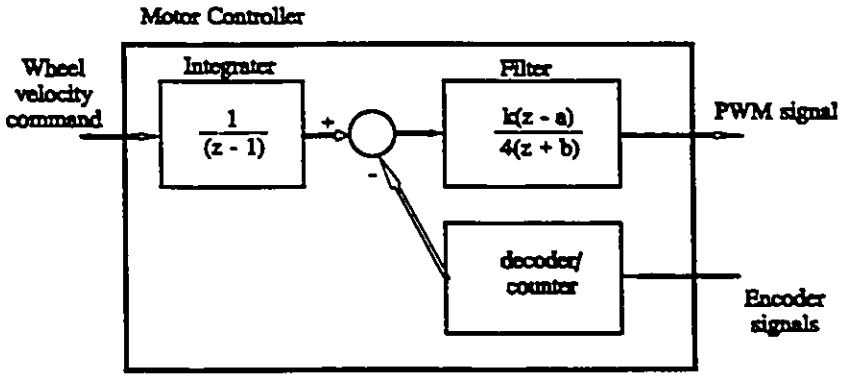


Figure 6.6. Hewlett Packard's HCTL-1000 motor control IC in the velocity mode with integral action.

The speed control system is shown in figure 6.7.

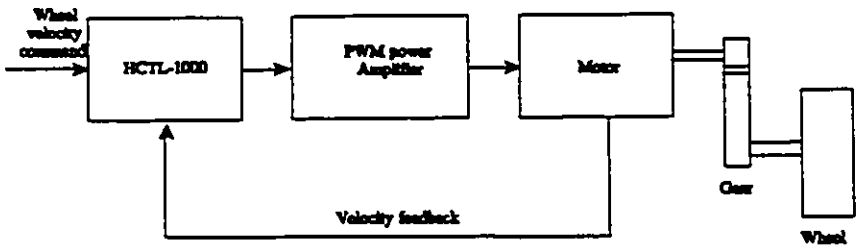


Figure 6.7. Servo-control system on each wheel.

The mobile robot is steered by sending the speed commands U_L and U_R to the left and right wheel speed controllers. In the case of obstacle avoidance, these speed commands are generated by the fuzzy logic controller.

3.2 Odometry

The mobile robot estimates its position via odometry. It is a method by which the position and orientation of the robot is estimated by counting the revolutions of a wheel, this techniques dates back to the time of Archimedes. Consider figure 6.8.

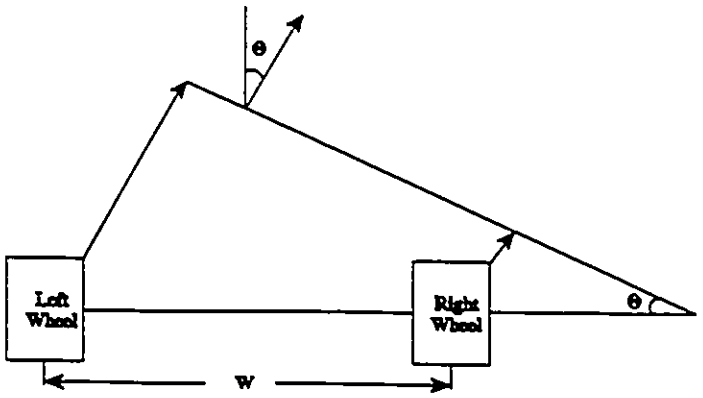


Figure 6.8. Steering of the mobile robot.

Referring to figure, one can easily conclude that the distance moved by the axil midpoint in one sampling interval Δt is

$$\Delta D = 1/2(\Delta D_R + \Delta D_L) \tag{6.1}$$

and the change in the orientation of the wheels, Δθ is given by

$$\Delta\theta = 1/W(\Delta D_R - \Delta D_L) \tag{6.2}$$

where W is the distance between the right and left wheel. Integrating equation 6.1 and 6.2 we get the distance D(t) travelled by the axil midpoint and the orientation

of the wheel θ(t), at time t. This is given by

$$D(t) = 1/2(D_R(t) + D_L(t)) \tag{6.3}$$

and

$$\theta(t) = \theta(0) + 1/W(D_R(t) - D_L(t)) \tag{6.4}$$

The x,y position of the mobile robot must be calculated using numerical integration and is given by

$$x(t) = x(0) + 1/2\int_0^t (\dot{D}_R(t) + \dot{D}_L(t))\cos(\theta(t))dt \tag{6.5}$$

$$y(t) = x(0) + 1/2\int_0^t (\dot{D}_R(t) + \dot{D}_L(t))\sin(\theta(t)))dt \tag{6.6}$$

3.3 Motion equations for a mobile robot

The overall state equation requires the evaluation of longitudinal and lateral

dynamics of the mobile robot. The longitudinal dynamics can be represented by [MIT82] the nonlinear equation of the form:

$$m\dot{v} = f_1(S_e, \omega_e, T_e, D) \tag{6.7}$$

where f_1 represents a complex function that contains gearing, inertial (S_e), engine speed (ω_e), throttle position, drag (D) due to rolling resistance, braking etc. Similarly the lateral dynamics can be given by the following set of equations, that is similar to a two degree of freedom planar bicycle model.

$$mv\dot{\gamma} = f_2(U_{f,r}, \mu, v, C_{f,r}, \theta_v, \varepsilon, \gamma) \tag{6.8}$$

$$I_z \frac{d\theta_v^2}{dt^2} = f_3(U_{f,r}, \mu, v, C_{f,r}, \theta_v, \varepsilon, \gamma) \tag{6.9}$$

where I_z is the vehicle moment of inertia about the vertical direction; $U_{t,r}$ and $C_{t,r}$ are the circumferential wheel and normal axle side forces respectively; μ is the coefficient of friction between the wheels and the floor; ε is the steering angle, θ_v is the vehicle inertial yaw rate; γ is the side slip angle. The steering control dynamics is given by:

$$\dot{\varepsilon} = c_1\varepsilon + c_2 K \tag{6.10}$$

where c_1 and c_2 are constants. Combining the above dynamic equation and the motion equations given in [ALE89], we can come up with a complex composite system whose state vector is (r, v, a, γ, θ_v, ε), with r, v and a being position, velocity and acceleration variables and control inputs being a and K. Pontryagin's maximum principle could be applied for small perturbations and constant forces ($U_{t,v}$) vehicle mass, friction etc. If the objective is minimum time through the control variable (a, K) subject to the environmental constraints:

$$|a| \le a_m \wedge K \le \left(\frac{a_f}{r^2}, K_m\right) \quad where\ a_f \ge Kv^2 \tag{6.11}$$

Then by determining the appropriate Hamiltonian in the system costates (λ), the optimal bang-bang controls can be given as:

$$a^*(t) = a_m(-Sgn\lambda(t)) \tag{6.12}$$

$$K^*(t) = \min\left(\frac{a_f}{v^2}, K_m\right)(Sgn\lambda_v(t)) \qquad (6.13)$$

The implementation of the above equations is not very easy. The equations are highly nonlinear, require computation of the system costates, which in turn requires the use of estimators and numerical optimization techniques, all of which has to be done in real time. The principle of incompatibility [ZAD73] is particularly applicable to obstacle avoidance in a mobile robot in an unstructured environment. A simpler and implementable approach to use fuzzy logic in the manner humans drive a vehicle. This approach is discussed in the sections that follow.

3.4 Detection of Safe Pathway

The robot determines a safe pathway to follow in order to avoid the obstacle by using the ultrasonic transducer header. The obstacle detection system stores a map of the robot and its relative position from the obstacles in front of it in a array of size of 250. The widest pathway is determined from this array by determining the interval x_L, x_R such that all y coordinates of points within this interval is greater than K_L*R_L where K_L is a constant and R_L is the length of the robot. For the widest pathway to be useful it should be greater than K_W*R_W where again K_W is a constant and R_W is the width of the robot. The constants K_L and K_R are

determined via experimentation and they generally lie in the range 1.2 to 2.25. Once the coordinates of the boundaries (x_L,y_L) and (x_R,y_R) are determined, the safe pathway is given by

$$P_{way} = K_x \frac{(x_L + x_R)}{2} \tag{6.14}$$

where K_x is constant dependent on the horizontal resolution of the ultrasonic transducer header. The nearest distance to an obstacle is taken to be the y coordinate of the boundary point that is smaller in magnitude. That is,

$$D = \min(y_L,y_R) \tag{6.15}$$

4.0 Fuzzy Rules and Fuzzy Reasoning

Obstacle avoidance is achieved by steering the mobile robot, this is achieved by varying the reference speed of the left and right wheel speed control systems. This is shown in figure 6.5. The equations for varying the speed of left and right wheel is given by:

$$U_L^k = U_L^{k-1} + K_{Ld} * \delta U + \alpha T \tag{6.16}$$

$$U_R^k = U_L^{k-1} - K_{Rd} * \delta U + \alpha T \tag{6.17}$$

where U_L and U_R are the speed control commands to the left and right wheel at the k^{th} sampling interval. K_{Ld} and K_{Rd} are the directional gain of the left and right wheel, δU the directional rate, α is the acceleration input and T is the sampling interval. Thus one can see that, four sets of fuzzy control rules are required to appropriately adjust K_{Ld}, K_{Rd}, δU and α. Next, we will consider each set of fuzzy rules separately.

4.1 Fuzzy rules and reasoning for direction rate

The directional rate depends on two factors, namely the direction the robot should move and its present rotational velocity. The direction in which the robot should move can be determined from its present position and safe pathway P_{way}, as given by equation 6.14. The present rotational velocity depends on the difference in velocity of the left and right wheel. Let δU be a fuzzy variable along with SD (speed difference) and PD (direction of the safe pathway). They can each be discretized as shown in table 6.1.

The fuzzy control rules relating SD, and PD to δU is shown in table 6.2.

These rules are intended to be read as:

IF path_direction (PD) is NS and Speed_difference (SD) is ZO then the direction-_rate (δU) is NS

ELSE..

The fuzzy variable identifiers used to describe the fuzzy values have the following meanings:

PB Positive Big

PM Positive Medium

PS Positive Small

ZO Zero

NS Negative small

NM Negative medium

NB Negative Big

and are consistent with those reported in most of the applications.

	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
NB	1.0	0.7	0.3	0	0	0	0	0	0	0	0	0	0
NM	0.3	0.7	1.0	0.7	0.3	0	0	0	0	0	0	0	0
NS	0	0	0.3	0.7	1.0	0.7	0.3	0	0	0	0	0	0
ZO	0	0	0	0	0.3	0.7	1	0.7	0.3	0	0	0	0
PS	0	0	0	0	0	0	0.3	0.7	1.0	0.7	0.3	0	0
PM	0	0	0	0	0	0	0	0	0.3	0.7	1.0	0.7	0.3
PB	0	0	0	0	0	0	0	0	0	0	0.3	0.7	1.0

Table 6.1. Fuzzy set definitions for PD, SD, and δU

PD \ SD	NB	NM	NS	ZO	PS	PM	PB
NB	ZO	NS	NS	NM	NM	NB	NB
NM	PS	ZO	NS	NS	NM	NM	NB
NS	PS	PS	ZO	NS	NS	NM	NM
ZO	PM	PS	PS	ZO	NS	NS	NM
PS	PM	PM	PS	PS	ZO	NS	NS
PM	PB	PM	PM	PS	PS	ZO	NS
PB	PB	PB	PM	PM	PS	PS	ZO

Table 6.2. Fuzzy control rules for directional rate.

The output δU is determined by using the centroid defuzzification method. The following C code shows the basic model for the centroid defuzzification method.

```
#include "Fuzzy.h"    // file containing fuzzy set definitions
double centroid(fuzzySet F)
{
        double x, y, j, cPoint;
        for(int i; i < fsMax+1; i++)
        {
                j = (double) i;
                x +=  F.vector[i]*j;
                y += F.vector[i];
        }
        cPoint = x/y;
        return (ScalarOf(F.vector[cPoint]);
}
```

The variable *cPoint* resolves to an index in the membership vector from which the expected value can be selected.

4.2 Fuzzy rules and reasoning for directional gain

The directional gain is used to take into account the distance of the mobile robot from the obstacles. The directional gain depends on the sign of the difference in velocity of the left and right wheels and the distance between the robot and the obstacles (y_L and y_R). If y_L and y_R are discretized as shown in table 6.3, the control rules for K_{Ld} and K_{Rd} are given by table 6.4 and table 6.5.

These rules are intended to be read as:

IF Sgn(Speed_difference) is PO and y_L is PS then the direction_gain (K_{Ld}) is Z0

ELSE ..

	0	1	2	3	4	5	6	7	8
ZO	1.0	0.8	0.6	0.3	0	0	0	0	0
PS	0	0.3	0.6	1.0	0.6	0.3	0	0	0
PM	0	0	0	0.3	0.6	1.0	0.6	0.3	0
PB	0	0	0	0	0	0	0.6	0.8	1.0

Table 6.3. Fuzzy set definitions for y_L, y_R

y_L / sign(SD)	ZO	PS	PM	PB
PO	ZO	ZO	PS	PS
NE	PB	PB	PM	PS

Table 6.4. Fuzzy rules for K_{Ld}

y_R / sign(SD)	ZO	PS	PM	PB
PO	PB	PB	PM	PS
NE	ZO	ZO	PS	PS

Table 6.5. Fuzzy rules for K_{Rd}

The control rules are modified with experimentation. Again, the resultant directional gain is determined by taking the centre of gravity according to the membership functions given in table 6.3.

4.3 Fuzzy rules and reasoning for accleration

The width of the safe pathway is used to change the acceleration gain α. The inputs are average velocity (V_{av}), and the width of the safe pathway (P_w). The output is the new reference velocity (V_{ref}). The same fuzzy set definitions given in table 6.3 is used for P_w and V_{av}. The control relating P_w and V_{av} to V_{ref} is given in table 6.6. These control rules are interpreted in the same manner as before and the resultant velocity V_{ref} is determined by taking the centre of gravity for the membership functions defined in table 6.3. Once V_{ref} is determined, the acceleration gain is given by the formula:

$$\alpha = \rho \frac{(V_{ref} - V_{avg})}{\tau} \qquad (6.18)$$

where ρ is a constant determined by experimentation and τ is the sampling time.

V_{avg} / P_w	ZO	PS	PM	PB
ZO	ZO	ZO	ZO	PS
PS	PS	PS	PS	PM
PM	PM	PM	PM	PB
PB	PM	PM	PB	PB

Table 6.6. Control rules relating V_{avg} and P_w to V_{ref}

5.0 Experimental results

Numerous experiments were carried out for various scenarios. The results for a particular scenario is given in figure 6.9. The obstacles used were a table and a dust bin. The figure shows the trace of mobile robot as it travels from the start position to the destination. The performance of the fuzzy logic controller was acceptable. Occasionally the mobile robot did not perform properly. This was not due to the fuzzy logic controller, but due the failure of the ultrasonic obstacle detection system. The ultrasonic system is cheap but suffers from ray geometry, angular resolution, signal absorption, and response time problems. These problems could be overcome by using a vision system that has real time blob analysis capability in conjunction with a laser range scanner.

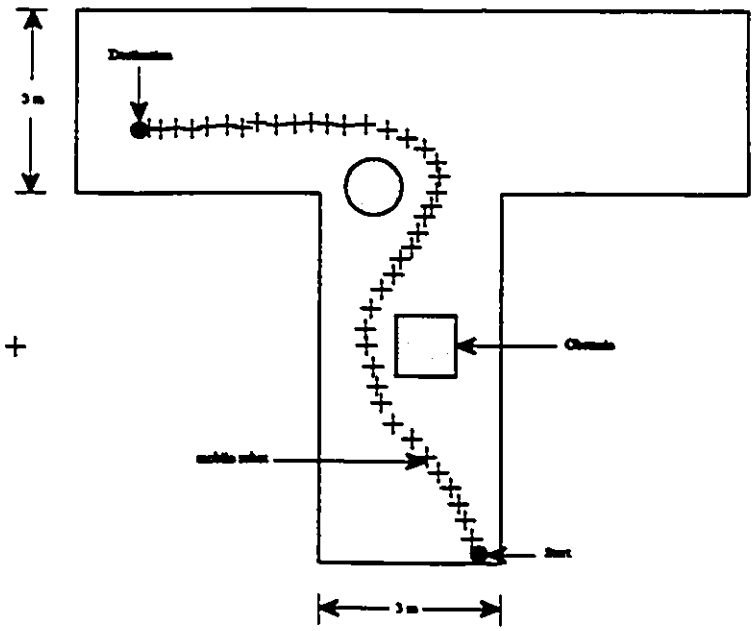


Figure 6.9. Trace of the mobile robot's obstacle avoidance route

Chapter 7

Designing Qualitative Control Systems

1.0 Introduction

Knowledge based controllers such as fuzzy logic [MAM75] and expert control [AST86] are based on the operator's heuristic knowledge of the process. In fuzzy logic control [BON89], the operator's control strategy is mimicked by the computer with the help of qualitative linguistic rules. Expert control, instead extends the range of conventional control algorithms by encoding general knowledge and heuristics about identification and adaptation in a supervisory expert system [ARZ90] [LIN90]. Experience has shown that there exist some nagging problems with the aforementioned knowledge based control techniques. These are mainly due to the manner in which knowledge is represented, each fact or rule stands on its own as an underivable 'axiom' making it impossible to question the validity or basis of it. Model-based reasoning (MBR), instead is based on deep knowledge represented in the form of various kinds of models of the system. To further clarify what is meant by 'model- based reasoning', consider figure 7.1, when contrasted with quantitative

methods, MBR is purely a qualitative approach to problem solving in the domain.

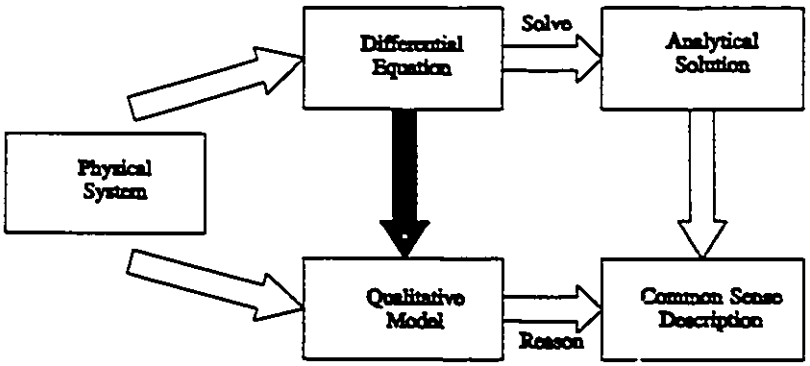


Figure 7.1. Quantitative versus Qualitative approaches.

When compared to conventional approaches to control, such as PID, model-based reasoning controllers have the potential to overcome the difficulties of gain tuning and the effects of nonlinearities.

In literature, model-based reasoning goes under different names which are closely related such as: qualitative reasoning, qualitative simulation, qualitative physics, deep knowledge, naive physics and commonsense reasoning. With respect to knowledge based process control systems, model-based

reasoning can briefly be described as follows: Given a description of the physical system as a set of state variables along with its initial condition, MBR typically involves in describing the temporal behaviour of the state, in terms of a chosen qualitative representation, while keeping track of the reasons for the temporal state change. Since MBR is an important step towards building intelligent controllers of the future, in this chapter, we summarize qualitative process theory, which is one of the many approaches to MBR, and then we use it to design a controller for robotic machining, an important manufacturing hard automation.

2.0 Approaches to Model based reasoning

There are several approaches to MBR that include 1) constraint centred approach [KUI84] [KUI86], where the whole system is described as a homogeneous set of constraints, 2) component centred qualitative simulation [KLE84] [WIL84], here a system is modelled by instantiating components from the library and connecting them explicitly, and 3) qualitative process theory [FOR84], this builds upon the component centred approach by modelling not only individual components but also the processes that act upon them.

An ideal model based reasoning should have the following features:

- The modelling language should be capable of describing complex physical systems to varying levels of detail and granularity. Consequently, the grain of the output of the reasoner may vary as the input grain changes.

- Processes and interactions must be describable and moreover the reasoning systems must be capable of instantiating such descriptions and, what is much more difficult, generating new descriptions to describe the dynamic behaviour of an entire system from the descriptions of the parts.

- The system should provide a unified structure for supporting a variety of different tasks such as design, diagnosis, planning and prediction.

- In view of the requirements to build descriptions of, and reason about, complex systems, it must be possible to model individual parts of a system separately, and have the system reason about the behaviour of the complete system. Thus the modelling language should support descriptions which are composable. Support should also be provided to help the user of the system to model their problem using the primitives. Such help might be provided partly by making the language sufficiently expressive, partly by providing a library of primitive description for processes, components, and their cause.

- Finally, one would like to be able to integrate quantitative knowledge and inference into a qualitative reasoner when available and link a quantitative reasoner to a traditional system.

No existing theory entirely meets the aforementioned requirements. The earliest piece of work in this area is de Kleer's [KLE77] NEWTON system. NEWTON is a system which attempts to solve simple mechanics problems qualitatively, and only uses quantitative methods if qualitative reasoning fails to provide a solution; however, even in this case, the results of the qualitative reasoner may guide the selection of equations and the way they are solved quantitatively. NEWTON introduced the important idea of *envisioning* which is a qualitative state transition graph, and captures the possible qualitative behaviour sequences. However, NEWTON was very limited in its representation and inference techniques, and seemed very tied to the simple mechanics domain in which it operated. Moreover, it had virtually no notion of time. The other piece of classical work referenced by most papers in this area is the work by Hayes [HAY79] [HAY85a] [HAY85b] on *Naive Physics*. Hayes' work mainly concentrated on axiomatizing our commonsense knowledge of the physical world within a logical framework, i.e. he was more concerned about modelling rather than reasoning aspects. The idea of *qualitative quantity space* was first suggested by Hayes. His first paper, the *Naive Physics Manifesto* [HAY79] and the revised version [HAY85a] give the

motivation for trying to build Naive Physics axiomatizations, the advantages of doing it in first order logic, and discuss the methodology and some particular problem areas. His first application was in the world of liquids [HAY85b], which he argued is particularly difficult to formalize. The reason is that liquids are denoted by mass terms, and the semantics of mass terms has been the subject of much philosophical debate, and moreover liquids can divide, combine, deform with incredible ease when compared to solids. One important concept introduced by Hayes in his papers is the idea of *history*, which is a piece of 4-D space time describing the 'life' of a particular object. He argues that histories can be of great help in solving the frame problem [McC69].

Although these papers have been very influential in the field of qualitative reasoning as a whole, Naive Physics itself remains relatively unexplored. Other attempts include Hobbs *et al.* [HOB85] and Cunnigham [CUN85]. Hayes' investigation of the liquid world is only an initial investigation, and there are many difficulties and problems with the axiomatization. Some of these problems are discussed in [WEL84] [HAY87]. McDermott [McD87] believes that the whole enterprise is impossible at least within the confines of formal logic.

Of the three approaches to qualitative reasoning mentioned earlier, Qualitative Process Theory (QPT) was found to be more suited for application in the process

control domain. A brief overview of QPT is given next and its application to a simple robotic machining system is discussed. Limitations and enhancements to this approach are discussed in the final section.

3.0 Qualitative Process Theory

This approach was first proposed by Forbus [FOR84]. The main idea behind QPT is that a physical system is made of many active processes, and all changes in the system occur due to interaction between these active processes. To predict and reason about these changes, all active processes are explicitly represented, which makes QPT a process centred approach.

In QPT, there are no time instants. They are represented as intervals with no subintervals. For example the predicate *robot(c_force, pass)* means that the contact force exists when the end effector executes a 'pass'. The characterization of time in this manner is based on Allen's [ALL83] interval calculus. QPT represents quantities, which are continuous functions of time by four components: magnitude, sign of the quantity (notated s[]), magnitude of the derivative of the quantity, and its sign. A sign is one of -1, 0 or 1, whereas magnitude is defined by a quantity space. As an aside, there is no fixed manner in which magnitude can be represented, in designing the controller for robotic machining, a approach similar to [MUR88] is taken. A quantity space with multiple resolution is used,

when ambiguities are encountered in one quantity space a switch to a higher resolution quantity space is made.

An object in QPT is represented by a *view*, which is a modelling primitive satisfying a particular behaviour. For example, in the case of robotic machining, different *views* are instantiated for different cutting conditions. A *view* is made of four parts: *Individuals, Quantity conditions, Preconditions*, and *Relations*. *Individuals* are the objects that should exist for a particular *view* to exist. *Quantity conditions* and *Preconditions* are a set of conditions that should hold for a *view* to exist. The difference between the two is that *Preconditions* contain the list of conditions that may be affected by the system, but Quantity conditions remain undisturbed by any *view*. Finally, *Relations* are the things that hold true when a particular *view* exist. In [FOR84], Forbus represents *views* by a frame like notation. The application in this paper is simulated in the C++ computer language, *views* are implemented as C++ objects. Certain C++ language features such as *inheritance, polymorphism* etc. have been exploited in connecting and instantiating *views*.

It is necessary to have some mechanism, to represent the *relation* field in each *view*. The mechanism should be such, that the functional relationship between the quantities should hold, so as to satisfy the *conditions* specified in that particular

view. To handle this, Forbus introduced *qualitative proportionalities*. For example, using the idea of *qualitative proportionalities* one may write *contact_force* α_{Q+} *depth_of_cut*, to signify that the contact force is qualitatively proportional to the depth of cut, and it is monotonically increasing in its dependence.

Process are represented in a manner similar to the way a *view* is represented, except there is an addition of a part that describes the influences on any quantity within the model. For example, in the robotic machining case, burrs have a negative influence on the surface finish, this can be represented by *I(surface_finish,burrs)(-)*. The influences represented this way are direct influences, influences that are indirect occur through propagation, and are handled by *qualitative proportionalities*. The derivatives of quantities are determined from the influences on them, this in turn affects the temporal behaviour of the quantity, which is determined through a integration procedure called *limit analysis*.

It may so happen that some phenomena in the system may not be easily represented by processes. Forbus [FOR84] introduced *encapsulated histories* to handle this condition. They are very similar to *views*, except that the *relation* field holds portions of histories for the individuals involved.

3.1 Deduction in QPT

Measurement interpretation and envisioning in QPT is handled by a combination of four basic deductions: *elaboration, process and view structure determination, influence resolution*, and *limit analysis. Elaboration* is the first step in the deduction process, in which all *processes* and *views* are instantiated. This gives a description of a particular domain. For example, in this paper, it gives a particular description of the robotic machining domain. *Process and view structure determination* is the next step in the deduction. It involves selecting from all the *processes* and *views*, a subset, which is consistent with the known facts about the physical system. Once the process structure is given, it is necessary that the influences on the different situation parameters should be resolved, this is called *influence resolution*. This is also required to determine the derivative of all the quantities. Finally, *limit analysis* is the deduction in which predictions are made about each possible process and view structure. *Process* and *view structure* change occur due to quantities crossing their thresholds. Basically, *limit analysis* combines information from step 2 and 3 in the deduction process, i.e. the quantity space and the derivative of each parameter, to determine which *processes* and *views* are active or inactive.

4.0 Application to robotic machining

The objective of the control system is to achieve a desired depth of cut by

maintaining a constant cutting force. To achieve higher control bandwidth and accuracy than is possible using the robots own controller, a system where the depth of cut is controlled by an active end effector has been developed [BONS8]. The end effector is controlled independently of the robot controller and allows small corrections to be made to the robot's path. A block diagram of the control system is shown in figure 7.2. The figure also shows all the active processes influencing the cutting process.

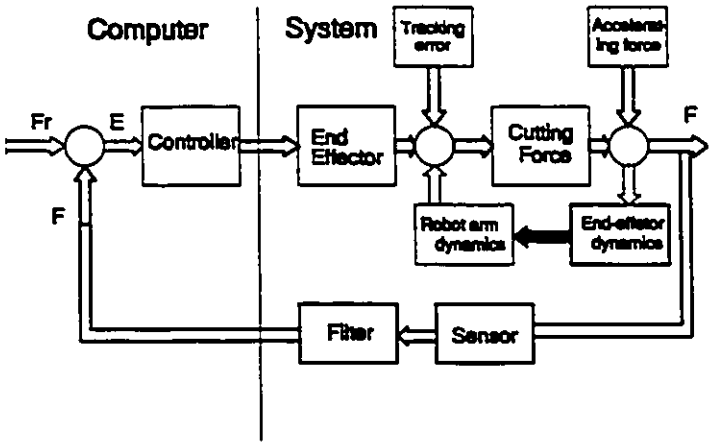


Figure 7.2. Block diagram of the robotic machining system.

In order to control the machining process, the QPT control system must perform several tasks. It must gather data and interpret data, and convert it to symbolic form. It must then perform the QPT deduction process by considering the goals

of the process, make control decisions, and transmit these control commands to the end effector. Because the QPT controller is computationally intensive, special purpose hardware is required to implement in real-time. To investigate the characteristics of the QPT controller developed in this paper, some computer simulations of the robotic machining control system were carried out. The robotic machining system was modelled as a single input single output (SISO) system. The transfer function between the cutting force and the end effector displacement was obtained by measuring the change in the cutting force while the displacement was varied. A pseudo-random binary signal (PRBS) signal with a 20ms sampling interval was used. A controlled autoregressive integrating moving-average (CARIMA) model was identified using maximum-likelihood parameter estimation method [BON89]. In brief, the qualitative controller works as follows: the qualitative error is determined by comparing the force output and the set point force, the rate of change of the error is found by reference to history. The qualitative deduction process uses this knowledge to deduce the required state of the variables that would achieve a zero error state. Finally, the qualitative change in the state variable representing end-effector is converted to a quantitative form, this is then input to the robotic machining system.

The results of the simulation are shown in figure 7.3.

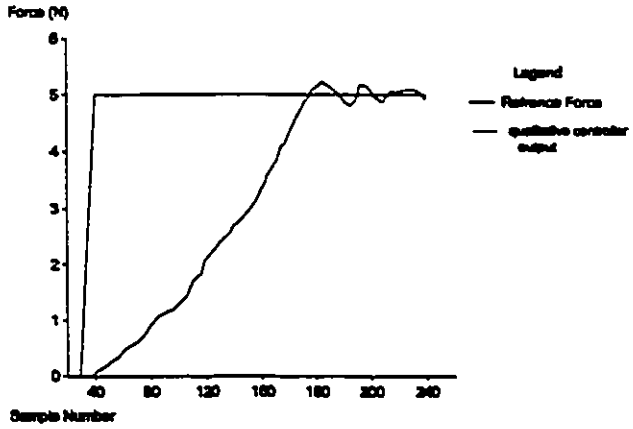


Figure 7.3. Output from the unmodified qualitative controller.

One can conclude from figure 7.3 that the unmodified qualitative controller does not provide a satisfactory response. The transient and steady state response of the system can be improved by augmenting the qualitative controller with domain specific quantitative knowledge. This quantitative knowledge includes the history of events required to evaluate the qualitative values of the derivative of the contact force, and the practical rates of these changes. This idea is similar that discussed in [ABD89]. The result of the simulation for this case is shown in figure 7.4. One can see that there are improvements in the steady state and the transient response of the system.

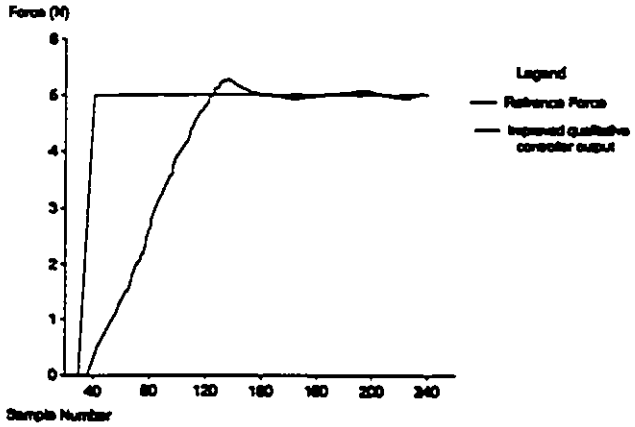


Figure 7.4. Improved qualitative controller response.

5.0 Limitations and Enhancements

In this chapter, we have used qualitative process theory, to design a model-based reasoning controller for robotic machining. It is seen that the performance of an unmodified MBR controller is not satisfactory. This is mainly due to the generality of the qualitative model represented in the knowledge base. An improvement in the system response occurs when quantitative knowledge is combined with the qualitative process model. The performance of the MBR controller is comparable to the extended PID controller [BON89]. Further research needs to be performed in order to speed up qualitative deduction, so as to allow real-time implementation. Some research has been done in this direction,

this is based on Assumption-based Truth Maintenance System of de Kleer [KLE86]. The adaptation of this approach to process control systems needs further exploration.

Chapter 8

Model-Based Diagnosis of Control Systems

1.0 Introduction

This chapter discusses a scheme to implement a model-based system for diagnosis of process control systems (PCS). Diagnosis of PCS involves generating possible causes of anomaly in the system in real time. These possible causes highlight the abnormal sensor values. To come up with these possible causes, the knowledge based system should be composed of appropriate components, including process analyzer, anomaly recognizer, etc.. Economic incentives for real time diagnosis include product quality, equipment protection, and environmental protection. The time frame for diagnosis can vary to a large degree depending on the system being diagnosed. For many biological and chemical processes the time frame may be several minutes to hours, whereas for electronic systems such as power systems, the time frame for diagnosis may be less than a second

Diagnosis of PCS using an model-based system requires the addressing of the following main issues.

1. Communication and coordination between the different components of the

diagnosis system should be handled efficiently, i.e there should be no latency in the diagnosis process if data is not available. The best answer should be generated in the allowed time.

2. The pruning of the search space that can be potentially large should be handled effectively.

There are other issues. These issues are discussed in detail in [LAF88]. The approach taken for diagnosis includes using a 'blackboard', the idea behind a blackboard system is briefly reviewed next.

2.0 Blackboard Model

The solution to diagnosis is appropriately handled by means of *blackboard* [NII86a] [NII86b]. The name 'blackboard' stems from the problem-solving analogy of a group of specialists gathered around a blackboard to collectively solve a problem, where the writing of an idea or fact on the blackboard by one specialist may act as a trigger for another expert to contribute to the next part of the solution. This is tantamount to one specialist's decision about a subproblem stimulating another specialist to solve another part of the problem. A blackboard model has several characteristics which are beneficial to the application considered in this chapter. These characteristics include:

- Problem may be easily broken down into several modules, as is the case for a diagnosis system,

- strategies that are data and goal driven can be easily employed,

- the blackboard is conducive to incremental development of the system,

- each problem solving module can be autonomous, thereby allowing each module to use methods suitable for their particular task, and

- allows limited form of non-monotonic reasoning (decisions can be retracted in light of new evidence).

A detailed review of the blackboard architecture, and a description of several applications may be found in [ENG88].

2.1 Structure of the Blackboard System

The generic architecture of a blackboard based system is shown in figure 8.1. It consists of three modules: the sensor, the model-based blackboard system, and the operator interface. The sensor sends messages containing the parameters of intercepted signals to the blackboard system. The operator interface is used to communicate results to the operator. The operator may also interrogate the system

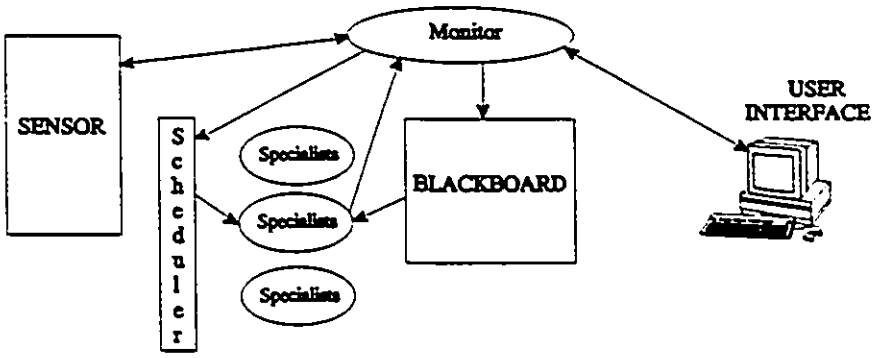


Figure 8.1. Structure of the knowledge based blackboard system.

about the current situation, or provide input to the system either on his own initiative or when asked by the system.

The model-based blackboard system consists of the blackboard, specialists, monitor and the scheduler. The blackboard is a global data structure that contains, at any given time, all known facts and hypotheses made by the system. Specialists represent domain knowledge that has been packaged into problem-solving units. They use information on the blackboard to calculate or deduce new information and hypotheses. The monitor is a mechanism of control by which specialists are invoked to respond to changes in the environment and to the deductions of other specialists. The monitor watches for changes to the blackboard, and modifies the agenda of the scheduler. The objects that do the watching are called daemons. The scheduler allocates time and resources to the different specialist tasks.

3.0 Diagnosis

There are many disadvantages in using an expert system (ES) for diagnosis. A typical industrial ES must interact with a mass of sensor data, which may not be entirely consistent. In case of failure, the ES uses the rules in the knowledge-base to determine the association between the set of sensor data and a fault. This method is generally called associational diagnosis. The earliest application of this method was in the medical diagnostic system called MYCIN [SHO76]. The

knowledge base of MYCIN consisted of explicit mappings from symptoms and intermediate causes to causes, or intermediate causes. The problem with this approach is that the system cannot deal with situations not envisioned by the designer, under these circumstances the system provides an incorrect answer rather than indicating that the answer is unobtainable. Another problem with such systems is maintaining a consistent knowledge-base. In an industrial diagnostic system, if we suppose that table is generated, listing all the possible faults, and the observed sensor states, then rules can be written that verify sensor data and compensate for errant sensor data. This system would become obsolete quickly in the real world due to system design changes; one new sensor in a critical location could require reworking most existing rules. In a dynamic environment, an ES based diagnosis system must constantly struggle to stay abreast of modifications and changes in machine configuration.

These problems have led to other approaches. One approach is to build in faulty behaviors into component descriptions, as is done in medical expert system KARDIO [BRA86] [BRA89]. This is much easier than specifying the faults globally for the entire system. The process of envisioning, will generate a list of all the ways in which faults or combination of faults affect the global system behavior. The disadvantage with this approach is that it will not handle assembly or design errors, nor faults that are not explicitly foreseen in the component description. In practice another problem arises when using this approach, namely,

the combinatorial explosion. This results from considering all the possible interactions of faulty and correct behaviors of individual components. Other alternative approaches have been proposed which do not use explicit component fault models. These are called consistency-based diagnosis. The knowledge base in consistency based diagnosis, describes the normal functioning of the domain, and diagnosis involves in identifying the minimal set of components that result in abnormal behavior of the system. Recently this approach has been investigated by de Kleer and Williams [KLE87] and, independently by Reiter [REI87a], with particular attention to the problem of diagnosis in the presence of multiple faults. Reiter's approach is more rigorous whilst deKleer and Williams have implemented a system based on the ATMS (Assumption-based Truth Maintenance System) [KLE86a] [KLE86b] [KLE86c] [REI87b]. There are certain disadvantages in using consistency based diagnosis, and they are discussed in detail in [REI87a].

3.1 Knowledge based systems for diagnosis

Real time diagnosis requires the knowledge to be comprised of different types of knowledge, such as:

1. Facts,
2. associations,
3. conditionals,
4. procedures, and

5. equations.

Object-oriented systems [BOO91] define classes of objects to represent facts or behaviors. Each class serves as a template for organization of data by defining the number and type of attributes that distinguish one type of object from another. Associations among facts are maintained by the structure of the object system, typically hierarchical, and by relations between objects. Rule based systems provide convenient mechanism for representing conditional knowledge. An inference, searches and executes the pertinent rules. The separation of rule base and the inference makes knowledge representation declarative. Procedural knowledge, functions and equations can be written in the underlying language in which the inference engine is written.

4.0 Architecture of a Diagnosis System

A hypothesis in the system include: the fault type, the cause or source of the problem, the propagation path and the system status. The fault type is either single fault or multiple independent faults. The source of the fault is the physical component that is defective. The propagation path describes the order in which the fault affected the other components. The system status describes the components affected by the fault and their operational status.

The input to the diagnostic system is a set of qualitative values which identifies

which sensors are abnormal and how they are abnormal, e.g. temperature is high. A fault monitor generates these symptoms by comparing the sensor readings to expected values computed from a numerical/qualitative simulation model. Schutte and Abbot [SCH86] describes such a kind of fault monitor in detail.

Diagnosis of known faults can be determined by using compiled associational knowledge. This knowledge basically consists of fault-symptoms associations. These fault-symptoms associations are determined from domain experts and by examining actual fault cases. These fault-symptoms can easily be captured in a rule based system that allows temporal functions defined by Allen [ALL83b] as part of the rules. These rules can adequately capture the sequence of symptoms as described by experts, but have representational limitations as discussed in [ABB87]. These limitations include the awkwardness of expressing all propagational behavior over time that could occur for any fault, as well as the temporal duration. The major question addressed in the rest of the chapter is, what to do if the fault is one, whose symptoms do not correspond to the associational knowledge. This requires reasoning with deep models. Deep models are based on the theories, facts, and principles of a specific domain. The qualitative reasoning approach used here is Kuipers' constraint centered approach [KUI84] [KUI86]. This approach is also called Qualitative Simulation or QSIM, and is reviewed next.

4.1 Qualitative Simulation

In QSIM, a physical situation is modelled by a single set of constraint equations which are essentially, qualitative differential equations. Thus, a model consists of a number of parameters, or variables, and some specified relationships between them. One important concept introduced by QSIM is *the qualitative variables*, i.e. a variable which can take one of a finite number of qualitative values, and which is a function of time. The representation of *qualitative variable*, constraint equations, situation, and the QSIM algorithm are discussed next.

Representation of qualitative variables: Landmarks are distinguished points on the real line. A quantity space is a finite set of totally ordered Landmarks. At any time t a function f is described by a pair $<qval, qdir>$. The value *qval* may be a landmark l or two adjacent landmarks (l1,l2), if f is between landmarks at t. The value *qdir* is one of {*dec, std, inc*}, standing for decreasing, steady or increasing and is qualitative derivative of f.

Constraint Equations: Three types of functional primitives are given by Kuipers, they are:

Arithmetic: plus(x,y,z), multiply(x,y,z) and minus(x,y)

Functional: Mplus(x,y), Mminus(x,y)

Derivative: derivative(x,y)

plus(x,y,z) represents z to be sum of x and y and similarly for multiply(x,y,z),

minus(x,y) constraints x to be -y. Mplus(x,y) states that y is some function of x which is strictly monotonically increasing. Similarly Mminus(x,y) states that y is some function of x which is strictly monotonically decreasing. The derivative primitive derivative(x,y) predicates that y is the derivative of x with respect to time.

The above constraint primitives can be used to model ordinary differential equations (ODE). Take for example the ODE:

$$\frac{d^2u}{dt^2} + \frac{du}{dt} = \cot(ku) \tag{8.1}$$

We can rewrite this as the set of simultaneous equations given on the left and the corresponding qualitative constraints on the right as:

$$\begin{aligned}
f_1 &= \frac{du}{dt} && derivative(u,f_1)\\
f_2 &= \frac{df_1}{dt} && derivative(f_1,f_2)\\
f_3 &= ku && multiply(k,u,f_3)\\
f_4 &= \cot(f_3) && Mminus(f_3,f_4)\\
f_2 + f_1 &= f_4 && plus(f_2,f_1,f_4)
\end{aligned} \tag{8.2}$$

The set of qualitative constraints so derived will always be equivalent to, or less

restrictive than the original ODE, so any solution to the ODE will also satisfy the qualitative constraints.

Situation description: A physical situation can be described by:

1. A set of parameters, associated with each is the quantity space, and the range limits which define the operating range of the constraints.
2. a set of constraint equations, and
3. a set of qualitative values for each of the parameters at the specified time point.

Qualitative state transitions: One of the key components of QSIM are the state transition rules. These state transition rules are subdivided in P-transitions and I-transitions. I-transitions represent transition from time interval to point, and P-transitions represent transition from point to an time interval.

P-Transitions

P1	$<l_j$, std>	→	$<l_j$,std>
P2	$<l_j$, std>	→	$<(l_j,l_{j+1})$,std>
P3	$<l_j$, std>	→	$<(l_{j-1},l_j)$,dec>
P4	$<l_j$, inc>	→	$<(l_j,l_{j+1})$,inc>
P5	$<l_j$, std>	→	$<l_j$,std>
P6	$<l_j$, dec>	→	$<(l_{j-1},l_j)$,std>

P7 $<(l_j,l_{j+1}), std> \rightarrow <(l_j,l_{j+1}),std>$

I-Transitions

I1 $<l_j,std> \rightarrow <l_j,std>$

I2 $<(l_j,l_{j+1}),std> \rightarrow <l_{j+1},std>$

I3 $<(l_j,l_{j+1}),inc> \rightarrow <l_{j+1},inc>$

I4 $<(l_j,l_{j+1}),inc> \rightarrow <(l_j,l_{j+1}),inc>$

I5 $<(l_j,l_{j+1}),dec> \rightarrow <l_{j+1},std>$

I6 $<(l_j,l_{j+1}),dec> \rightarrow <l_j,dec>$

I7 $<(l_j,l_{j+1}),dec> \rightarrow <(l_j,l_{j+1}),dec>$

I8 $<(l_j,l_{j+1}),inc) \rightarrow <l^*,std>$

I9 $<(l_j,l_{j+1}),dec> \rightarrow <l^*,std>$

Transitions P4, P5, P6 and P7 uniquely determine the next value of a variable which matches their left hand side; if a function is increasing or decreasing at a landmark then in an instant it will move to a adjoining open interval in the direction of change (P4 and P6) and a function cannot reach the endpoint of an interval instantaneously (rules P5 and P7). The rationale for using the aforementioned rules is given in more detail in [KUI86].

QSIM: The algorithm consists of selecting an active state, generating possible successor states, filtering these and repeating. If more than one successor state

results then a branching tree of states is generated. Initially the set of active states consists of a single specified state.

To generate the possible successor states, individual function transition rules from P1 to P7 or I1 to I9 are selected. Potential rules that violate invariant transitions are discarded. Then, for each constraint, two or three tuples of value pairs are built, using the previously selected transition rules, according to the strictures of the constraints. The individual constraints checks are of two kinds: the directions of change must be consistent and corresponding values must be adhered to. For example *Mplus* requires both function to have the same direction of change, *plus* requires that if the first two arguments are increasing then so is the third, *derivative(f,g)* requires the sign of *g* to agree with the *qdir* of *f* and *Mplus* requires both functions to reach specified corresponding values at the same time instant.

Consistency between pair of constraints is checked using a Waltz filtering algorithm [WAL75]. Complete state descriptions are now generated from the filtered tuples and these new states are made children states of current state. Finally, global filters are applied to these states and any which pass are added to the set of active states. These global filters include:

1. No change -- is the present state identical to the parent state?
2. Cycle -- is the state identical to some previous state?

3. Quiescence -- is the state time instant and all *qdirs* are *std.*

4.2 The functional components of the Model-based Diagnosis system

The input to the diagnosis system consists of set of significant events, and the output is a list of faulty components or subsystems. The model based diagnosis system is based on a causal blackboard multi-process model. It consists of a shared blackboard, scheduler and the following specialist that execute in parallel:

1. Trigger generator,
2. Event checker,
3. Cause of event generator,
4. Consequence of event generator, and
5. Search space manager.

The structure is shown in figure 8.2. The *trigger generator* is a process that attaches triggers to appropriate events on the blackboard. The *event checker* checks the constraints of the different events against actual plant data. Events that pass the checks are called confirmed events. The *cause of event generator* identifies the potential causes of an event, and the *consequence of event generator* determines what events could follow from a given event. The *search space manager* instructs the scheduler to favor more likely hypotheses and prunes unlikely parts of the search space. Several criteria are used for determining the

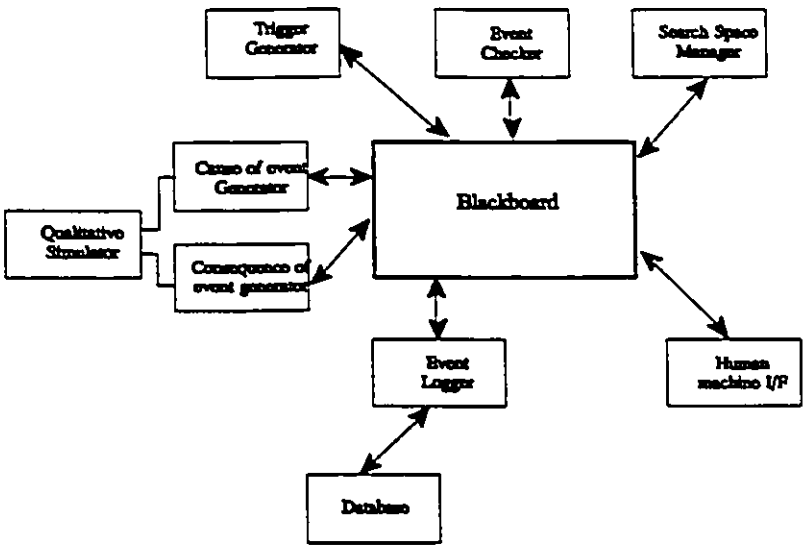


Figure 8.2. Functional block diagram of the diagnosis system.

likelihood of a hypothesis. They include the number of verified events, a priori likelihoods, etc..

The *scheduler* allocates resources, and time, to the various diagnosis components. Several domain dependent heuristic may be incorporated in the scheduler to make it more efficient.

The *blackboard* itself can be implemented in two ways. These are:

1. Each component is run as an independent process, and running possibly on its own processor that write their conclusions on the blackboard managed as an independent process.

2. Each component can be threads within the inference engine sharing a common address space.

Both approaches are viable alternatives, having its own advantages. The first approach is more elegant from a software engineering point of view, whereas in the second approach there is less overhead in sharing knowledge between the different components. How diagnosis is carried by the blackboard architecture described above is discussed next.

4.3 Diagnosis Model

Let us imagine that the blackboard is empty when a trigger t is received. Each trigger is associated with a set of events that describe the possible cause of the trigger. For each event the set of hypotheses are associated with it. Once a trigger exists on the blackboard, various components of the blackboard system run in parallel. The *event checker* runs to confirm each hypothesis for each event. The *cause of event generator* runs to explain the event in each hypothesis originating from some problem with the system. Each event has a set of possible consequences. The *consequence of event generator* run to determine the set of possible consequences for each event. Kuipers' qualitative simulator is used by the *cause of event generator* and the *consequence of event generator*. The *search space manager* instructs the scheduler to favor more likely hypothesis and prunes unlikely parts of the search space. The causal event trigger structure stored in the blackboard is then used for diagnosis and presentation of diagnosis to the user.

5.0 Final Analysis

In this chapter, a strategy to implement a real time process diagnostic system is proposed. Such a system can result in a number of benefits, including: reduced manning levels, reduced need for a highly skilled operator always being present, reduced training costs, increased safety, improved quality, less down time, and a more consistent, higher quality monitoring.

The strategy proposed in this thesis has not been implemented for a real system. The proposed method may require modification depending on the application domain. It may also be a imposing task to develop the system from scratch, a commercially available real time expert system may be a good starting point, for example the G2 system [G2].

Chapter 9

Evaluation and Conclusion

1.0 Review

This work began with an interest in pursuing a symbolic, knowledge based approach to control of complex engineering systems. The work presented in this thesis has been based on two premises. First, it was assumed that knowledge based approach offers potential for reasoning about control of complex engineering physical systems, especially when they are poorly understood or the capability for making observations of the systems is limited. Second, we surmise that problem solving involves an iterative process of building, applying, and patching models of the system under consideration.

On the basis of these premises we examined three different approaches to knowledge based control, namely:

1. *Expert control*: In this approach, a knowledge based component acts as a supervisor to the traditional controller. This approach is most appropriate when the mathematical model of the system is known.

2. *Fuzzy logic based control*: The knowledge based component is placed in the control loop in this approach. This type of controller circumvents the problem of designing a controller based on a detailed dynamic model, by employing the approach of a human operator to an ill-defined system.

3. *Qualitative reasoning for control and diagnosis:* Again, in this approach the knowledge based component is part of the control loop. This approach overcomes the problem of representing control knowledge with the help of a deep model of the system. A deep model is based on theories, facts, and principles of the domain. Hence, this approach is easily extended to assist diagnosis of process control system.

These three approaches are briefly evaluated in the next section.

2.0 Evaluation

Each of the aforementioned approaches requires different information from the user, and places different restrictions on the type of the system. The idea behind using expert control is, to overcome the difficulty that control laws are generally incomplete. So it is necessary to integrate some known logical functionalities about the plant and the control law algorithm. These functionalities ensure that the assumption behind the control law is satisfied, and also incorporate the

physical limitations of the plant being controlled.

An architecture for expert control is proposed, and to prove its usefulness, the proposed system is applied to a manufacturing system. Certain limitation to the proposed approach were noticed, hence several enhancements to the proposed architecture are suggested.

When systems are difficult to model, then it becomes difficult to control them. Nevertheless, human operators can control many such systems very easily, for example a bicycle. Fuzzy logic control promises to help control such systems, because it mimics the way human control. Although much research has been done in area of fuzzy logic control systems, developing them for a particular application is an art. Two fuzzy logic control systems of increasing complexity are developed for two very useful applications. Several new insights to fuzzy logic control systems are discussed, and the implementation details of such systems are given.

The last and most complicated approach described is qualitative control. This approach overcomes some of the problems with rule based systems such as expert control and fuzzy logic based control. There are several approaches to qualitative control. One approach is Qualitative process theory, which was used in designing a controller for robotic machining and evaluated. This approach was found to be

inefficient for real time implementation purposes, a few ideas on how to make the qualitative controller efficient are also discussed.

Once the system is modelled qualitatively, the qualitative model can, not only be used to control the system, but can also be used for diagnosis. An architecture using a blackboard system is given. The modelling method used is Kuipers' Qualitative Simulation (QSIM). QSIM is used during diagnosis when novel faults occur within the system.

3.0 Further Research

The research described in this thesis, as is often the case, is only a beginning, and much work needs to be done in each of the three areas of knowledge based control discussed in this thesis.

The architecture for supervision of traditional adaptive system needs to be modified to take advantage of better environments available on the market. An ideal environment would be a C++ language based object oriented database management system (OODBMS). The advantages of using such a system has discussed in section 6 of chapter 3. When making use of an OODBMS for process control, several processes make use of the OODBMS resources, hence issues such as garbage collection, object retrieval efficiency, exception handling,

concurrency control (object locking mechanism), etc. have to be carefully researched and analyzed.

The application of fuzzy control in this thesis has been successful, due the use of the fuzzy relational algorithm, to directly express expert control knowledge in an operational form. However, it has been found that it could be quite difficult to acquire the requisite control knowledge. This is similar to the situation in rule-based expert systems, where the *knowledge acquisition bottleneck* is widely recognized. A better approach as suggested by Tong [TON85] would be the application of some of the techniques used in building rule-based systems to alleviate the knowledge acquisition problem. Specifically, he recommends removing the restriction of relating output parameters to input parameters, and suggests the use of a more complex conceptual model, which includes intermediate state parameters, as a way of factoring knowledge acquisition.

An alternative approach would be fuzzy-model-based control, which is similar to fuzzy control except that a more complex representation of the process model is used. One possible candidate for representing the process model is qualitative process (QP) theory, but QP theory does not lend itself to use in a simple direct fashion. More research has to be done in this direction to adapt QP theory to be the model-based reasoning component of a fuzzy logic based controller.

In applying QP theory for qualitative control of systems, much research has to be done in making QP theory more efficient. Some ideas to this respect are discussed in the concluding section of chapter 7.

For diagnosis of process control systems, some ambiguities and inefficiencies that arise when using Qualitative Simulation (QSIM) algorithm should be resolved. Kuipers' considers the complexity of QSIM and concludes that the complexity of each step is a linear function of the number of parameters. In practice, the runtime of QSIM seems to be O(kt) where k is number of constraints and t the length of the longest generated behavior. QSIM may also produce extra behaviors, which do not correspond to any actual behavior of a physical system modelled by constraints. This makes QSIM 'unsound'. The reason for this problem is that simulation is local and a qualitative state description may not have enough information to determine the next state appropriately.

4.0 Conclusion

This thesis is an initial foray into the young field of knowledge based control. Several approaches have been proposed in this thesis. Each approach is applicable to a certain class of problems. These problems are delineated, and each approach is employed by taking a typical system representative of that class. Several new theoretical and implementation insights to knowledge based control are presented,

evaluated, suggestions for improvement, and directions for further research are given.

Glossary

Algorithm A procedure that specifies a logical step-wise strategy for solving a specific problem.

Artificial Intelligence (AI) A branch of computer science dedicated to the development of computer "intelligence".

Attribute The definable properties of an object. For example, the property intelligence is an attribute of the object person.

Backtracking The control strategy built into Prolog which allows a program to search backward to find alternative solutions to a problem.

Backward Chaining The process of starting with a goal state and working backward with an assumption in order to prove the specified goal. Backward-chaining is the reasoning process often used in expert system.

Binding The process of assigning a value to a variable.

Bound Variable A variable that has been assigned a specified value. In Prolog, variable stay bound for the duration of a clause.

Certainty Factor A value used to measure the confidence of a relationship or a fact. Certainty factors are often used in expert systems. A typical certainty factor ranges from -1 to 1, where -1 represents a condition of the greatest uncertainty

and 1 represents a condition of the greatest certainty.

Clause A fact or rule. Clauses are defined in Prolog with the clause statement.

Compound Goal A goal consisting of more than one sub-goal.

Cut(!) A Prolog operator that is used to stop backtracking in a program.

Decision Tree A tree representation for the decisions and sub-decisions needed to solve a given problem.

Declarative Knowledge Knowledge which can be represented as data in the form of facts and rules.

Declarative Language A language such as Prolog that allows programmers to solve problems by specifying the constraints in the form of facts and rules.

Deep Knowledge Knowledge that is based on the theories, facts and principles of a specific topic.

Deterministic Program A program or algorithm that is capable of finding only one solution to a specific problem.

Diagnostic System An expert system that solves diagnostic problems. Expert systems in this category include automotive diagnostics, process control diagnostic, etc.

Domain The definable boundaries of knowledge relating to a given topic.

Expert System Computer programs that contain the knowledge and expertise of human experts. Expert systems are one of the most popular and successful applications of artificial intelligence.

Expert System Shell A tool that is used to build expert systems. The expert system shell consists of human experts of an inference engine, knowledge interface, and user interfaces.

Fact A statement or premise about one or more objects that is true. Facts are easily written in Prolog with user-defined clauses.

Fail A standard Prolog operator that forces backtracking to occur in a program.

Formal Reasoning The process of using simple logic to make conclusions about objects.

Forward-Chaining The process of starting with an assumption and working forward to find a valid goal.

Frame Representation A technique of knowledge representation.

Frames A type of knowledge structure useful for representing complex relationships about objects. Frames function like tables or questionnaires and use locations called slots to store attributes.

Functor A term used to describe a complex object in Prolog.

Goal A problem to be solved in a Prolog program.

Goal State A condition that must be met in order to solve a specified goal. A goal state typically represents the known facts, relationships, and attributes that are related to the final goal.

Goal Tree A tree structure that is used to represent the organization of a goal. The goal tree typically indicates the necessary sub-goals that must be met in order

to prove a goal.

Heuristic A technique or structure used in programs to improve the performance of searching for solutions. One common heuristic structure is represented by a tree.

Heuristic Knowledge Knowledge that has a built-in hierarchical or top-down order.

Hierarchy A network structure in which some objects are subordinate to other objects. A common tree structure is often used to represent object hierarchy.

Inference The process of deriving new facts from old facts and rules.

Inference Engine The reasoning mechanism of an expert system. Inference engines make deductions based on known facts and user responses.

Inheritance The process of deriving values and relationships for objects based on the attributes and values of other objects. Inheritance is often used in frame-based representation systems.

Knowledge A collection of rules, facts, properties, heuristics, and relationships needed to solve a specific problem.

Knowledge Acquisition The science and art of gathering the necessary knowledge to solve a specific problem.

Knowledge Base A collection of all the knowledge in the form of rules, facts, relationships, and attributes needed for an expert system.

Knowledge Structure Structures that are used to present object, facts, rules,

relationships, procedures, and attributes. Knowledge structures are developed with traditional data structures and other structures such as frames, semantic networks, and scripts.

List An ordered collection of objects and attributes.

Meta-rule A rule that defines other rules.

Modus Ponens The fundamental inference rule used in formal logic systems. The *modus ponens* rules states the following principle: If (A is true) and (A and B are true) then B is true.

Object A conceptual or physical entity that can be expressed with attributes and relationships.

Predicate A function that can only have a true or false value. Predicates are used to express simple properties about objects.

Problem Domain The collection of knowledge that is needed to solve a specified problem.

Procedural Knowledge Knowledge that can be represented as a procedure or process. Procedural programming languages represent knowledge in the form of algorithms.

Production Rule A rule of the IF-THEN format that consists of a premise and conclusion. Production rules are often used in expert systems to represent the knowledge of the expert.

Proposition An expression in which either a true or false statement is made

about an object.

Propositional Calculus A formal logic system used to represent the true or false value about objects.

Propositional Logic *See propositional Calculus.*

Prototyping The technique of building a preliminary version of a program. Prototyping techniques are often employed by expert system developers.

Recursion A technique of defining an object or action in terms of itself. Recursion is frequently used in Prolog to control execution of programs.

Resolution The inference process used in Prolog to solve for goals.

Robotics The branch of engineering and computer science that is involved in developing functional robots.

Rule A clause that expresses relationships between facts.

Scripts A knowledge structure based on the frame representation. Scripts are used to represent sequences of events.

Semantic Network A general type of knowledge representation in which objects and values are represented as nodes and relations are represented as connections or arcs.

Semantics The meaning of expression.

Shallow Knowledge Knowledge that is based on the more superficial aspects of the specific topic.

Slot The storage unit in a frame. Slots contain information such as an object's

name, attributes, values, default value, ranges, and references to other frames.

Symbolic Processing The technique of computing with symbols. Symbolic processing is important to artificial intelligence programs because it is believed that people use symbolic processing techniques to solve problems.

Syntax The structure of an expression.

Term A simple object, compound object, list or variable.

Tree A hierarchical structure used to represent both data and knowledge.

Unification The built-in pattern matching technique that Prolog uses to match goals and sub-goals in a program.

Variable A name that can be assigned a value. In Prolog all variables must begin with an uppercase.

Vision Systems Computer systems that have the capability to recognize objects in the world.

References

[ABB87] Abbot, K., Schutte, P., Palmer, M., and Ricks, W., "Faultfinder: a diagnostic expert system with graceful degradation for onboard aircraft applications", in *14th International Symposium on Aircraft Integrated Monitoring System*, Friedrichshafen, Germany, September 1987.

[ABD89] Abdulmajid, B., and Wynne, R. J., "Initial experience in applying qualitative reasoning to process control," *Proceeding of the 28th conference on Decision and Control*, Tampa, Florida, December 1989.

[ALE83a] Allen, J. F., "Maintaining knowledge about temporal intervals," Communication ACM, vol. 26, no. 1, pg. 832-843, 1983.

[ALE83b] Allen, J. F., "Towards a general theory of action and time," *Aritificial Intelligence*, vol. 23, pp. 123-154, 1983.

[ALE85] Allen, J. F., and Hayes, P. J., "A common-sense theory of time, " in *Proceedings of the 9th Int. Jont Conf. Artificial Intelligence*, Los Angeles, CA, pp. 528-531, 1985.

[AST83] Astrom, K. J., "Theory and applications of adaptive control -A survey," *Automatica*, vol. 19, no. 5, pp. 471-486, 1983.

[AST86] Astrom, K. J., Anton, J. J., and Arzen, K. E., "Expert Control", *Automatica*, vol. 22, no. 3, pp. 277-286, 1986.

[ARZ90] Arzen, K. -E., "An architecture for expert system based feedback control," *Automatica*, Vol 25, No. 6, pp. 813-827, 1989.

[BER89] Berenji, H., Chen, Y., Lee, C., Murugesan, S., Jang, J., "A experiment-based comparative study of fuzzy logic control," in *Proceedings of the 1989 American Control Conference,* Pittsburg, pp. 2751-2753, June 1989.

[BLA37] Black, M., "Vagueness: an exercise in logical analysis," *Philosophy of Science*, vol. 4, pp. 427-455, 1937.

[BOB85] Bobrow, D. G., "If Prolog is the Answer, What is the question? or what it takes to support AI paradigms," *IEEE Trans. Software Engineering,* vol. SE-11, no. 11, pp. 1401-1408, Nov. 1985.

[BON89a] Bone, G. M., "Robotic force control for deburring using an active end effector," *M.Eng thesis*, McMaster University, Hamilton, Ontario, Canada L8S 4L7.

[BON89b] Bone, G., Lingarkar., Elbestawi, M. A., and Liu, L., "A comparative study of advanced control methods for Robotic Deburring", *ASME Winter Annual meeting, Control issues in*

manufacturing process, DSC Vol. 18, pp. 129-136, Dec 1989.

[BON91] Bone, G. M., Elbestawi, M. A., Lingarkar, R., and Liu, L., "Force control for robotic deburring," *ASME J. of Dynamics, Measurement, and Control*, 1991.

[BOO91] Booch, G., *Object Oriented design with applications*, The Benjamin/Cummings Publishing Company Inc., 1991.

[BOR76] Borrison, U. and Syding, R., "Self-Tuning Control of an Ore Crusher," *Automatica*, vol. 12, no. 1, pg. 564, 1976.

[BRA86] Bratko, I., Mozetic , I., and Lavarac, N., Automatic Synthesis and Compression of Cardiological Knowledge, *Machine Intelligence 11* (eds J. Hayes, D. Michie, J. Richards), Oxford University Press, 1986.

[BRA89] Bratko, I., Mozetic, I., and Lavrac, N., KARDIO: *A study in Deep and Qualitative Knowledge for Expert Systems*, MIT press, Cambridge, 1989.

[BUR90] Burhanpurkar, V.P., Lingarkar, R., and Sinha, N. K., "Implementation of a surveillance mobile robot for unstructured environments", *International Symposium on Robotics and Manufacturing (ISRAM)*, Vancouver, July 18-20, 1990).

[CAU88] Caudell , T. P., Doaln, C. P., Ebeid, N., and Goddard, N. H., "Real-time Issues in Knowledge-based decision support systems",

ESD/SMI Expert Systems Proceedings, pg. 389-412, Detroit, MI, USA 1988.

[CLO87] Clocksin, W. F. and Mellish, C. S., *Programming in Prolog*, Third, Ed. Berlin: Springer Verlag, 1987.

[CUN85] Cunningham, J., "Comprehension by model building as a basis for an expert system," in *Proceedings of Expert Systems 85* (ed. Merry), pp. 259 - 272, Cambridge University Press, 1985.

[CZO81] Czogala, E., and Pedrycz, W., " Some problems concerning the construction of algorithms of decision-making in fuzzy systems," *International Journal of Man-Machine Studies*, vol. 15, pp. 201-211, 1981.

[CHU89] Chunyu, H. T., Toguchi, K., Shenoi, S., Fan, T. L., "A technique for designing and implementing fuzzy logic control," in *Proceedings of the 1989 American Control Conference*, Pittsburg, pp. 2754-2755, June 1989.

[DAN86] Daneshmend, L. K., and Pak, M. A., "Model reference adaptive control of feed force in turning, " *Trans. ASME, Journal of Dynamics Systems, Measurement and Control*, vol. 108, pp. 215-222, 1986.

[DOD89] Dodhiawala, R., Sridharan, N. S., Raulefs, P., and Pickering, C., "Real-Time AI Systems: A Definition and an Architecture", *IJCAI*

proceedings, Morgan Kaufmann, August 1989.

[ELB87] Elbestawi, M. A., and Sagherian, R., "Parameter adaptive control in perpheral milling," *Int. J. Mach. Tools Manufact.* vol. 27, no. 3, pp. 399-414, 1987.

[ENG88] Englemoore, R., and Morgan, T (Editors), *Blackboard Systems,* Addision-Wesseley, 1988.

[FIK85] Fikes, R., and Kehler, T., "The role of frame-based representation reasoning," *Communication of the ACM,* vol 29, no. 9, pp. 904-920, September 1985.

[FIS88] Fisher, W. D., and Mujtaba, M. S., "Using the the HCTL-1000 Motor control chip for coordinated robot motions," *Proceedings of the American Control conference,* pp. 663-664, June 1988.

[FOR81] Fortesque, I. R., Kershenbaum, I. S., and Ydstie, B. E., "Implementation of self-tuning regulators with variable forgetting factors," *Automatica,* vol. 17, pp. 831-836, 1981.

[FOR84] Forbus, K. D., "Qualitative process theory," *Artificial Intelligence,* Vol 24, pp. 85-168, 1984.

[FOR85] Forbus, K. D., "The problem of existence", *UIUCDCS-R-1288,* university of Illinois, 1985.

[FUJ89] Fujimoto, J., Nakatani, T., and Yoneyama, M., "Speaker-independent word recognition using fuzzy pattern matching," *Fuzzy*

Sets and Systems, vol. 32, no. 2, pp. 181-192, 1989.

[G2] G2 Real-time expert system, *Gensym Corporation,* 125 Cambridge Park Dr., Cambridge, MA 02140.

[GOO84] Goodwin, G. C. and Sin, K. S., *Adaptive Filtering, Prediction and Control,* Prentice Hall, New Jersey, 1984.

[GUP89] Gupta, M. M., Kiszka, J. B., "A Study of multivariable fuzzy controller under Godel's implications," in *Proceedings of the 1989 American Control Conference,* Pittsburg, pp. 2759-2764, June 1989.

[HAY79] Hayes, P. J., "The naive physics manifesto," in *Expert Systems in the micro Electronic Age,* (ed. D. Michie), Edinburgh University Press, 1979.

[HAY83] Hayes, P. J., "The logic of frames," in *Frame Conceptions and Text Understanding,* D. Metsing, Ed. Berlin: Walter de Gruyter, 1983.

[HAY85a] Hayes, P. J., "The second naive physics manifesto," in *Formal theories of common sense,* (ed. Hobbs and Moore), Abex Publishing Corporation, Norwood, New Jersey 1985.

[HAY85b] Hayes, P. J., "Ontology of liquids," in *Formal theories of common sense,* (ed. Hobbs and Moore), Abex Publishing Corporation, Norwood, New Jersey 1985

[HIR89] Hirota, K., Arai, Y., and Hachisu, S., "Fuzzy controlled robot arm playing two dimensional ping pong game," *Fuzzy Sets and Systems,*

vol. 32, no. 2, pp. 161-180, 1989.

[HOB85] Hobbs, J. R., and Moore, R. C., *Formal theories of the Commonsense world*, Ablex Publishing Corporation, Norwood, New Jersey 1985.

[ISE85] Isermann, R., and Lachmann, K. H., "Parameter-adaptive control with configuration aids and suervission fucntions," *Automatica*, vol. 21, pp. 443-440, 1985.

[JAY89] Jay, C., and Knaus, R., "Frames in Prolog," Part I and II, *AI Expert*, vol. 4, no 3, pp. 19-24, March 1989, no. 5, pp. 19-24, May 1989.

[KAM88] Kamran, P., and Chignell, M., *Expert Systems for Experts*, New: John Wiley 1988.

[KIC76] Kickert, W., and Lemke, H. V. N., "Applications of a fuzzy controller in a warm water plant," *Automatica*, vol. 12, pp. 301-308, 1976.

[KIT86] Kitzmiller, C. T., and Kowalik, J. S., "Symbolic and numerical computing in knowledge-based systems," in *Coupling Symbolic and Numerical Computing in Expert Systems*, J. S. Kowalik, Ed. Amsterdam: North Holland, 1986, pp. 3-15.

[KLE84] Kleer, J. De., and Brown, J. S., "Theories of Causal ordering," *Artificial Intelligence*, Vol 29, pp. 33-61, 1984.

[KLE87] Kleer, J. De., "Multiple representations of knowledge in a mechanics problem-solver," in *Proceeding of the IJCAI*, Morgan Kaufmann, Los Altos 1977.

[KLE86a] Kleer, J. de, An Assumption-based TMS, *Artificial Intelligence,* vol. 28, pg. 127-162, 1986.

[KLE86b] Kleer, J. de, Extending the ATMS, *Artificial Intelligence*, vol. 28, pg. 163-196, 1986.

[KLE86c] Kleer, J. de, Problem solving with the ATMS, *Artificial Intelligence*, vol 28, pg. 197-224, 1986.

[KLE87] Kleer, J. de, and William, B., Diagnosis of Multiple faults, *Aritficial Intelligence*, vol. 32, pg. 97-130, 1987.

[KLI88] Klir, G. J., and Folger, T. A., *Fuzzy Sets, Uncertainty, and Information*, Prentice Hall, 1988.

[KOF89] Koffman, S. J., Fullmer, R., and Brown, R., "Fuzzy logic control of a fluidized bed combuster," in *Proceedings of the 1989 American Control Conference,* Pittsburg, pp. 2756-2758, June 1989.

[KOS88] Koschmann, T., and Evens, M. M., "Bridging the gap between object-oriented and logic programming," *IEEE Software,* pp. 21-27, July 1988.

[KOS90] Kosko, B., "Fuzziness Vs Probability," *International Journal of General Systems*, Vol 17 pp. 211-240, 1990.

[KUI84] Kuipers, B., "Commonsense reasoning about causality: deriving behaviour from structure," *Artificial Intelligence*, vol 24, pp. 169-203, 1984.

[KUI86] Kuipers, B., "Qualitative simulation," *Artificial Intelligence*, vol 29, 289-338, 1986.

[LAF88] Laffey, T. J., and Cox, P. A., Schmidt, J. L., Kao, S. M., and Read, J. Y., "Real-time knowledge-based systems," *AI Magazine*, vol. 9, no. 1, pp. 27-45, Spring 1988.

[LIN88] Lingarkar, R., Bone, G., Liu, L., Elbestawi, M. A., and Sinha, N. K., "Fuzzy logic controller for robotic machining," *Proceeding of Canadian Conference on Electrical Engineering Conference*, pp. 595-599 Vancouver, November 1988.

[LIN89] Lingarkar, R., Liu, L., Elbestawi, M. A., and Sinha, N. K., "A Knowledge-based approach to adaptive control in manufacturing systems" *Proceedings of the American Control Conference*, pp. 366-371, Pittsburg, Pennysylvania, June 1989.

[LIN90] Lingarkar, R., Liu, L., Elbestawi, M. A., and Sinha, N. K., "Knowledge-based adaptive control in manufacturing systems: A case study ", *IEEE transactions on SMC*, Vol 20, No. 3, pp. 606-618, May/June 1990.

[LIN90} Lingarkar, R., Elbestawi, M.A, and Sinha, N. K., "Model-based

resoning for knowledge-based control systems", *Proceedings of the Canadian Conference on Electrical Engineering,* Ottawa, September 1991.

[LIU89] Liu, R., Han, Y., Lingarkar, R., Sinha, N. K., "On compliant motion control of robots," *Proceedings of the IEEE Industrial Electronics Conference,* Philadelphia, Nov 1989.

[LUG89] Luger, G. F., and Stubblefield, W. A., *Artificial Intelligence and the design of Expert Systems,* The Benjamin/Cummings Publishing Company, Inc. 1989.

[LJU83] Ljung, L., and Soderstrom T., *Theory and Practice of Recursive Identification,* Cambridge: MIT Press, 1983.

[MAM74] Mamdani, E. H., "Applications of fuzzy algorithms for simple dynamic plant," *Proceedings of IEE,* vol. 121, pp. 1585-1588, 1974.

[MAM75] Mamdani, E. H., and Assilian, S., "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man Machine Studies,* vol. 7, no 1, pp. 1-13, 1975.

[MAN89] Manzoul M., and Sripadam, K., "Fault-tolerant systolic array for single-input single-output fuzzy logic controllers," in *Proceedings of the 1989 American Control Conference,* Pittsburg, pp. 2765-2766, June 1989.

[McC69] McCarthy, J., and Hayes, P. J., "Some philosphical problems from the standpoint of AI," *Machine Intelligence 4* (ed. Meltzer and Michie), Edinburgh, University Press, UK 1969.

[MIN75] Minsky, M. A., "Framework for representing knowledge," in *The Psychology of Computer Vision*, P. Winston, Ed. New York: McGraw Hill, 1975.

[MOH88] Mohammed, Y, "Parameter adaptive control in peripheral milling," *M. Eng. thesis*, McMaster University, Hamilton, Canada 1988.

[MOO87] Moore, R. L., Hawkinson, L. B., Levin, M., Hofman, A. G., Mathews, B. L., and David, M. H., *Expert System Methodology for Real-Time Process Control*, Automatic Control - World Congress, IFAC, Oxford UK, pg. 274-281, 1987.

[MOR88] Morgan, A., "Real-time Expert Systems for Industrial Applications", *Fourth International Expert System Conference"*, London, June 1988.

[MUR88] Murty, S. S., Qualitative reasoning at multiple resolution, *Proceeding of the Seventh National Conference on Artificial Intelligence AAAI-88*, Vol I, pp. 296-300, St. Paul, Minnesota.

[OST76] Ostergaard, J. J., "Fuzzy logic control of a heat exchanger process," *Internal report 7601*, Technical University, Denmark 1976.\

[NII86a] Nii, H. P., "Blackboard Systems: The Blackboard Model of

Problem Solving and the evolution of Blackboard Architecture," *AI Magazine*, vol. 7, no. 3, pg. 38-53, 1986.

[NII86b] Nii, H. P., Blackboard Systems: Blackboard Application Systems, *AI Magazine*, vol 7, no. 3, pg. 82-106, 1986.

[ONO89] Ono, H., Ohnishi, T., and Terada, Y., "Combustion control of refuse incineration plant by fuzzy logic," *Fuzzy Sets and Systems*, vol. 32, no. 2, pp. 193-206, 1989

[ORE85] O'Reilly, C. A., and Cromarty, A. S., "Fast is not real-time: Designing effective real-time AI systems", *Proceeding SPIE*, Vol 548, pg. 249-257, 1985.

[PAT87] Patridge, D., "The scope and limitations of first generation expert system," *Future Generation Computer System*, Amsterdam: North Holland, pp. 1-10, 1987.

[PAU82] Paul, F. W., Getty, T. K., and THomas, J. D., "Defining of iron castings using a robot positioned chiper," *Robotics research and advacned applications,* ASME, pp. 269-288, 1982.

[REI87a] Reiter, R., "A theory of Diagnosis from first principles", *Artificial Intelligence*, vol 32 (1), pg. 57-95, 1987.

[REI87b] Reiter, R., and Kleer, J. de, "Foundations of Assumption Based Truth Maintenance System, Preliminary Report", In *Proceedings of the AAAI*, Morgan Kaufmann, Los Altos.

[RIC88] Ricketts, G. V., "How to do more in less time", *AI Expert*, pg. 46-52, January, 1988.

[RUT76] Rutherford, D. A., and Carter, G. A., "A heuristic adaptive controller for sinter plant," *Proceeding of the 2nd IFAC Symposium on Automation in Mining, Mineral and Metal Processing*, Johannesburg, pp. 315-324, 1976.

[ROW88] Rowe, N. C., *Artificial Intelligence Through Prolog*, Prentice Hall, 1988.

[SAN85] Sanoff, S. P., and Wellstead, P. E., "Expert Identification and Control," in *Identification and System Parameter Estimation*, (Ed. H. A. Barker and P. C. Young), Pergmon Press, Oxford U. K., 1985.

[SCH86] Schuttle, P., and Abott, K., "An artificial intelligence approach to onboard fault monitoring and diagnosis for aircraft applications", in *AIAA Guidance*

[SEB86] Seborg, D. E., Edgar, T. F., and Shah, S. L., "Adaptive Control Strategies for Process Control: A survey," *AIChE, J.*, 32, pg. 881, 1986.

[SHO76] Shortliffe, E. H., *Computer-based Medical Consultations: MYCIN*, American Elsevier Publishing Company, Inc., New York, 1976.

[SIN83] Sinha, N. K., and Kuzsta, B., *Modeling and Identification of*

Dynamic Systems, Van Nostrand Reinhold, New York, 1983.

[STA88] Stankovic, J. A. and Ramamritham, "Hard Real-Time Sytems: A Tutorial," *Computer Society Press* IEEE, Washington, D. C., 1988.

[STE86] Stefik, M. and Bobrow, G., "Object oriented programmiing: Themes and Variations," *The Artificial Intelligence Magazine*, vol. 6, no. 4, pg. 40, 1986.

[STE86] Sterling, L., and Shapiro, E., *The Art of Prolog,* Cambridge: MIT Press, 1986.

[SUG89] Sugeno, M., Murofushi, T., Mori, T., Tatematsu, T., and Tanaka, J., "Fuzzy algorithmic control of a model car by oral instructions," *Fuzzy Sets and Systems*, vol. 32, no. 2, pp. 181-192, 1989.

[TLU77] Tlusty, J., and Elbestawi, M. A., "Analysis of transients in a adpative control servomechanism for milling with constant force," *Trans. ASME, J. Engg, Ind.*, vol. 27, no. 3,, pp. 399-414, 1977.

[TLU86] Tlusty, J., and Wegerrif, D., "Compensating for deflections of a robot in light machinining operations," *Robotics: theory and applications,* DSC-vol3, ASME, pp. 91-100, 1986.

[TOG83] Togai, M., "Analysis and control of fuzzy dynamic systems," *NAFIP-II*, Schenectady, New York 1983.

[TON76] Tong, R. M., "Some problems with the design of fuzzy logic controllers," *CUED*, Cambridge University, United Kingdom 1976.

[TON79] Tong, R.. M., "The construction and evaluation of fuzzy models," in *Advances in Fuzzy set theory and applications* (M. Gupta et al, eds), pp 559-576, Amsterdam: North Holland, 1979.

[TON85] Tong, R. M., "An annotated bibliography of fuzzy control", in M. Sugeno, Editor, *Industrial Applications of Fuzzy Control*, pg. 249-269, Elsevier Science, 1985.

[UMB80] Umbers, I., and King, P., "An analysis of human decision-making in cement kiln control and the implications for automation," *International Journal of Man Machine Studies*, vol. 12, pp. 11-23, 1980.

[URA76] Uragami, M., Mizumoto, M., and Tanaka, K., "Fuzzy robot controls," *Journal of Cybernetics*, vol. 6 pp. 39-64. 1976.

[VAN77] Van Amerongen, Lemke, H. R., and Van der Veen, J. C. T., "Auto pilot for ships designed using fuzzy sets," in *Digital Computer applications to process contro*l, IFAC and North-Holland, 1977.

[WAL88] Walters, J. R., and Nielsen, N. R., *Crafting Knowledge-based Systems*, New York: John Wiley, 1988.

[WEI88] Weiskamp, K., and Hengl, T., *Artificial Intelligence Programming with Turbo Prolog*, New York: John Wiley, 1988.

[WEL81] Wellstead, P. E., and Sanoff, S. P., "Extending self-tuning algorithm," *Int. J. of Control*, vol. 34, pp. 433-442, 1981.

[WIL84] Williams, B. C., "Qualitative analysis of MOS circuits," *Artificial Intelligence*, Vol 24, pp. 281-346, 1984.

[WIT84] Wittenmark, B., and Astrom, A. J., "Practical issues in the implementation of self-tuning control," *Automatica*, vol. 20, pg. 595-603, 1984.

[YAM89] Yamkawa, T., "Stabilization of an invertted pendulum by a high speed fuzzy logic controller hardware system," *Fuzzy Sets and Systems*, vol. 32, no. 2, pp. 161-180, 1989.

[YDS85] Ydstie, B. E., "Adaptive Control and Estimation with Forgetting Factors," in *Indentification and System Parameter Estimation*, (Ed. H. A. Barker and P. C. Young), Pergmon Press, Oxford U.K, pg. 1761, 1985.

[YEH88] Yeh, S., and Wu. C., *Solutions to Time Variant problems in Real-Time Expert Systems*, Fourth Conference on Artificial Intelligence for Space Applications, Washington DC, pg. 19-28, 1988.

[ZAD65a] Zadeh, L. A., "Fuzzy Sets," *Information and Control*, Vol 8, pp. 338-353, 1965.

[ZAD65b] Zadeh, L. A., "Fuzzy Sets and Systems," In: *System theory*, edited by J. Fox, Polytechnic Press, Brooklyn, N.Y., pp. 29-37, 1965.

[ZAD71] Zadeh, L. A., "Similarity relations and fuzzy ordering," *Information Science*, vol. 3, pp. 177-200, 1971.

[ZAL82] Zalucky, A., and Hardt, D. E., "Active control of robot structure deflections," *Robotics research and advanced applications*, ASME, pp. 83-100, 1982.

[ZIM85] Zimmermann, H. J., *Fuzzy Set Theory and Its Applications*, Kluwer Academic Publishers, Boston, 1985.