DISTRIBUTED MODELING AND CONTROL OF NONLINEAR SYSTEMS WITH APPLICATIONS IN ROBOTICS: A FUZZY LOGIC-BASED SWITCHING APPROACH

BY

JOSE ALEJANDRO RUEDA MEZA

94.

A Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba
CANADA

ISBN 0-612-13490-3

Name Jose Alejandro Rueda Meza

Dissertation Abstracts International and *Masters Abstracts International* are arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation or thesis. Enter the corresponding four-digit code in the spaces provided.

Electronics and Electrical Engineering
SUBJECT TERM

0544
SUBJECT CODE

UMI

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES
Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984

Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

DISTRIBUTED MODELING AND CONTROL OF NONLINEAR SYSTEMS WITH APPLICATIONS IN ROBOTICS: A FUZZY LOGIC-BASED SWITCHING APPROACH

BY

JOSE ALEJANDRO RUEDA MEZA

A Practicum submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Abstract

The problem of modeling and control of complex systems is studied in this thesis. A hierarchical structure designed using qualitative information is used to coordinate the operation of a number of local modules. The coordination system consists of a fuzzy logic processor. A design method is proposed and a number of analytical results are presented. The hierarchical architecture is applied to the design of distributed controllers and distributed models. Application examples of modeling and control in the area of robotics are considered. This thesis demonstrates that certain artificial intelligence techniques along with standard methods can provide excellent and practical results in complex modeling and control problems.

Keywords: nonlinear systems, nonlinear control, control of robots, fuzzy control, modeling.

Acknowledgements

I am very grateful to the National Council of Science and Technology of Mexico (Consejo Nacional de Ciencia y Tecnologia, or CONACYT), for sponsoring my graduate studies.

The advice and support of Professor Witold Pedrycz during the development of this work is gratefully acknowledged. I would like to thank him for all the ideas and suggestions. Without his help this work could not have been done. I also thank him for his financial support to attend conferences, and during the last year and a half when the research was completed.

I thank Professor Steve Onyshko for reviewing the preliminary versions of this thesis and providing me with valuable feedback. I particularly would like to thank him for all the discussions that have extended my vision of engineering and teaching. I also thank him for all the help and support that I received from the Department of Electrical and Computer Engineering at the University of Manitoba while he was the head.

I also would like to thank The National Science and Engineering Research Council of Canada (NSERC) and MICRONET Canada for the financial support to attend conferences and for the support during the last few months while this thesis was being completed. Thanks to the Faculty of Graduate Studies of the University of Manitoba for awarding me a Graduate Studies Fellowship for eight months prior to the completion of this thesis.

I thank Professor Edgar Sanchez of the Mechanical and Electrical Engineering School at Universidad Autonoma de Nuevo Leon for being a member of the committee as an external examiner, and for all his very valuable suggestions.

I also would like to thank Professor Subramaniam Balakrishnan for being a member of the examining committee.

I would like to thank Mr. Hugh Pollitt-Smith for proofreading this thesis, and for all the useful suggestions.

To my dear mother Lucia Josefina,
my brothers Jorge Adalberto,
Miguel Angel and Victor Manuel,
very specially to Alice for all her support,
and to the memory of my grandparents
Mama Chepina and Jose.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preliminaries

Control theory is a well-established area of research which creates an intersection between mathematics, engineering, computer science, physics, and often economics.

This science was recognized by the Greeks as they developed the first float regulator mechanisms in the period from 300 to 1 B.C. (Dorf [14]). Since then, there have been many engineering applications which have provided a platform for the development of theoretical results, as well as abstract situations which have entertained the curious scientific mind.

Today, it is widely accepted that a *control system* is a "system consisting of interconnected components built to achieve a desired purpose" (Dorf [14]). The generalized tool is the concept of *feedback*. The tremendous amount of research in control theory has provided us with sometimes redundant procedures for the solution of problems, and the criteria to select a specific technique is still based on human judgment and need. There are many ways to design a controller for a specific problem, including different levels of complexity, generalization, and performance. Thus, for a set of specifications there are many control techniques that can be used with different levels of success. The success or failure of a technique is based on the degree of satisfaction of the specifications.

There are many control techniques that are not physically realizable, or simply unfeasible, as the technology required may not be available. In other cases, economic design factors and the need for prompt availability might also constrain the set of techniques that would eventually be chosen to solve a control problem.

Due to the existence of standard requirements in industry, control engineering

has created a set of standard methods. These methods incorporate objectives and human assessment, as well as available technology to provide functional control. In the following section, the general process of design of a controller is presented.

General Control Problems

The analysis and design of a control system can be summarized as follows:

- 1. Understanding of the problem: this stage consists of mathematical modeling and identification, and formally establishing the control problem.
- 2. Design of the controller: this stage consists of a constrained search for control techniques. It is carried out according to the specifications of the control problem. If a solution cannot be found in the existing set of methods, then research is started and usually aimed at extending the theory and sometimes the engineering. Research in control often produces results that force a compromise in the control specifications, which results in a partial solution to the problem, and this is one of the reasons why theory and practice in control seem to be different disciplines.
- 3. Implementation: this stage consists of an implementation of the control algorithm using the existing technology, and often developing new technology. Many control techniques are aborted at this stage when the technology required is beyond expectations or simply unavailable. The control requirements are sometimes reformulated so that a standardized technology can be applied.

At every stage there are human factors that shape the process. Decisions have to be made as to which model is the best, which controller can offer the optimal performance, and which controller is the easiest to implement, or the most convenient and reliable.

These questions have been posed many times and the answers can be summarized as follows (Astrom [5]). The most common way to model a real world system is by using linear differential and difference equations of 1st and 2nd orders. More than 90% of the industrial processes can be modeled this way. These models are good enough in many industrial applications, including, temperature and speed control, and chemical, biological and economic processes.

The above indeed suggests that control engineering uses a very small set of methods from control theory. An excellent monograph on modeling of real world problems is presented in Brown [9].

Standard Controllers

Proportional-Integral-Derivative controllers (or simply *PID*) have been the "bread and butter of control engineering" (Astrom [5]). They are the product of a combination of feedback signals, weighted in such a way as to produce the control command. These controllers are sufficient in solving many problems when the performance requirements are modest (as they are, usually). For example, it is often required to keep a temperature or speed constant.

Technological changes have not affected the PID control algorithm; instead, they have been improved from pneumatics to electron tubes, transistors, integrated circuits, microprocessors and computers. PID controllers, along with other engineering creations like sequential machines, are used to build the automation systems for energy production, transportation, and manufacturing (Astrom [5]).

In addition to PID controllers, linear controllers (designed using the theory of linear systems) are also very common. These are found, for example, in aircraft control systems. The design of linear controllers is more complicated than the design of PID controllers. There exist many design techniques with different goals. Linear controllers can be used when theoretical support is required, for which these controllers are generally a more reliable alternative to PID controllers. Linear control is an extensive area of control theory, and a very active research area. Some of the most common methods of linear control are applied in optimal control.

Standard Performance Requirements

In many applications, the range of operation of the system permits the use of a linear model and a simple controller with acceptable performance. However, when the required range of operation is wide, a more advanced model and controller are needed. Although PID or optimal controllers are generally robust to changes in the dynamics of the process, their performance can deteriorate as the conditions of operation change. In some cases they have been replaced by "intelligent controllers" which incorporate techniques from artificial intelligence (Astrom [6]).

Sometimes a more complicated nonlinear model is available, and it is still desired to accomplish the control objective with the simplest controller. In this case, performance is often compromised and the operating conditions must be carefully planned, since a simple controller cannot guarantee good performance for every situation, and changes in the operating conditions may be caused by the nonlinearities of the model. In this case, a nonlinear controller may be the solution, but the implementation of the algorithm might not be feasible.

In other cases, several linear models are identified that are valid in specific conditions, and several controllers are designed; for example, the PUMA 500 robot manipulator has two PID-type controllers designed based on two models of the robot (Wen [81]).

The theory of nonlinear systems can be used in the control of complex systems. This area of control contains a vast amount of results, but these are often very specialized and difficult to implement. In this thesis, the problems of modeling and control of nonlinear system are considered. The approach taken is based on both standard control methods and artificial intelligence, as explained in the following section.

1.2 A New Approach: Fuzzy Switching

The technique presented in this thesis for control of complex systems is based on a coordination system that provides a smooth switching of simple controllers and models of the plant. One of the motivations for considering this approach is that linear models are easy to obtain in practice, and simple controllers are readily available. Thus, it is not necessary to replace an existing controller; instead, the performance can be improved by adding modules composed of simple control blocks and a coordination system.

The Architecture

In control theory, there has been a strong emphasis on the development of exact methods for well-defined problems, and in comparison, little work has been done in the development of qualitative methods, which would incorporate non-numerical properties in the design. These qualitative methods give rise to knowledge-based systems (Astrom [6]).

An architecture will be introduced which is based on switching that is general enough to be used in many modeling and control applications. A knowledge-based system will be considered for coordination of local elements, and it will be shown in simulations that an excellent performance can be achieved.

1.3 The Thesis

The **problem** under consideration is the design of an architecture for modeling and control of complex systems based on modular control elements. Each control module is designed to efficiently deal with a specific task, covering in this way, a wide range of operating conditions. A design method and a number of theoretical studies are presented.

The **scientific method** used to analyze the proposed system is based on control theory. Concepts from fuzzy logic theory are used in the design of a knowledge-based system for coordination.

The **novelty** of the approach is the use of simple modules in an architecture for modeling and control based on a smooth switching accomplished by a knowledge-based system.

The **applications** considered are a water-level control system of a tank, and two robot manipulators with one and two degrees-of-freedom. The resulting systems are analyzed and simulation results are presented.

There are two conceptual parts of this thesis. Firstly, the problem of distributed control is addressed. Subsequently, the analysis is concentrated on the problem of distributed modeling.

Chapter 2

Distributed Control

In this chapter, the switching architecture is introduced. The chapter starts with a discussion on measurement of performance of control systems, and the control objective is defined. This is followed by a discussion on hierarchical control and fuzzy control. The concept of distributed control is introduced.

2.1 Performance of Control Systems

In general, a control system is designed to satisfy a set of specifications, one of the most important being stability. Performance is difficult to measure taking into account all the factors of a control system. The following factors are considered to estimate the performance of a control system:

- Observation by the designer or operator at the application level,
- Numerical performance indices,
- Concepts of stability in the sense of Lyapunov.

In the context of performance indices, improving the performance of a control system means minimizing or maximizing the value of a function. A performance index can be qualitative or quantitative. The following interpretation of performance indices is considered.

Qualitative Performance Indices: These include observations of the response of a system, such as, overshoot, speed of response, and disturbance rejection or robustness. Robustness is defined as the ability of the system to maintain its qualitative

behaviour under disturbances and uncertainty, where disturbances are external unknown signals and uncertainty is present when the model of the system is not perfectly known. These qualitative performance measurements are sometimes quantified by sensors so that a numerical estimate is available. The design method that will be proposed requires information about the response of the system and it is reasonable to assume that this information is available.

During the design of a control system, it often occurs that some of the desired qualitative properties must be compromised. For example, if the response time is decreased, it is very likely that the overshoot will increase and the system will become very sensitive to disturbances. On the other hand, if good disturbance rejection properties are desired, it might be necessary to compromise the response time.

Quantitative Performance Indices: The following are some commonly used formulas that depend on error measurements. The error is the difference between the current output and the desired response:

$$\begin{aligned} I_1 &= \int_o^t e(\tau)^2 \, d\tau \\ I_2 &= \int_o^t \tau e(\tau)^2 \, d\tau \\ I_3 &= \int_o^t \mid e(\tau) \mid \, d\tau \\ I_4 &= \int_o^t \tau \mid e(\tau) \mid \, d\tau \end{aligned} \tag{2.1}$$

where I_1 and I_3 represent a measure of the overall performance in terms of the error signal without considering the time when different values of error occur. I_2 and I_4 include information regarding the time frame of the error signal. This is useful in ensuring that small errors occurring when a long time has passed carry more weight in the performance measurement. The first two indices are smooth functions, mainly used in theoretical analyses, while the last two, which are non-smooth functions due to the absolute value, are used in practical implementations.

The implementation of a controller is an engineering task, where the control algorithm is applied in a specific problem in a computer simulation, or it is put to work in a piece of hardware. In this thesis the term implementation will refer to the application of an algorithm to a specific problem at the simulation level, and thus the terms *implementation* and *simulation* will be used interchangeably throughout the thesis.

2.2 The Control Objective

In this study, it is necessary to satisfy qualitative and quantitative performance indices simultaneously, specifically, the following requirements:

- maximize the speed of response, that is, minimize the response time,
- maximize regulation, that is, minimize sensitivity to disturbances.

It is required to use standard control modules to satisfy these requirements. A knowledge-based system to coordinate their operation will be designed. The design method must encapsulate qualitative requirements into a quantitative algorithm for control of complex systems, and also maximize the best properties of each module minimizing unwanted effects.

The control objective can be viewed in the context of optimal control as the problem of optimizing the transient response with good behaviour in the steady-state. The problem is complicated. A solution is provided by Rotstein and Sideris [54] where a controller is designed with time-domain constraints. In their approach the response time can be set and a suitable optimal controller can be designed. However, they only provide a time-constraint to the optimization algorithm, and they do not address performance issues. In the approach presented in this thesis, the performance of the controller is fully considered. To obtained the best results, an integration of several disciplines is proposed.

The goal is achieved by proposing a two-level control architecture using control modules and a knowledge-based coordinator arranged in a hierarchical structure. Before the detailed presentation of the architecture, a discussion of hierarchical control structures is included in the following section.

2.3 Overview of Hierarchical Control Structures

The fast computers available today allow the use of hierarchical structures for control of complex systems. A hierarchical control structure is composed of several levels of control. Usually, there is a standard controller at a lower level directly operating the plant, and a knowledge-based system at a higher level supervising, adjusting the parameters of the controller, and protecting the control system. A knowledge-based system is also known in the literature as an *intelligent system*, and in control applications as an *intelligent controller*.

Intelligent systems have also been applied directly as controllers, but this direction will not be pursued in this thesis since, in general, an automatic control system based on control theory is more accurate and reliable. However, an intelligent system is better in making decisions that depend on quantitative measurements of qualitative information.

There have been successful applications of expert systems, neural networks, fuzzy systems and combinations (Fukuda [18], He [22], Lee [36], Rovithakis [55], Sanchez [63], Shibata [64], Tan [74], Tseng [76]). Intelligent systems are used for tuning of regulators by He [22], Tan [74], and Tseng [76]. These controllers work well, but there are problems when trying to adjust the parameters due to nonlinear couplings. For this reason, these techniques can only be successfully applied to the control of simple systems. The approach taken by Chen [39], He [22], and de Silva [65] is to design a PID controller with variable parameters. These parameters are adjusted according to *if-then* rules, and since the parameters of simple regulators as PIDs are strongly coupled, it is not guaranteed that by modifying a single parameter, the effect of the others will not be affected.

A natural application of intelligent systems in hierarchical control structures is their use as supervisors or coordinators of simple regulators (He [22], Rueda and Pedrycz [57]-[60], Tan [74]). This approach is considered, and a fuzzy system is used to select degrees of activation of different local regulators working in parallel. The fuzzy system makes decisions based on the status of the control system.

This idea of switching dynamic systems is much more complex than the switching of scalar functions for regression-type approximations presented by Takagi and Sugeno [72]. The design methods, the range of applications, and the analyses are quite different. A brief overview of fuzzy control is presented next.

2.4 Overview of Fuzzy Control

Fuzzy logic was introduced by L.A. Zadeh [84], and almost immediately adopted by control engineers. Fuzzy logic-based controllers, also known as fuzzy controllers have been applied in industry, in most of the cases to processes that have already been controlled successfully by a human operator. Some of the pioneer experimental work was done by Mamdani and Kickert [32], [40], [41].

The standard fuzzy controller is a rule-based system that captures the control abilities of a human operator. Several implementations have been reported, but

the control inference and generation of a control command follow some standard methods. In terms of applications, fuzzy controllers are generally very simple and very robust, but they lack formality, and for this reason, they have not been fully accepted as a valid method in control theory. Nevertheless, fuzzy logic is a very good tool in capturing qualitative properties of a system, and thus, fuzzy controllers are becoming very popular in control engineering.

Due to the nature of the operations in a standard fuzzy controllers, they are not as accurate as automatic controllers derived from control theory. Perhaps the best application for fuzzy systems in control loops is as supervisors or coordinators. In this way the qualitative knowledge that has been almost totally ignored in control theory can be incorporated.

The standard fuzzy controllers are nonlinear static mappings, and in some cases even dynamic nonlinear mappings. Due to the nonlinearities, the analysis in control loops can be a nontrivial matter. However, techniques from nonlinear system theory can be applied since there is a considerable amount of research available that can be used in proving stability, for example. The details of the proposed architecture are presented in the following section.

2.5 The Control Architecture

The proposed hierarchical control structure is composed of a coordinator, several regulators or local controllers, and an aggregation block as shown in Figure 2.1.

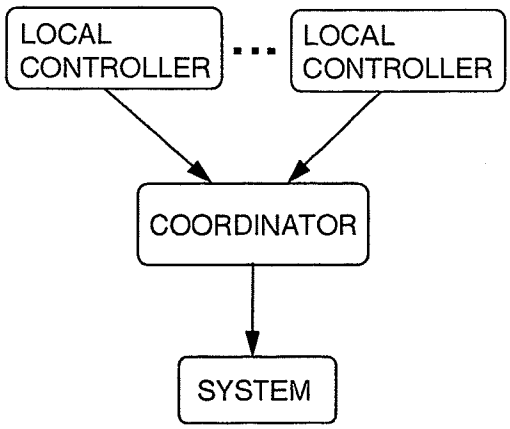

Figure 2.1: Hierarchical Controller.

The coordination system is designed to identify the current status of the system, select and activate the control modules, and coordinate their contributions to the final control action. The coordinator uses feedback measurements to make decisions, which are based on the qualitative knowledge about control encapsulated in the fuzzy system.

The proper selection of the local control modules is a very important factor in the design of the hierarchical controller. They must be selected so that each one satisfies one of the qualitative control requirements. This information is encoded in the fuzzy system and used to calculate degrees of activation of the local controllers.

A block diagram of the control architecture is shown in Figure 2.2. The system makes use of the two basic feedback quantities which are error e and its change, which is denoted by Δe. These variables are taken with respect to a reference, which may be constant or time-varying.

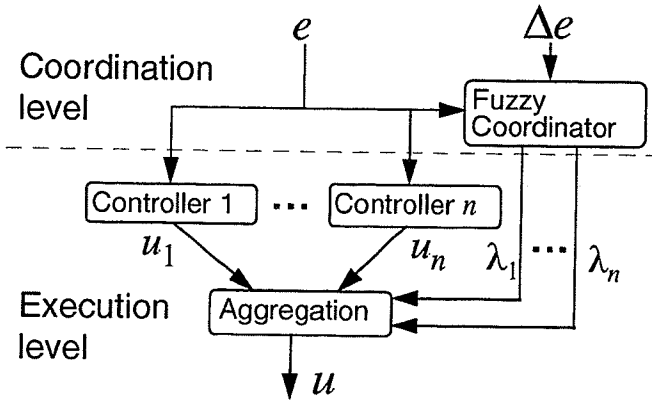

Figure 2.2: Fuzzy Hierarchical Controller.

The coordinator produces an activation variable λ_i for each of the local controllers i, which is then used to produce the final control signal u. The corresponding contribution of each of the controllers is u_i.

The design method consists of the following steps:

- Specify the range of operation in terms of error and change of error, and the control objectives for each of the required local controllers; as well, specify the number of controllers.

- Construct the fuzzy coordinator based on information derived from the observation of the response of the individual control modules. At this stage the functionality of fuzzy logic allows the design of a simple static system that can make decisions about control based on standard feedback signals.

The objectives of the local controllers may be very different. In the regulation and tracking problems being considered, it seems natural to use two controllers: one to provide the fastest possible response, and the other to provide the best regulation characteristics, namely, no overshoot, zero steady state error, and good disturbance rejection.

Stability for each individual loop must be assured since, as it will be shown, the stability properties of the overall system depend upon the stability of each loop. The objectives can be achieved individually by applying some standard techniques of control engineering. Notice that the two mentioned controllers have conflicting objectives, since a short response time usually increases oscillations, and regulation usually increases the response time. One of the main features of the proposed architectures is the possibility of minimizing unwanted effects by the coordinator.

2.5.1 Distributed Control

In this thesis, distributed control refers to the use of several controllers working as a whole to achieve a control objective on a system. In this approach, the operation of the controller composed of n local controllers can be interpreted as follows: the symbol '$\rightarrow$' is read 'produce' and y is the output of the plant.

$$e \rightarrow \left\{ \begin{array}{c} u_1 \\ u_2 \\ \vdots \\ u_n \end{array} \right\} \rightarrow u \rightarrow y \tag{2.2}$$

This model represents an error signal being distributed among n controllers whose outputs are processed to generate the control command u. This control command is applied to the plant producing the output y. The intelligent system is used in this model to decide how the individual controllers must be combined to produce the control command. The motivation for using distributed control as opposed to nonlinear control is that simple controllers are readily available, more robust, and easier to operate than complicated nonlinear controllers. In general, it is easier to

identify a series of points of operation and assign local controllers than to design a nonlinear control based on a nonlinear model of the system. The following section, presents details of the method used for combining the controllers.

2.6 The Fuzzy Coordinator

A standard multi-input, multi-output fuzzy system is considered. It is represented by the following rule-based model where R^i represents the ith rule

$$R^i : \text{if } x_1 \text{ is } X_1^i \text{ and } \cdots x_p \text{ is } X_p^i \text{ then } c_1 \text{ is } C_1^i \text{ and } \cdots c_N \text{ is } C_N^i \tag{2.3}$$

where x_j is the jth input, and c_h the hth output, and X_j^i and C_h^i are the corresponding fuzzy sets. In this model, the input is the vector defined below by the *fuzzified* error and change of error. The fuzzification operator, $FUZZ$, is defined as a mapping of a quantity into the membership function values of a number of fuzzy sets defined over a range of values. This range of values is known in the fuzzy sets literature as universe of discourse. The input is given by

$$\begin{aligned} \mathbf{x} = FUZZ([\mathbf{e}\ \boldsymbol{\Delta}\mathbf{e}]^T) &= [x_1 \cdots x_p]^T \\ &= [E_1 \cdots E_{p/2} \cdots \Delta E_1 \cdots \Delta E_{p/2}]^T \in [0,1]^p \end{aligned} \tag{2.4}$$

where E_i and ΔE_h are the membership functions of e and Δe in the fuzzy sets of error and change of error defined as $\mathbf{E} = \{X_1, \cdots, X_p\} \in U_e$, where U_e is the universe of discourse of the input.

For example, if three fuzzy sets are considered for error, and three for change of error, with the following labels 'negative', 'zero', and 'positive' error and change of error, respectively, then the vector of input fuzzy sets is given by

$$\mathbf{x} = [x_1 \cdots x_6]^T = [NE\ \ ZE\ \ PE\ \ N\Delta E\ \ Z\Delta E\ \ P\Delta E]^T \in [0,1]^6 \tag{2.5}$$

where the letters N, Z, and P stand for the labels of the fuzzy sets.

The output of the model is the following vector of degrees of activation of each controller

$$\mathbf{c} = FUZZ(\mathbf{u}_n) = [c_1 \cdots c_n]^T \in [0,1]^n \tag{2.6}$$

where c_i is the membership function of controller i in the corresponding fuzzy set of output. The vector of output fuzzy sets is defined as $\mathbf{C} = \{C_1, \cdots, C_n\} \in U_c$, where U_c is the universe of discourse of the output, and $\mathbf{u}_n = [u_1 \cdots u_n]^T$ is the collection

of outputs of the local controllers. The variable c has been selected here to represent the degrees of activation λ. This has been done in order to keep the discussion in a more general context.

It is easy to see that the universe of discourse U_e as defined is isomorphic to the space $\{U_E, U_{\Delta E}\}$, where the two components are the universes of discourse of error and change of error. The number of input fuzzy sets is p, and the number of output fuzzy sets is n, (that is, the number of controllers).

The output fuzzy sets are defined as the degrees of activation of the local controllers. Thus, the degrees of activation are the fuzzified control outputs of each controller. The output fuzzy sets are not defined in the standard way. It is in these sets where the qualitative information necessary for the fuzzy system to make decisions is captured. It is here where the novelty of the proposed fuzzy system resides. The concept of dynamic fuzzy singletons is introduced (see next page) to calculate the output fuzzy sets online.

The input to the fuzzy model is $\mathbf{x} \in \mathbf{E}$ and the output is the vector of degrees of activation $\mathbf{c} \in \mathbf{C}$. The rule can be represented in a multivariable format as follows

$$R^i : \text{if } \mathbf{x} \text{ is } \mathbf{E}^i \text{ then } \mathbf{c} \text{ is } \mathbf{C}^i \tag{2.7}$$

where $\mathbf{E}^i$ and $\mathbf{C}^i$ are the fuzzy sets of input and output for rule i. The output fuzzy sets C_j are considered as fuzzy singletons (Zadeh [84]), which means that the fuzzy sets are composed of a single element. The single element being u_i, the output of controller i.

Thus, the output fuzzy singletons C_i are composed of a single element u_i with a membership value c_i. The membership value is the degree of activation λ_i of controller i.

The elements of the fuzzy singletons $\mathbf{C}$ are not fixed values since they depend on the output of the controllers. This makes the problem more complicated and a need arises for a method to encapsulate qualitative information for the calculation of the selection variables. The problem in hand is to design an inference method to calculate the selection variables and the final control signal. The following section, details the concept of dynamic fuzzy singletons.

2.6.1 Dynamic Fuzzy Singletons

A fuzzy singleton is a fuzzy set composed of one static element (Zadeh [84]). A dynamic fuzzy singleton is a fuzzy set composed of a single time-varying element.

This idea was developed with the objective of assigning a fuzzy set to each controller, where the degrees of activation are the membership values, and the fuzzy singletons are the values of the outputs of the controllers.

The problem is how to calculate the degrees of activation. The solution is a system capable of making decisions based on the input fuzzy sets, and capable of learning information about the control operations and requirements. This system must be designed from encoded qualitative information. Once the information has been encoded, the system must be able to provide the degrees of activation.

The intelligent system that has been chosen for this task is a logic processor (Pedrycz [48],[50]). The logic processor architecture and operation will be discussed in the following chapter. For now, the production of the control signal is addressed. This stage is known in the fuzzy control literature as defuzzification.

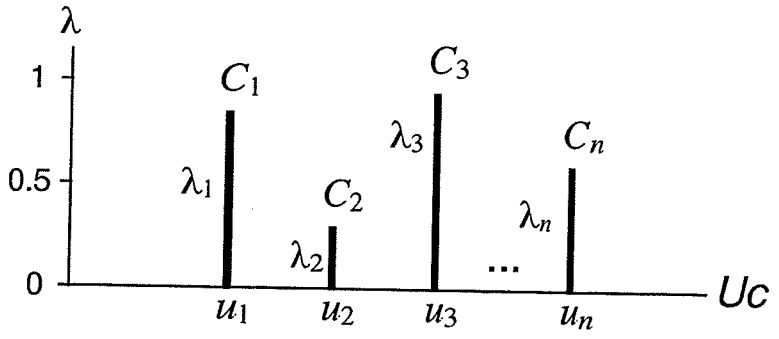

Figure 2.3: Dynamic Fuzzy Singletons.

Figure 2.3 represents the concept of dynamic fuzzy singletons. The fuzzy sets C_i are composed of a single element u_i, which is the output of controller i, with a membership value c_i equal to the degree of activation λ_i.

2.6.2 The Control Signal

A survey of standard defuzzification methods has been included in Appendix A. The center of gravity method has been chosen to produce the final control signal . The fuzzification stage is referred to in this thesis as *aggregation* whenever there is dynamic fuzzy singletons at the output.

Figure 2.4 represents the architecture of the overall fuzzy coordinator as a network. The aggregation of the control signals has been represented by a single-neuron

network, where the selection signals are the inputs with the control commands as weights.

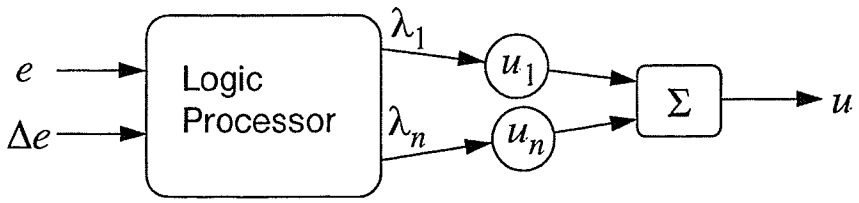

Figure 2.4: Network Representation.

The center of gravity method is applied to calculate the control command as follows.

$$u = \frac{\sum_{h=1}^{n} \lambda_h u_h}{\sum_{k=1}^{n} \lambda_k} \tag{2.8}$$

This equation represents the normalized weighted average of the control signals u, the weights being the degrees of activation λ (represented by c in Section 2.6).

Summary

In this chapter the architecture of the proposed controller has been presented. A natural use for an intelligent system as supervisor or coordinator was identified. In this way, the precision of standard controllers can be maintained while some qualitative knowledge can be encoded in the coordinator. The idea is to utilize the coordinator to activate the local controllers based on feedback measurements, in such as way that a wide range of operating conditions can be covered. The coordinator calculates degrees of activation of the local controllers, and these values are used to aggregate the responses of the controllers to produce the final control output.

The following chapter presents details on the production of the degrees of activation.

Chapter 3

Fuzzy Decision Making

In this chapter, the decision making process of the fuzzy coordinator is addressed. Logic processors are similar to neural networks (Simpson [66]), the difference being that in logic processors, algebraic operations from fuzzy logic theory are used (Simpson [67], [68], Pedrycz [48], [50]). Their structure can be visualized as a set of if-then fuzzy rules (Pedrycz [50]). The details of their operation are discussed in the following sections.

3.1 Logic Processors

The architecture of a logic processor is shown in Figure 3.1. It can be represented by a feedforward neural network with p inputs, n outputs, and a hidden layer of m units ($m \geq n$). This architecture has been considered to implement the fuzzy coordinator since the network can be trained to learn a complicated mapping as opposed to standard fuzzy logic techniques where training is not available.

The fuzzy coordinator is designed so that the decisions are made in terms of the fuzzified feedback signals, which are the error e and its change Δe. It has been explained that for the tracking problems under consideration, it seems natural to select two controllers, one to provide a short response time, and one to regulate or maintain the output at a reference point.

In this case, a decision table, such as the one shown in Figure 3.2, represents the desired reasoning. This figure represents a fuzzy logic-based partition of the Cartesian space of the input fuzzy sets. According to the selection of local controllers, the selection regions have been labeled u_1 and u_2, respectively. The shape of the shadowed region labeled u_2 represents a smooth transition between the regions. There

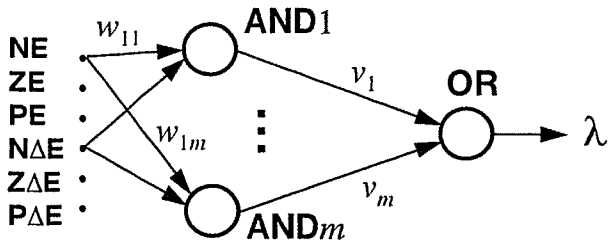

Figure 3.1: Logic Processor.

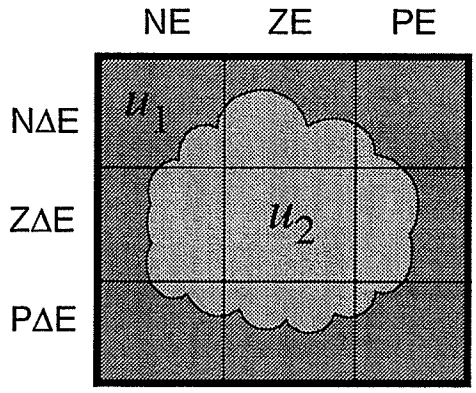

Figure 3.2: Fuzzy Partition.

is no clearly defined boundary between the regions, and this is precisely the idea of smooth transition, or fuzzy switching.

The fuzzy coordinator is, in fact, composed of a collection of multi-input/single-output logic processor units. One logic processor unit is needed per controller. The inputs are all the same and the outputs are the individual activation levels.

This discussion considers a p-input/single-output logic processor without loss of generality. Let the output of the ith AND neuron be z_i, $\mathbf{z}$ be the vector of all the outputs of the AND units, $\mathbf{w}_i$ and $\mathbf{v}$ be the vectors of connection weights, $\mathbf{x}$ be the vector of the p input fuzzy sets of e and Δe as defined previously, and λ is the output of the logic processor denoting the degree of activation of a local controller.

The AND and OR neurons are described by (Hirota [23], Pedrycz [50]):

$$\begin{array}{rcccl} z_i & = & \mathrm{AND}(\mathbf{x}, \mathbf{w}_i) & = & \mathrm{T}_{l=1}^{p}(x_l \, \mathrm{s} \, w_{il}) \\ \lambda & = & \mathrm{OR}(\mathbf{z}, \mathbf{v}) & = & \mathrm{S}_{l=1}^{m}(z_l \, \mathrm{t} \, v_l) \end{array} \tag{3.1}$$

where $i = 1, ...n$, and s and t are $s-$ and $t-$ *norms*, respectively[1] (Pedrycz [49]). One logic processor as in Equation 3.1 is designed to calculate the degree of activation for each local controller, and the collection of these constitutes the fuzzy coordinator.

The connections $\mathbf{w}_i$ and $\mathbf{v}$ are learned in supervised mode (Pedrycz [49]). The qualitative information is encoded into the logic processor during learning. Learning is accomplished by presenting a training data set to the network while adjusting the weights to produce the desired response. Thus, training is a numerical procedure that depends on the data set and the learning algorithm.

The data set is composed of values of error and change of error at the input, and selection variable at the output. The method of collecting the representative training set will be discussed in the following section. Notice that since the input to the network will be the fuzzy sets of e and Δe, λ in fact becomes a function $\lambda(e, \Delta e)$.

3.2 Encoding of Qualitative Knowledge

The technique presented in this section has been developed as a tool to capture the qualitative evaluation of the control performance. In the tracking problems under consideration, the goal is achieved by assigning a task to each individual controller. Each controller partially satisfies the goal, and the overall control problem is solved by an appropriate combination of all the responses. The specific control objectives in tracking problems are the following: (i)obtain the fastest response, (ii)obtain good regulation characteristics.

As a preliminary step, the number of local controllers must be chosen depending on the characteristics of the system. This can be determined by analyzing the qualitative properties of the individual controllers, the complexity of the plant, and the range of operation. As explained before, two controllers are enough in tracking problems.

The individual controllers must be designed separately. In general, the design of a regulator requires the selection of a control structure and the tuning of its

[1] The probabilistic sum $asb = a + b - ab$, and the product $atb = ab$ are considered in this thesis.

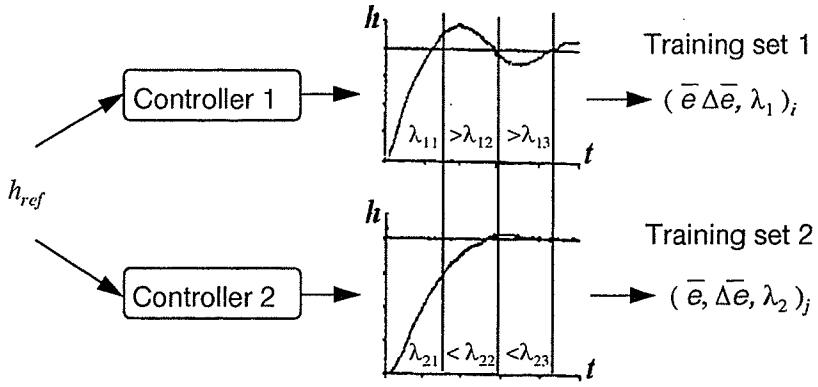

Figure 3.3: Windowing Technique.

parameters. In the case under consideration, the first controller can be tuned to achieve the response time goal, while the second can be tuned to provide good regulation. In both cases it is required that the two controllers guarantee stability of the closed-loop system.

Notice that a fast-response controller will generally have an overshoot. On the other hand, a good regulator generally provides a slow response. The labels *fast* and *slow* are used for the controllers in the present discussion. Although the mentioned controllers are, in a certain way, complementary, the design method will include qualitative observations that will assure that only the desired features of each one are incorporated into the final control signal.

Once the number, type, and parameters of the local controllers have been decided, the next step is to extract qualitative information about performance via a series of simulations or, if possible, experiments. These simulations/experiments should be selected to cover the wide range of operation of the plant for each controller. Step-response tests are appropriate in the evaluation and visualization of the qualitative properties of the response. The step-response is perhaps the most common way to evaluate the performance of a controller. It provides qualitative information about the speed of response, overshoot, and regulation, and it is a very good method to test the robustness of a system against disturbances.

The information is extracted from plots of the obtained responses in time, subdivided into k small windows, as shown in Figure 3.3. The plots should run for the same period of time and cover all the desired conditions of operation.

The designer then can assign degrees of preference, $\hat{\lambda}_{ij} \in [0, 1]$, to each controller i and window j, expressing a choice in the performance achieved in the window. This choice must be in agreement with the specific task for which the controller was designed. Thus, higher values of $\hat{\lambda}$ should indicate better performance for the specific region. This stage is very important in the overall design procedure; it is in this phase where the qualitative knowledge is being encoded.

The degrees of preference are assigned subjectively to reflect the designer's opinion. In the case of two or three local controllers, this can be easily done. As the number of controllers increases, it is more difficult to maintain consistency of the numerically expressed preferences.

Once the degrees of preference have been assigned, the next step is to design the membership functions of the fuzzy sets of error e and change of error Δe. The proper selection of the scale and number of fuzzy sets is very important. This step mainly relies on the expertise and good judgment of the designer. A good analysis on this topic is presented by Palm [47].

The scaling factors can be selected by identifying the smallest and yet significant error that could make a difference in the qualitative description of the response. This can be done simply by observation. Anything larger than this limit is viewed as a large error, while anything smaller is considered a small error. The selected value can be the parameter that defines the fuzzy sets, where all the fuzzy sets are defined based on the value of this parameter. For example, let the identified value be $\alpha/2$, for α as shown in Figure 3.4.

The regions of operation of the local controllers in terms of the fuzzy sets can now be viewed as in Figure 3.2, where U_1 is favored to operate for large values of error and change of error, while U_2 is chosen for small values. The ranges of the membership functions for error e and change of error Δe in the figure are represented by μE and $\mu\Delta$E, respectively.

The next step is to subdivide each window into a number of time intervals. Using these time intervals, the representative values of error and change of error can be statistically estimated for the window. This can be done by taking the average values. Next, the membership values for the average error and change of error are computed. These values, along with the assigned levels of preference, constitute a training data set to complete learning. In this way the qualitative information about performance is captured in the fuzzy coordinator. In the following section an overview of the selected learning algorithm is presented.

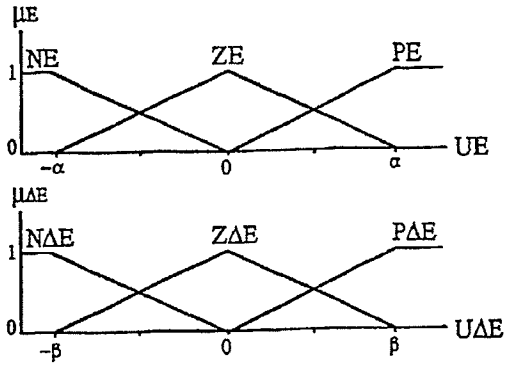

Figure 3.4: Fuzzy Sets of e and Δe.

3.2.1 Learning of the Connection Weights

Learning in neural networks and logic processors, in general, refers to the process of selecting the connection weights. The technique known as supervised learning is considered. The weights are calculated by minimizing a cost function based on a training set (Simpson [66], Pedrycz [50]). The training set is composed of input-output pairs (not necessarily scalars) representing the desired behaviour.

The algorithm presented by Hayashi [20] for a similar architecture has been considered. In general, a supervised learning algorithm is composed of a formula to calculate the increments, and another to update the weights. The formula to calculate the increments depends on the current values of the outputs and the training set. The idea is to apply an input that should provide a desired output, then modify the weights by a small amount based on the difference between the current and desired outputs, and run the system again to calculate a new output. The process is repeated until the algorithm reaches an acceptable minimum difference. Thus, every supervised learning algorithm based on weight increments is essentially the same. There is a slight difference on the calculation of the weight increment in different algorithms, but this is insignificant since the increment is still multiplied by a scaling parameter. This parameter is called the learning rate.

The weights initially have random values. An illustration of the formulas considers a network with no hidden layers. In this case the new weight is given by the following equation

$$w_i(k+1) = w_i(k) + \eta \Delta w_i(k) \tag{3.2}$$

where η, the learning rate, scales the increment $\Delta w_i(k)$, which is given by

$$\Delta w_i(k) = \sum_{h=1}^{H} (\lambda_h^2 - \lambda_h \lambda_{dh})(1 - \lambda_h) x_{hi} \tag{3.3}$$

where H is the order of the training set, λ_h is the output of the processor to the hth input training element x_{hi}, and λ_{dh} is the desired output. The idea is to minimize a cost function that serves as a learning index. The learning index is interpreted as an equality index of the form $Q = \sum(output \equiv target)$ (Pedrycz [50]). The cost function is given by

$$Q = \sum_{h=1}^{H} (\lambda_h - \lambda_{dh})^2. \tag{3.4}$$

For a logic processor with a hidden layer, the learning rule is divided into two parts, since there are two sets of weights. Referring to Figure 3.1 and considering only one output, the modified rule as presented by Hayashi [20] is given by

$$w_{ij}(k+1) = w_{ij}(k) + \eta \Delta w_{ij}(k) \tag{3.5}$$

for the first stage, where $\Delta w_{ij}(k)$ is given by

$$\Delta w_{ij}(k) = \sum_{h=1}^{H} (\lambda_h^2 - \lambda_h \lambda_{dh})(1 - \lambda_h) v_j z_{hj} (1 - z_{hj}) x_{hj} \tag{3.6}$$

and for the second

$$v_j(k+1) = v_j(k) + \eta \Delta v_j(k) \tag{3.7}$$

where $\Delta v_j(k)$ is given by

$$\Delta v_j(k) = \sum_{h=1}^{H} (\lambda_h^2 - \lambda_h \lambda_{dh})(1 - \lambda_h) z_{hj}. \tag{3.8}$$

These two rules are the learning algorithm considered in this thesis. In the following section, some issues on the algebraic construction of the logic processors are addressed.

3.3 Construction of Logic Processors

A logic processor is made of a series of elements, and for a large number of components, the construction of the terms at the output is a combinatorial problem. In order to analyze the static mapping in detail, the number and nature of the algebraic

terms introduced by the fuzzy logic operations must be determined. The processor can be easily implemented as a recursive algorithm in a computer; however, the algebraic structure and the implementation in analog electronics are not simple.

The structure of the logic processor was studied, and formulas were developed to assist in visualizing the complexity of the logic processor. This is useful in organizing the terms, and provides a mechanism to express the complex mapping in a simple expression.

A p-input/single-output processor is considered as in Equation 3.1. AND and OR units define the input/output map. Thus, the overall map consists of two parts. In this fashion, the following mapping is defined

$$L_P = S_m \circ A \tag{3.9}$$

where $L_P : X \rightarrow L$ is a mapping that represents the logic processor, $X \subset [0,1]^p$ is the input space, L is the output space, $A : X \rightarrow Z_m$ is a mapping that represents the AND nodes, $Z_m \subset [0,1]^m$ is the space of outputs of the AND nodes (equal to inputs to the OR node), $S_m : Z_m \rightarrow L$ is a mapping that represents the OR node, and $L \subset [0,1]$ is the output space of the OR node. This composition operator '$\circ$' used in the definition of the mapping should not be confused with the fuzzy composition. The symbol is the same but the context is different. The individual maps are studied in the following discussion, and after that, the overall logic processor is considered.

3.3.1 The AND Elements

The mapping $A : X \rightarrow Z_m$ is given by

$$z = A\mathbf{x} = \{\mathrm{T}_{l=1}^{p}(x_l \,\mathrm{s}\, w_{1l}), \cdots, \mathrm{T}_{l=1}^{p}(x_l \,\mathrm{s}\, w_{ml})\} \tag{3.10}$$

where $\mathbf{x} \in X$ is the input to the AND elements and $z = \{z_1, \cdots, z_m\} \in Z_m$ is the output with X and Z_m. After substituting the s and t norms, z_i becomes

$$z_i = (x_1 + w_{i1} - x_1 w_{i1})(x_2 + w_{i2} - x_2 w_{i2}) \cdots (x_p + w_{ip} - x_p w_{ip}). \tag{3.11}$$

Thus, each z_i is the product of p terms which are easily computed.

3.3.2 The OR Element

In the case of the OR element, the output is produced by a more complicated function. A procedure is presented that allows the visualization of the algebraic operations involved. This procedure is useful when an analog implementation is required.

In the case of a digital implementation, a recursive algorithm can easily be used to code the operation.

The mapping $S_m : Z_m \rightarrow L$ is described in Equation 3.1, and can be written as follows

$$\lambda \ = S_m \mathbf{z} = \ \mathrm{S}_{l=1}^{m} \left(z_l \, \mathrm{t} \, v_l \right) . \tag{3.12}$$

where $\lambda \in L$. After substituting the t norm, define $a_l = z_l v_l$ and $a = \{a_1, a_2, \cdots, a_m\}$. The order of a is m, which is equal to the number of AND elements in the logic processor.

The set a is well-ordered. It contains the input variables to the OR element multiplied by the weights of the OR layer, that is, $z_1 v_1, z_2 v_2, \cdots, z_m v_m$. The s-norm is recursevely applied to these elements, producing a final output which is the sum of many terms. Each term is a product of the elements of a.

Let C_j be the subset of a defined as follows

$$C_j = \{a_1, \cdots, a_j\} . \tag{3.13}$$

Let (jCq) be the sum of the products of the j elements of C_j taking q at a time. The following simple result will be useful in calculating the mapping S_m.

Lemma: For $j \geq 2$ and $j + 1 \geq q$ the following relation on the set a holds

$$(jCq) = ((j-1)Cq) + ((j-1)C(q-1))\, a_j. \tag{3.14}$$

$\Diamond$

Proof: The first term on the right-hand side provides the sum of products of the elements of a up to the $(j-1)$th element. Each of these terms is made of the product of q elements, which is precisely the size of the required terms in (jCq). There is a new element in (jCq) that is not present in the the first term in the right-hand side, that is, a_j. The missing terms are then constructed by multiplying the products of the first $j-1$ elements of a taking $q-1$ at a time by the new element, as in the second term in the right-hand side. The terms introduced in this way have the right number of elements, and they are the products missing. $\square$

The output of the OR element is calculated in the following Proposition.

Proposition 1 *The OR element of the Logic Processor with m units in the hidden layer, with $m \geq 2$, and a single output is described by the mapping*

$$S_m = \sum_{q=1}^{m} (-1)^{q-1} (mCq) \tag{3.15}$$

$\Diamond$

Proof: The principle of mathematical induction is used in this proof. The first step is to show that the proposition is true for $m = 2$. Consider the following relation

$$S_2 = \sum_{q=1}^{2} (-1)^{q-1} (2Cq) = (2C_1) - (2C_2). \tag{3.16}$$

and the definition of (jCq). The equation is equivalent to $S_2 = a_1 + a_2 - a_1 a_2$, which is as expected from Equation 3.1.

The next step is to show the assumption that if the proposition is true for S_j, it implies that it is also true for S_{j+1}. Since S_j represents the fuzzy OR operation of the j inputs, as in the next equation

$$S_j = \mathrm{S}_{l=1}^{j} a_l \tag{3.17}$$

the relation $S_{j+1} = S_j \,\mathrm{s}\, a_{j+1}$ must hold. Assume that $Sj = \sum_{q=1}^{j} (-1)^{q-1} (jCq)$, then

$$S_{j+1} = \left(\sum_{q=1}^{j} (-1)^{q-1} (jCq) \right) \,\mathrm{s}\, a_{j+1} \tag{3.18}$$

which yields $S_{j+1} = S_j + a_{j+1} - S_j a_{j+1}$. Substituting S_j, expanding the sum, and rearranging the terms yields

$$\begin{aligned} S_{j+1} = \; & ((jC1) - (jC2) + (jC3) - (jC4) \cdots + (-1)^{j-1} (jCj))(1 - a_{j+1}) \\ & + a_{j+1}. \end{aligned} \tag{3.19}$$

This can be written as

$$\begin{aligned} S_{j+1} = \; & ((jC1) + a_{j+1}) - ((jC2) + (jC1) a_{j+1}) \\ & + ((jC3) + (jC2) a_{j+1}) + \cdots \\ & + \left((-1)^{j-1} (jCj) - (-1)^{(j-1)-1} \, (jC(j-1)) \, a_{j+1} \right) \\ & - (-1)^{j-1} (jCj) a_{j+1}. \end{aligned} \tag{3.20}$$

Using the Lemma on each of the terms between parentheses gives the following relation

$$\begin{aligned} S_{j+1} = \quad & ((j+1)C1) - ((j+1)C2) + ((j+1)C3) - \cdots \\ & +(-1)^{j-1}\left((j+1)Cj\right) + (-1)^{j}\left((j+1)C(j+1)\right) \end{aligned} \tag{3.21}$$

where in the last term, the sign has been modified using the relation $(-1)^{j-1} = (-1)^{j}$, and also $((jCj)a_{j+1} = (j+1)C(j+1))$, which can easily be proven. The sum can be written as

$$S_{j+1} = \left(\sum_{q=1}^{j+1}(-1)^{q-1}\left((j+1)Cq\right)\right) \tag{3.22}$$

which proves the proposition by the principle of mathematical induction. □

This proposition provides an ordered algebraic description of the elements generated by the OR unit. It is also useful in determining the number of terms of S_m, as follows.

Corollary: Let $|S_m|$ be the number of terms in S_m. $|S_m|$ is equal to the sum of the number of products of the m elements of a taking $q \in \{1, 2, \cdots, m\}$ at a time:

$$|S_m| = \sum_{q=1}^{m} mCq \tag{3.23}$$

where $mCq = \dfrac{m!}{q!(m-q)!}$ denotes the number of combinations of m elements taking q at a time. ◇

Proof: The proof follows straight from the proposition since (mCq) in Equation 3.15 represents the sum of products of the m elements of a in groups of q elements. Thus the size of (mCq) is given by

$$|(mCq)| = mCq = \frac{m!}{q!(m-q)!} \tag{3.24}$$

and the corollary is proven by adding the size of all the terms. □

Example: For a logic processor with 8 units in the hidden layer, the number of product terms at the output is $|S_8| = 8 + 28 + 56 + 70 + 56 + 28 + 8 + 8 = 262$.

The Lemma introduced is useful in constructing a logic processor, since it provides a recursive relation to calculate terms of the sum, which can simplify the implementation by reducing the number of variables. All the terms can be built from the following two basic formulas and the Lemma

$$(2C2) = a_1 a_2 \qquad\qquad (2C1) = a_1 + a_2 \tag{3.25}$$

Summary

The chapter has presented a discussion of the coordinator based on a logic processor. The logic processor is the decision-making part of the fuzzy coordinator. A procedure has been presented to capture the qualitative information that is used to train the logic processor. The algebraic structure has also been considered and formulas have been introduced to calculate the number of components for an analog implementation of the logic processor.

The following chapter presents a dynamic analysis of the overall architecture for control problems.

Chapter 4

Dynamic Analysis

In this chapter, a dynamic analysis of the hierarchical control structure is presented. The analysis is based on the assumption that the average value of the selection variables varies at a slower rate than the response of the controllers in a given interval of time. This assumption allows the consideration that the selection variables are constant for a short interval of time. This is not very restrictive since the time increment of the logic processor can be chosen so that it is larger than the time increment of the controllers. Notice that the fuzzy coordinator is a static mapping that depends on error and change of error, and when the system is in operation in the boundary cases (that is, when only one controller is active), the selection variables are constant.

The architecture presented makes the coordination system operate as a smooth switch, in which at least one controller is active at a given time. It is also assumed that the coordinator has learned the criteria for switching, and that all the conditions of operation have been considered. Intuitively, it can be noted that the stability of the equilibria of the overall system is guaranteed as long as the individual controllers provide stable closed-loop equilibria. These assumptions simplify the analysis while still considering the dynamics of the hierarchical control structure.

The following sections introduce some notation and definitions, followed by an analysis of the problem of stability using operators.

4.1 Operator Representation

Consider a plant G_p that belongs to a family of nonlinear systems of the form

$$G_p : \begin{array}{l} \dot{x} = f(x) + g(x)u \\ y = Cx \end{array} \tag{4.1}$$

where $f(x)$ and $g(x)$ are continuous smooth functions of the state $x \in R$, C is a constant, u is the input and $y \in R$ is the output. A single dimensional case is considered without loss of generality, since the analysis can easily be adapted to higher dimensions of the state, with the condition that the system be observable for a single-input single-output system with $x \in R^n$. It is assumed that all the signals are of the appropriate dimension.

Let y_r be the desired value of the output and define the error as

$$e = y - y_r \in E \subset R \tag{4.2}$$

where E is the space of error signals. The closed-loop system can be represented by an operator

$$G : U \rightarrow E \tag{4.3}$$

where the input is given by

$$u \in U \subset R \tag{4.4}$$

and U is the space of control signals. The operator G is defined by

$$y = G(u) = e + y_r. \tag{4.5}$$

Let $Y \subset R$ be the space of plant output signals. The operator G is not necessarily linear nor continuous. In the case where there are no disturbances and zero initial conditions, G is an operator from U *onto* E, since all the errors are produced by control signals, and there is no element of E that is not produced by an element of U. However, the general case is considered where G is an operator from U *into* E. It is also considered that the elements of E and U are discrete values which approximate the continuous functions. The time variable t is used as an index.

Similarly, the controller can be represented by an operator

$$F : E \rightarrow U \tag{4.6}$$

defined by $u = F(e)$. Thus, F is an operator from E *onto* U. Without loss of generality it is considered that the elements of U are sequences of discrete values. If

the controllers are continuous they can be discretized. One of the advantages of using discretization is that the controllers can be implemented on a digital computer. Also, in the case of a linear system with a singular state feedback matrix, the discretized version is always nonsingular (Francis [17]). This topic will be addressed in more detail in the following chapter.

The closed-loop system can be represented by a composition of operators as follows

$$e_{new} = G \circ F \ e_{old} \tag{4.7}$$

which represents the calculated value of the new error after the current error is processed by the controller, and a control command has been processed by the plant. The operator $\circ$ is the standard composition of operator theory (Naylor [45]). Define the mapping

$$T : E \rightarrow E \tag{4.8}$$

as the composition of the mappings G and F as follows

$$T = G \circ F. \tag{4.9}$$

Therefore the closed-loop system is represented as a mapping

$$T : E \rightarrow U \rightarrow E \tag{4.10}$$

which defines the recursive equation of the error, given by

$$e = G(F(u)) - y_r. \tag{4.11}$$

The mapping T is typically nonlinear.

In general, and similarly to the discussion of the fuzzy coordinator, the above discussion should consider a space E of elements $[e \ \dot{e}]^T \in R^2$. This space is the state of the closed-loop system. The notation is the same as above and all the mappings remain the same with the appropriate dimensionality. In this representation, e is the instantaneous error and $\dot{e}$ its derivative, also interpreted as its rate of change. In addition, e and $\dot{e}$ can be vectors, depending on the dimension of the output y. In the following general discussion, the state of the closed-loop system is considered as $\mathbf{x} = [e \ \dot{e}]^T \in R^{2n}$. In the specific case of the fuzzy coordinator a single dimensional case will be considered. The distinction between the notation used in the analysis of the dynamic system, and that used in the description of the fuzzy coordinator should be noted.

4.2 Equilibria of Dynamic Systems

For an n-dimensional dynamic system $\dot{\mathbf{x}} = f(\mathbf{x})$, for $\mathbf{x} \in R^{2n}$, an *equilibrium point* $\mathbf{x}_o$ is defined as the value of $\mathbf{x}$ such that $f(\mathbf{x}_o) = 0$. Without loss of generality, it is considered that the equilibrium is the origin of the state plane. If the equilibrium is not the origin, then a change of coordinates can always be used to translate the equilibrium point to the origin. Part of the analysis must be concentrated in proving that $\mathbf{x}_o$ is an *isolated* equilibrium, or better that it is *unique*. This is necessary in order to satisfy the requirements of stability in the sense of Lyapunov (Vidyasagar [78]).

In the following discussion, the *Euclidean norm* $\|\cdot\|$ in R^s is considered, where s is the dimension of the space. The dimensionality will be clear from the context. Next, some concepts of stability are presented. These concepts have been taken from Vidyasagar [78].

4.2.1 Stability

If the equilibrium $\mathbf{x}_o$ of a system is stable, then for each $\epsilon > 0$, there exists $\delta > 0$ such that

$$\|\mathbf{x}(0) - \mathbf{x}_o\| < \delta \rightarrow \|\mathbf{x}(t) - \mathbf{x}_o\| < \epsilon. \tag{4.12}$$

As mentioned before, it can be considered that $\mathbf{x}_o = 0$, and thus the stability condition can be written as $\|\mathbf{x}(0)\| < \delta \rightarrow \|\mathbf{x}(t)\| < \epsilon$. In other words, if it is desired that the error trajectory remains inside a hypersphere of radius ϵ in $\mathbf{x}$, then a hypersphere of radius δ must be found to start the system from within.

In terms of operators, stability of a system can be represented with respect to the stability properties of its equilibria. The stability condition implies that the error decreases or remains the same at every instance of time. Thus, the error is a sequence satisfying the following property

$$\|T_s\mathbf{x}\| \leq \alpha_s\|\mathbf{x}\| \tag{4.13}$$

for $\alpha_s \in (0, 1]$ at each instant. This parameter can only tend to zero asymptotically, but it cannot reach that value, except when the initial condition is the equilibrium. The oscillatory response occurs when $\alpha_s = 1$. The system can be described as an iterative system, where the interval between values of $\mathbf{x}$ is very small. In this context, the continuous time system is approximated by a discrete system, but this approximation is used only for analysis purposes.

4.2.2 Asymptotic Stability

If the equilibrium $\mathbf{x}_o$ of a system is asymptotically stable, then it is stable, isolated, and there exists $\delta > 0$ such that

$$\|\mathbf{x}(0) - \mathbf{x}_o\| < \delta \rightarrow \lim_{t \rightarrow \infty} \|\mathbf{x}(t) - \mathbf{x}_o\| = 0. \tag{4.14}$$

Considering $\mathbf{x}_o = 0$, the limit can be written as $\|\mathbf{x}(t)\| \rightarrow 0$. In other words, the error trajectory that starts inside a hypersphere of radius δ in $\mathbf{x}$ tends to the equilibrium point. Therefore $\mathbf{x}_o$ can be interpreted as a *local attractor.*

In terms of operators, asymptotic stability of a system can be represented in terms of the stability properties of its equilibrium. The asymptotic stability condition implies that the error decreases at every point in time. Thus the error signal is a sequence satisfying the following property

$$\|T_a\,\mathbf{x}\| \leq \alpha_a \|\mathbf{x}\| \tag{4.15}$$

for $\alpha_a \in (0, 1)$ at each instant. An operator with the properties of T_a is known as a *contraction mapping* (Khalil [31]). It is well known that a contraction mapping has one *fixed point* (Khalil [31]), in this case $\mathbf{x}_o$, such that $T\mathbf{x}_o = \mathbf{x}_o$. Therefore the closed-loop state approaches the equilibrium with each iteration of the asymptotically stable closed-loop system.

4.3 Qualitative Control in Tracking Problems

It has been explained that for the tracking problems under consideration two controllers are enough to satisfy the control objective. The analysis will be carried out using two controllers with the properties described below. The analysis is easily extended to n controllers, since multidimensional operators are being used.

Consider that there exist two controllers, T_1 and T_2, acting on a plant G_p such that T_1 provides the shortest response time to drive the closed-loop state to a neighborhood of the equilibrium, and T_2 provides the response with the best regulation characteristics. The controllers will be referred to as fast and slow. The second controller is slow enough so that short disturbances do not have time to affect the response. As explained previously, these qualitative labels can be assigned based on observation of the response after a few step-response simulations. Thus, for this analysis it is assumed that the two controllers have been properly tuned.

The fuzzy coordinator is a static system with the capability of combining the responses of individual controllers. The two controllers considered have the properties that the equilibrium of the closed-loop system of T_1 is stable and that of T_2 is asymptotically stable.

The fast controller T_1 should dominate the control task for large error and large changes of error, and the regulator T_2 should be used when the error and its change are small. Let $\lambda_1, \lambda_2 \in [0, 1]$ be the selection variable for T_1 and T_2, respectively. The two boundary cases are then $\lambda_1 = 1, \lambda_2 = 0$ for operation far from the equilibrium point (for large e and Δe), and $\lambda_1 = 0, \lambda_2 = 1$ around the equilibrium point (for small e and Δe).

The single-dimensional case is considered where e and $u(e)$ represent scalar functions. The adaptation to higher dimensions is almost immediate. It is assumed that all the signals are of the appropriate dimension.

Next, the switching of the controllers is studied according to the selection variables assigned by the fuzzy coordinator. This operation is called fuzzy switching and it is considered in the following section.

4.4 Fuzzy Switching

During operation of the hierarchical controller each controller receives the same error signals as input and calculates a control signal. The fuzzy system weights the two control signals and generates the signal that is applied to the plant. Since the control signal applied to the plant is not always exactly that calculated by one of the controllers, the error signal that is generated is different from the expected signal by each controller if its control signal had been applied directly. Thus, the overall dynamics of the closed-loop system depend on the dynamics of the individual controllers and on the properties of the switching. It will be shown that under proper design conditions, the response of the system under the fuzzy coordinator is determined by the qualitative properties of the individual controllers.

It can be observed that no additional dynamic properties are introduced by the fuzzy system since its effect is to scale the control signals, and the overall fuzzy system can be described by a static mapping. Thus, the effect of the fuzzy system in the control loop is the same as that of a series of gains that affect each of the local controllers. Thus, the fuzzy system can be considered as one gain per controller.

4.4.1 The Hierarchical System

Let the hierarchical system be composed of a plant Gp as in Equation 4.1, a fast stable controller T_1, and an asymptotically stable regulator T_2. Consider that the control signal u is produced using the center of gravity method as follows

$$u = \frac{\lambda_1 u_1 + \lambda_2 u_2}{\lambda_1 + \lambda_2}. \tag{4.16}$$

The weights λ_i determine the contribution of the control signals in the final control action. During the operation, these weights have a fixed value calculated by the fuzzy system. Once the values are set, the dynamic properties of the system are determined by the plant and the controllers. These values of λ_i are calculated by a static mapping and they will be considered as constants.

4.4.2 The Local Controllers

According to the representation of the stability concepts in operators, the controllers satisfy the following relations

$$\begin{aligned} \|T_1\, e\| &\leq \alpha_1 \|e\| \\ \|T_2\, e\| &\leq \alpha_2 \|e\| \end{aligned} \tag{4.17}$$

for

$$\begin{aligned} \alpha_1 &\in (0, 1] \\ \alpha_2 &\in (0, 1). \end{aligned} \tag{4.18}$$

Considering the smallest upper bound that satisfies the inequality, then $\|T_1\| = \inf \alpha_1$ and $\|T_2\| = \inf \alpha_2$. In this case, α_1 and α_2 are the smallest values that satisfy the relations, and therefore

$$\begin{aligned} \|T_1\, e\| &= \alpha_1 \\ \|T_2\, e\| &= \alpha_2 \end{aligned} \tag{4.19}$$

The individual closed-loop systems are obtained by replacing u by u_1 and u_2 in the equation of the plant. After substituting these values, the following equations are obtained

$$\begin{aligned} \dot{x} &= f(x) + g(x)u_1 = \dot{x}_1 & \qquad \dot{x} &= f(x) + g(x)u_2 = \dot{x}_2 \\ y_1 &= Cx_1 & \qquad y_2 &= Cx_2 \end{aligned} \tag{4.20}$$

These equations represent the contribution to the output by each individual controller. The dynamics of the plant with u_1 and u_2 produces the signals $\dot{x}_1$ and $\dot{x}_2$, respectively. The first integral of these signals produce the corresponding outputs y_1 and y_2. Next, the closed-loop system is analyzed.

4.4.3 The Closed-Loop System

The overall closed-loop system is obtained by substituting the weighted control signal into the plant as follows

$$\dot{x} = f(x) + g(x)\left(\frac{\lambda_1 u_1 + \lambda_2 u_2}{\lambda_1 + \lambda_2}\right). \tag{4.21}$$

This equation can be rearranged into

$$\dot{x} = \frac{\lambda_1}{\lambda_1 + \lambda_2}\left(f(x) + g(x)u_1\right) + \frac{\lambda_2}{\lambda_1 + \lambda_2}\left(f(x) + g(x)u_2\right) \tag{4.22}$$

which is equivalent to

$$\dot{x} = \frac{\lambda_1}{\lambda_1 + \lambda_2}\left(\dot{x}_1\right) + \frac{\lambda_2}{\lambda_1 + \lambda_2}\left(\dot{x}_2\right). \tag{4.23}$$

The output is given by the first integral of the state equation, and since the first integral of x_i produces y_i the following equation is obtained

$$y = \frac{\lambda_1}{\lambda_1 + \lambda_2} y_1 + \frac{\lambda_2}{\lambda_1 + \lambda_2} y_2. \tag{4.24}$$

This equation can be written as

$$y = C\left(\frac{\lambda_1}{\lambda_1 + \lambda_2} x_1 + \frac{\lambda_2}{\lambda_1 + \lambda_2} x_2\right) \tag{4.25}$$

where x_i is the state that each individual loop would produce. This result can also be obtained by the principle of superposition, since y is linear with respect to x.

4.4.4 The Error Signal

The new error signal is defined as $e = y - y_r$, and substituting Equation 4.25 yields

$$e = \frac{\lambda_1}{\lambda_1 + \lambda_2}\left(y_1 - y_r\right) + \frac{\lambda_2}{\lambda_1 + \lambda_2}\left(y_2 - y_r\right). \tag{4.26}$$

Let $e_i = y_i - y_r$, where e_i is the error produced by controller i acting on the plant Gp. In other words, if controller i is acting alone, then $e = e_i$. Notice that the error is in

fact a combination of the individual errors as if the individual systems were acting alone. This is as expected since each controller calculates its *recommended* signal for each value of the current state. Therefore the error equation is given by

$$e = \frac{\lambda_1}{\lambda_1 + \lambda_2} e_1 + \frac{\lambda_2}{\lambda_1 + \lambda_2} e_2. \tag{4.27}$$

In terms of operators, the feedback system can be described by the above equation and the following

$$\begin{aligned} e_1 &= T_1\, e \\ e_2 &= T_2\, e \end{aligned} \tag{4.28}$$

The following section addresses stability of the overall control system.

4.5 Stability of the Switching Controller

It is shown in this section that the stability of the overall controller depends on the stability of the individual controllers.

4.5.1 The Switching Operator

From the above equations, the overall closed-loop system can be represented by a switching operator $T_u : E \rightarrow E$ with the following property

$$T_u e_i = T_i e \quad \text{for } e \in E_i \tag{4.29}$$

where E_i is the space of error for controller i acting independently, since the switching does not affect the operation of a controller if the other one is not present. Notice that, in fact, this operator is acting on the Cartesian product of the error spaces

$$T_u : E_1 \times E_2 \rightarrow E. \tag{4.30}$$

The application of this operator to Equation 4.27 results in the following expression

$$T_u e = T_u \left(\frac{\lambda_1}{\lambda_1 + \lambda_2} e_1 + \frac{\lambda_2}{\lambda_1 + \lambda_2} e_2 \right) \tag{4.31}$$

which yields

$$T_u e = \frac{\lambda_1}{\lambda_1 + \lambda_2} T_1 e + \frac{\lambda_2}{\lambda_1 + \lambda_2} T_2 e. \tag{4.32}$$

The following proposition summarizes the stability analysis.

Proposition 2 *The control structure as described above is stable.* $\diamondsuit$

Proof: Taking the norm of both sides and using the triangle inequality yields

$$\|T_u e\| \leq \frac{\lambda_1}{\lambda_1 + \lambda_2}\|T_1 e\| + \frac{\lambda_2}{\lambda_1 + \lambda_2}\|T_2 e\|. \tag{4.33}$$

Using the stability properties of the operators T_i the following expression is obtained

$$\|T_u e\| \leq \frac{\lambda_1}{\lambda_1 + \lambda_2}\alpha_1\|e\| + \frac{\lambda_2}{\lambda_1 + \lambda_2}\alpha_2\|e\| \tag{4.34}$$

which can be rearranged into

$$\|T_u e\| \leq \left(\frac{\lambda_1\alpha_1 + \lambda_2\alpha_2}{\lambda_1 + \lambda_2}\right)\|e\|. \tag{4.35}$$

Given the nature of α_i, the norm of the switching operator is given by

$$\|T_u\| = \frac{\lambda_1\|T_1\| + \lambda_2\|T_2\|}{\lambda_1 + \lambda_2} \tag{4.36}$$

which states that the dynamics of the system under the fuzzy coordinator is determined by the dynamics of the individual controllers. Therefore, the stability of the overall system can be represented by

$$\|T_u e\| \leq \beta\|e\| \tag{4.37}$$

where $\beta = \|T_u\| \in [0, 1]$, which implies that the system is stable. $\square$

Notice that the boundary case of large error corresponds to a fast response $\beta < \alpha_1$, and that of small error, corresponds to good regulation $\beta < \alpha_2$. Thus, the extremes are precisely the properties of the individual controllers.

4.5.2 Quality of Response

By assigning the minima and maxima of λ_1 and λ_2 in the boundary cases to values strictly less than 1 and greater than 0, it can be guaranteed by a proper choice of these values that the response of T_u will have the speed of T_1 and the regulation of T_2. At the extremes, only one controller is in operation. The qualitative knowledge and the number of controllers were selected to cover the whole range of operation; thus, there is no deterioration of the performance as long as the design specifications are followed.

At the extremes, the hierarchical architecture behaves like a single controller, and thus, the performance can be predicted. At these points, the stability of the system is guaranteed by the stability of each controller.

The critical point of operation would occur when the controllers are applied with the same degree of activation. This does not mean that the levels of the control signals are the same. It means that with half of the level of the available signals the system can be driven to the equilibrium. The response of the system is then the fastest with the best regulation possible, satisfying the qualitative performance index.

Summary

In this chapter, the dynamic properties of the proposed architecture were studies. Conditions for stability of the overall controller have been established. The analysis is based on the assumption that the rate of change of the average output of the logic processors is less than the dynamics of the controllers. In this way, the coordinator can be approximated by a series of gains affecting each of the local controllers. The operator representation of the local controllers and of the stability concepts facilitated the analysis of the system. The stability of the overall controller depends on the stability of the local controllers. This is as expected, since the fuzzy coordinator implements a static mapping between the error signals and the selection variables.

The desired properties of the individual controllers have been established and in the following chapter some practical aspects are considered in an overview of PID and optimal controllers.

Chapter 5

Local Controllers

This chapter presents details on the local control algorithms that have been selected for this thesis. Two control methodologies are discussed: PID and optimal controllers. The PID control algorithm will be used in the application examples of the hierarchical control structure, and an optimal control will be applied in the example on distributed modeling.

5.1 Overview of PID Controllers

The proportional-integral-derivative (PID) controller is one of the most popular controllers available. A brief discussion is presented on the basic (continuous-time) and the practical (discrete-time) PID versions.

5.1.1 Continuous PID

PID controllers are very common control algorithms in applications. The basic PID is described by (Astrom [5]):

$$u = K_P e + K_I \int_o^t e d\tau + K_D \dot{e} \tag{5.1}$$

where K_P, K_I and K_D are the proportional, integral and derivative gain matrices respectively, e is the feedback error signal, and u the control signal. In general, e is a vector and the gain matrices are not restricted to being square. It is considered in this thesis that e is of the same dimension as u. The scalar case is considered, and the results can easily be extended to the multivariable case.

5.1.2 Discrete PID

The discrete version of the PID controller is used more often in implementations and simulations. It includes many improvements over the basic version (for details on the construction of the equations refer to Astrom [5]). The control law is represented by

$$w = K\,[b\,h_{ref} - h] + \left[I_{(t-1)} + \frac{K\,\Delta t}{T_i}e\right] + \left(\frac{T_D}{T_D + M\,\Delta t}\right)\left[D_{(t-1)} - \frac{K}{M}\Delta e\right] \tag{5.2}$$

where $b \in [0,1]$ is the set point weight factor and M is the maximum derivative gain. The integral and derivative terms are updated as follows

$$\begin{aligned} I_{(t)} &= I_{(t-1)} + (u - w)\frac{\Delta t}{T_T} \\ D_{(t)} &= \frac{T_D}{T_D + Mh}D_{(t-1)} - \frac{K T_D M}{T_D + Mh}e \end{aligned} \tag{5.3}$$

where T_T is the tracking constant and Δt is the sampling period. In applications there is a time delay and this is also considered in the model. The output before the delay is given by

$$z = \left\{ \begin{array}{ll} u_{min} & w < u_{min} \\ w & u_{min} \leq w \leq u_{max} \\ u_{max} & w > u_{max} \end{array} \right\} \tag{5.4}$$

and the control command is

$$u = z_{(t-\delta)} \tag{5.5}$$

where δ is a time delay, and u_{max} and u_{min} are the maximum and minimum values of the control output. This produces the final time-delayed control signal.

It is considered that the time derivative of error represents the change of error. This is represented by $\dot{e} = \Delta e$.

5.1.3 The Equivalent State-Dependent PID Controller

Let the PID controller be represented by an operator L, and the logic processor by the corresponding λ. These two are functions over the same domain and while L is a numerical function, λ is a logical one. The general control law can be written as

$$u = \frac{\lambda_1(e,\dot{e})u_1(e,\dot{e}) + \lambda_2(e,\dot{e})u_2(e,\dot{e}) + \cdots + \lambda_n(e,\dot{e})u_n(e,\dot{e})}{\lambda_1(e,\dot{e}) + \lambda_2(e,\dot{e}) + \cdots + \lambda_n(e,\dot{e})}. \tag{5.6}$$

After substituting the PID algorithms u_1 to u_n, the following expression is obtained

$$\begin{aligned} u = \; & \{\lambda_1(e,\dot{e})\left[K_{P1}e + K_{I1}\int_o^t e(\tau)d\tau + K_{D1}\dot{e}\right] + \\ & \lambda_2(e,\dot{e})\left[K_{P2}e + K_{I2}\int_o^t e(\tau)d\tau + K_{D2}\dot{e}\right] + \cdots + \\ & \lambda_n(e,\dot{e})\left[K_{Pn}e + K_{In}\int_o^t e(\tau)d\tau + K_{Dn}\dot{e}\right]\}/ \\ & \{\lambda_1(e,\dot{e}) + \lambda_2(e,\dot{e}) + \cdots + \lambda_n(e,\dot{e})\}. \end{aligned} \tag{5.7}$$

Grouping the P, I, and D terms, the control law can be rewritten as

$$\begin{aligned} u = \; & \left\{\frac{\lambda_1(e,\dot{e})K_{P1} + \lambda_2(e,\dot{e})K_{P2} + \cdots + \lambda_n(e,\dot{e})K_{Pn}}{\lambda_{tot}}\right\}e + \\ & \left\{\frac{\lambda_1(e,\dot{e})K_{I1} + \lambda_2(e,\dot{e})K_{I2} + \cdots + \lambda_n(e,\dot{e})K_{In}}{\lambda_{tot}}\right\}\int_o^t e(\tau)d\tau + \\ & \left\{\frac{\lambda_1(e,\dot{e})K_{D1} + \lambda_2(e,\dot{c})K_{D2} + \cdots + \lambda_n(e,\dot{e})K_{Dn}}{\lambda_{tot}}\right\}\dot{e} \end{aligned} \tag{5.8}$$

where $\lambda_{tot} = \sum_{i=1}^{n} \lambda_i(e,\dot{e})$. It is required that $\lambda_{tot} > 0$ for all e and $\dot{e}$.

Therefore, the overall controller can be treated as a state-dependent PID controller with the gains scheduled according to

$$\begin{aligned} K_P(e,\dot{c}) &= \frac{\sum_{i=1}^{n}\lambda_i(e,\dot{e})K_{Pi}}{\sum_{i=1}^{n}\lambda_i(e,\dot{e})} \\ K_I(e,\dot{e}) &= \frac{\sum_{i=1}^{n}\lambda_i(e,\dot{e})K_{Ii}}{\sum_{i=1}^{n}\lambda_i(e,\dot{e})} \\ K_D(e,\dot{c}) &= \frac{\sum_{i=1}^{n}\lambda_i(e,\dot{e})K_{Di}}{\sum_{i=1}^{n}\lambda_i(e,\dot{e})}. \end{aligned} \tag{5.9}$$

Hence, the fuzzy switching architecture is equivalent to a PID controller with variable coefficients. The design method for the overall architecture provides a technique to choose the parameters according to the desired operation of the system. This can be a difficult task in general, and the reader is referred to Astrom [5] for a in-depth discussion on PID controllers with variable gains.

5.2 Overview of Optimal Control

The technique known as optimal control is based on the minimization of the energy spent by a system by means of optimizing the control signal. This overview is based on the work presented by Francis [17].

Some of the advantages of discrete-time systems over continuous-time have already been mentioned. The discretization procedure of a linear system is presented, followed by an overview of discrete optimal control. This control method will be used later in the application example in the distributed modeling section. The procedure presented here was implemented for this thesis as a Matlab [42] command.

5.2.1 Discretization

Consider a continuous-time linear system given by the following equation:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\,\mathbf{x}(t) + B\,v(t) \qquad & \mathbf{x}(0) = \mathbf{x}^o \in R^n \\ y(t) &= C\,\mathbf{x}(t) \end{aligned} \tag{5.10}$$

where $\mathbf{x}$ is the state and y the output. Assure that the matrices A, B and C are of adequate dimensions. The discretized systems become

$$\begin{aligned} \dot{\mathbf{z}}(k+1) &= A_d\,\mathbf{z}(k) + B_d\,v_d(k) \qquad & \mathbf{z}(0) = \mathbf{z}^o \in R^n \\ y_d(k+1) &= C_d\,\mathbf{z}(k) \end{aligned} \tag{5.11}$$

where

$$\begin{aligned} A_d &= e^{hA} \\ B_d &= \int_0^h e^{\tau A} d\tau \; B \\ C_d &= C. \end{aligned} \tag{5.12}$$

In this representation, h is the sampling period, the new output $y_d = SAMP(y)$ is the output of a sampler whose input is y, and the new discrete control is related to the continuous version by $u = HOLD(v)$. The functions $SAMP$ and $HOLD$ represent a sampler and a zero-order hold.

In optimal control, the control signal is generated according to a feedback law that minimizes the cost function

$$J = \sum_{k=0}^{\infty} \left(\mathbf{z}(k)^T\, Q\, \mathbf{z}(k) + v_d(k)^T\, R\, v_d(k) \right) \tag{5.13}$$

where $Q \geq 0$ and $R > 0$. The optimal control obtained is called LQR (linear quadratic regulator). The control signal is given by

$$v_d(k) = F\mathbf{z}(k) \tag{5.14}$$

where F is calculated as described in the following procedure. The procedure guarantees that J is minimized. For a detailed description of the algorithm see Francis [17].

- Calculate the matrix (the matrix H is called simplectic)

$$H = \begin{bmatrix} I & B_d R^{-1} B_d^T \\ 0 & A_d^T \end{bmatrix}^{-1} \begin{bmatrix} A & 0 \\ -Q & I \end{bmatrix}. \tag{5.15}$$

- Assume that H has no eigenvalues on the unit circle. Calculate the matrix T whose columns are the generalized eigenvectors of H corresponding to the stable eigenvalues. Partition T as follows by selecting blocks of the same size

$$T = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}. \tag{5.16}$$

- Calculate $X = T_2 T_1^{-1}$.

- The feedback is then given by

$$F = -\left(I + B_d^T X\, B_d\right)^{-1} B_d^T X\, A_d. \tag{5.17}$$

The general procedure is to choose Q and R, then compute F, simulate, and choose new Q and R to modify the response. This algorithm will be used in an application example of distributed modeling at the end of this thesis.

In this chapter an overview of two control techniques has been presented. The continuous and discrete versions of a PID have been discussed, as well as discretization of linear time-varying systems and discrete optimal control. These techniques will be used in the following chapters.

The following chapter considers an application of the hierarchical control structure in the control of the level of a liquid in a tank.

Chapter 6

Control of a Tank

In this chapter, an application of the proposed architecture to the level control of a fluid in a tank is considered. The system under control is the water tank displayed in Figure 6.1. This system is similar to that presented in Astrom [4]. The nonlinear model of the system is presented and it will be shown that a tracking control objective can be achieved with the fuzzy switching control architecture using two PID controllers.

6.1 Dynamic Model of the Plant

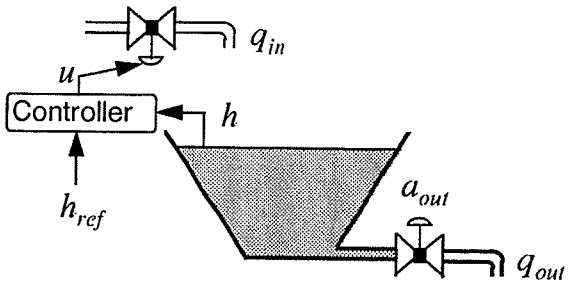

Figure 6.1: Water-Level Control System.

The system is described by the following differential equation:

$$\frac{d}{dt}h = \frac{(q_{in} - q_{out})}{area} - K_a area \tag{6.1}$$

where

$$q_{in} = q_{max} cval$$
$$q_{out} = a_{out}\sqrt{2gmax(h,\epsilon)} \tag{6.2}$$

and the constant $\epsilon > 0$ represents a small constant consumption factor. A value of $\epsilon = 10^{-3}$ is assigned for this example. The control signal is saturated at $u \in [0, 100\%]$, and it is applied to the input valve, and the output is the level h. The functions *area* and *cval* represent the shape of the tank and the saturation in the outlet valve. They are given by

$$cval = \left\{ \begin{array}{ll} 0 & u < 0 \\ u & 0 \leq u \leq 1 \\ 1 & u > 1 \end{array} \right\} \tag{6.3}$$

$$area = \phi(h)$$

where $q_{max} = 1m^3/sec, g = 9.81m/sec^2, K_a$ is an evaporation factor, and $a_{out} \in [0, 1]$ is the area of the outlet valve. It is considered as a random variable and interpreted as a disturbance at the output. Furthermore, a transportation delay is considered that, for simplicity, is modeled as a part of the discrete PID local controllers. Two local controllers are used. A time delay of 2 sampling periods ($\delta = 2$) in the valve of the tank has been considered.

The closed-loop system is nonlinear given the nonlinearities of the plant, the saturations, and the time delay of the controllers. This system is difficult to control using standard techniques (Astrom [5]), and it will be shown that a tracking control objective can be achieved with the proposed control architecture.

Three profiles of the tank are considered, as shown in Figure 6.2.

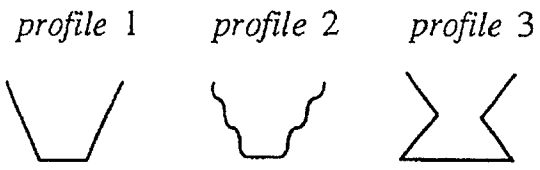

Figure 6.2: Profiles of the Water Tank.

These profiles have been modeled as follows:

$$\begin{aligned}
&\text{Profile 1:} && area = a_{min} + \frac{h}{\sqrt{3}} \\
&\text{Profile 2:} && area = a_{min} + \frac{h}{\sqrt{3} + \frac{\sin(5h)}{5}} \\
&\text{Profile 3:} && area = \left\{ \begin{array}{ll} -\frac{5h}{3} & h < 3 \\ h - 1 & h \geq 3 \end{array} \right\}
\end{aligned} \tag{6.4}$$

where $a_{min} = 2m$, $a_{max} = 7m$, and $h_{max} = 8m$.

6.1.1 The Control Objective

The control objective is to follow a reference signal as closely as possible, with a fast transient and minimum oscillations. This is clearly a tracking problem. The objectives translate into two control specifications: a fast response (ideal tracking) and minimum oscillations (good regulation). Initially, the goals are achieved independently, and then a switching controller is designed to accomplish the tracking objective.

The controller will be designed in discrete-time, since most of the industrial controllers are digital. The error and change of error are defined as follows:

$$\begin{aligned}
e(t) &= h_{ref}(t) - h(t) \\
\Delta e(t) &= e(t) - e(t-1)
\end{aligned} \tag{6.5}$$

where t will be considered as a discrete variable. The local control algorithms will be implemented as PID controllers as presented in the previous chapter. The following section addresses the problem of stability.

6.2 Stability of the System

At this point it will be shown that a single controller is capable of satisfying a single control objective. As mentioned before, a dynamic change in the parameters in a PID does not guarantee that the quality of response remains unchanged, and therefore, stability is not automatically guaranteed. It must be shown that a single PID controller is at least stable in order to be able to use it in the switching controller. As usually required in stability analysis, regulation and tracking properties of the closed-loop system will be addressed separately. The following analysis is general,

valid for n continuous-time PID controllers. It is applicable to a discrete-time implementation for a sufficiently high sampling rate. There are several methods to calculate the sampling rate, but in a more realistic approach, this value is fixed by the hardware available.

6.2.1 Stability of a Single PID

The following proposition summarizes the stability analysis of a single PID in closed-loop with the tank.

Proposition 3 Consider the plant described in Equation 6.1, in closed-loop with a continuous-time PID controller with constant gains as in Equation 5.1. The point $[e\ \dot{e}]^T = [0\ 0]^T$ is a stable equilibrium of the closed-loop system and h converges to a stationary reference. $\diamondsuit$

Proof:

The closed-loop system is obtained by letting $q_{in} = u$, as represented in the following equation:

$$area\ \dot{h} + q_{out} + K_a area^2 = K_P\ e + K_I \int_o^t e\ d\tau + K_D\ \dot{e}. \tag{6.6}$$

Consider $u = 0$, and notice that $\dot{h} = -q_{out}/area - K_a area$ so $h \to 0$, at least by the evaporation effect.

Now, consider the derivative of q_{out} with respect to time, $\dot{q}_{out} = a_m \dot{h} = a(\dot{h}_{ref} - \dot{e})$. where $a_m = a_{out}\ g/\sqrt{2g \max(h, \epsilon)} \geq 0$. Notice that the changes in $\dot{h}$ will be reflected to a greater extent in q_{out} when a_m is maximum. Let $a(h) = \max(a_m)$, with $a_{out} = 1$ and $\max(h, 0) = h_{max}$. Thus, a is a constant. Also notice that $\lim_{h \to \infty} a(h) = 0$.

The integral term in the PID controller increments the order of the closed-loop system, which is represented as a first time-derivative of Equation 6.6. After substituting the expression for $\dot{q}_{out}$, this equation can be written as

$$(K_D + area)\ddot{e} + (K_P + a)\dot{e} + K_I e = area\ \ddot{h}_{ref} + a\ \dot{h}_{ref}. \tag{6.7}$$

Let the state vector of the system be $\mathbf{x} = [e\ \dot{e}]^T$. Rewriting the above, the state equation of the system is obtained. This is given by

$$\dot{\mathbf{x}} = \frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} \dot{e} \\ \left(area\ddot{h}_{ref} + a\dot{h}_{ref} - (K_P + a)\dot{e} - K_I e\right)/(K_D + area) \end{bmatrix}. \tag{6.8}$$

It is clear that the origin $[e\ \dot{e}]^T = [0\ 0]^T$ is an equilibrium when h_{ref} =constant.

Regulation: Let $\gamma = K_I/(K_D + area) > 0$, and consider the following positive definite matrix

$$P = \begin{bmatrix} \gamma & 0 \\ 0 & 1 \end{bmatrix} \tag{6.9}$$

and the Lyapunov candidate function

$$V(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T P \mathbf{x}. \tag{6.10}$$

The derivative of V with respect to time is given by

$$\dot{V}(\mathbf{x}) = -\mathbf{x} R \mathbf{x} \tag{6.11}$$

where

$$R = \begin{bmatrix} -area\ \ddot{h}_{ref} + a\ \dot{h}_{ref}/(K_D + area) & 0 \\ 0 & (K_P + a)/(K_D + area) \end{bmatrix}. \tag{6.12}$$

If h_{ref} =constant, then $\dot{V}(\mathbf{x}) \leq 0$, and it follows that the origin is a stable equilibrium of Equation 6.8 in the sense of Lyapunov.

Convergence: It is reasonable to assume that h_{ref} is bounded, and for this system h and $\dot{h}$, are bounded. This can be expressed by

$$h_{ref},\ h,\ \dot{h} \in L_\infty \Rightarrow e,\ \dot{e} \in L_\infty \Rightarrow \ddot{e} \in L_\infty \tag{6.13}$$

where L_2 and L_∞ are the vector spaces of continuous functions with bounded 2- and ∞-norm, respectively. It is shown next that h converges to a stationary point.

Since $\dot{V}(\mathbf{x}(t)) \leq 0$, then

$$V(\mathbf{x}(0)) \geq V(\mathbf{x}(t)) \geq \frac{\gamma}{2}e^2 + \frac{1}{2}\dot{e}^2 \geq \frac{1}{2}\dot{e}^2 \tag{6.14}$$

and it follows that

$$\int_o^t V(\mathbf{x}(0))\,d\tau \geq \frac{1}{2}\int_o^t \dot{e}^2\,d\tau. \tag{6.15}$$

The expression in the left hand side is finite, given the properties of V. Now let $t \to \infty$ in the above expression and notice that

$$\int_o^\infty \dot{e}^2\,d\tau = \|\dot{e}\|_2 < \infty \tag{6.16}$$

which by definition implies that $\dot{e} \in L_2$. Recalling that if $\ddot{e} \in L_\infty$ and $\dot{e} \in L_2$, then $\lim_{t\to\infty} \dot{e} = 0$ (Vidyasagar [78]), and it can be concluded that $\dot{h} \to 0$, and since h_{ref} is constant, then h converges to a stationary point. □

6.2.2 Stability of the General Architecture

The following proposition addresses the stability of the overall system.

Proposition 4 Consider the plant described in Equation 6.1 in closed-loop with the hierarchical system with n PID controllers of the type described in Equation 5.1. Then, the origin of the phase plane $[e\,\dot{e}]^T = [0\,0]^T$ is an asymptotically stable equilibrium of the closed-loop system. In addition, the overall controller is equivalent to a PID controller with state-dependent gains $K_P(e,\dot{e}) > 0$, $K_I(e,\dot{e}) > 0$, and $K_D(e,\dot{e}) > 0$. $\diamondsuit$

Proof:

It is straight forward to show that the origin of the state plane is an equilibrium of the closed-loop system. Consider the following Lyapunov candidate function:

$$V(e,\dot{e}) = \frac{1}{2}\dot{e}^2 + \int_o^e \varepsilon K_m(e,\dot{e})d\varepsilon + \int_o^{\dot{e}} \xi K_n(e,\dot{e})d\xi \geq 0 \tag{6.17}$$

where

$$\begin{aligned} K_m(e,\dot{e}) &= \frac{K_I(e,\dot{e})}{K_D(e,\dot{e}) + area} > 0 \\ K_n(e,\dot{e}) &= \frac{K_P(e,\dot{e}) + a}{K_D(e,\dot{e}) + area} > 0. \end{aligned} \tag{6.18}$$

It can be seen that V is a positive definite function, since the first term is positive and the integrals can be interpreted as potential energy factors.

The time derivative of V along the state trajectories of the closed-loop system is given by $\dot{V}(e,\dot{e}) = -\dot{e} \leq 0$. This is a negative, semidefinite function, and therefore, stability of the origin is directly concluded. As in the case of a single PID, it can be shown that e is a bounded function, while $\dot{e}$ is a square integrable and bounded function.

To show asymptotic stability, LaSalle's Theorem[1] (Vidyasagar [78]) is used, noticing that the closed-loop is an autonomous system. Consider the following region

$$\Omega = \left\{ \begin{bmatrix} e \\ \dot{e} \end{bmatrix} \text{ for } \dot{V}(e,\dot{e}) = 0 \right\} = \left\{ \begin{bmatrix} e \\ 0 \end{bmatrix} \in R^2 \right\} \tag{6.19}$$

[1] LaSalle's Theorem states that for an autonomous equation $\dot{x}(t) = f(x(t)), x(0) \in R^n$, $\forall t \geq 0$, with the origin as the unique equilibrium, and assuming that a Lyapunov candidate function exists such that $V(x) \leq 0, \forall x \in R^n$, then the origin is asymptotically stable if the equation $f(s) = 0$,for $s \in \Omega = \left\{ s \in R^n \mid \dot{V}(s) = 0 \right\}$, has a unique invariant $s = 0$.

and notice that the only invariant is $e = 0$. Invoking LaSalle's Theorem, it can be concluded that the origin is asymptotically stable. □

In the following section, some practical aspects are presented about the selection of the PID controllers required for the present application example.

6.3 Design of the Hierarchical Control Structure

This section details the design of the PID controllers, followed by the design of the hierarchical structure.

6.3.1 Selection of the Parameters of the PID Controllers

Table 6.1 provides a qualitative summary of the design requirements and relationships between the parameters of PID controllers. The discrete PID algorithm chosen are explained in Chapter 5.

Goal	Increase	Decrease	Side Effects
Increase speed	K (try to keep it as small as posible)	T_I T_D	oscillations smoothness increases overshoot increases
Decrease overshoot	K, T_I, T_D	T_T, b	speed decreases noise sensitivity increases
Decrease noise Sensitivity	T_T (try to keep it as small as possible), b	M	overshoot increases

Table 6.1: Intuitive Design of a PID.

For the present application, it was determined that two local controllers were enough to satisfy the tracking control problem. The parameters were determined by simulations. The parameters of the first controller PID_1 were selected to achieve the fastest possible response. The second controller PID_2 was tuned to achieve good regulation, which requires that the controller react slowly to changes in the errors. Step response graphs are shown in Figure 6.3.

In order to compare the responses, an additional PID controller was designed. This PID_3 is optimal in the sense of Ziegler-Nichols [5]. The standard Ziegler-Nichols method uses a step-response of the system to adjust the parameters of the controller based on the readings from the graph of the response, according to the following

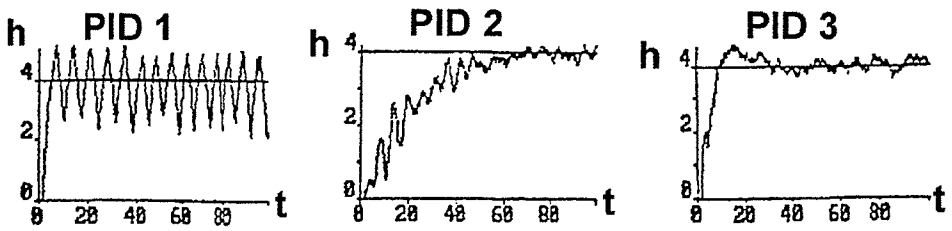

Figure 6.3: Response with the PID controllers.

criteria [5]:

$$K_P = \frac{1.2}{a} \qquad K_I = \frac{K_P}{2L} \qquad T_D = \frac{K_P L}{2} \tag{6.20}$$

where $a = K_s L/T$. These values are determined from the graph of the response as shown in Figure 6.4.

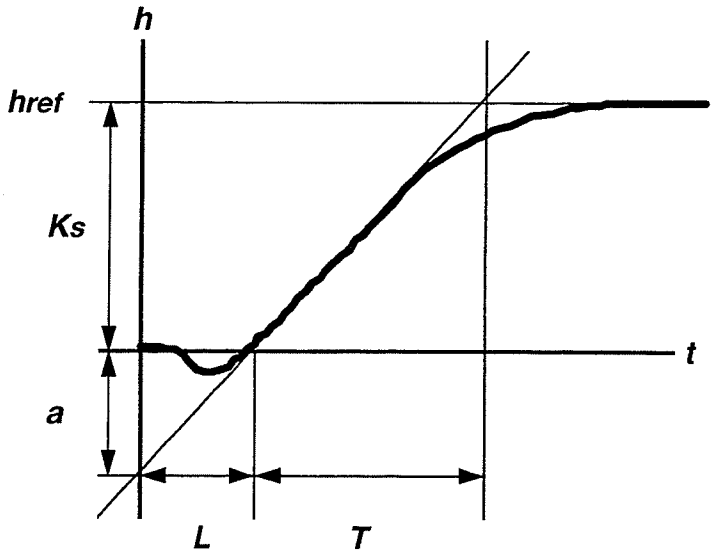

Figure 6.4: Ziegler-Nichols Criterion.

This criterion is applicable only when $0.1 < L/T < 1$. In the simulation, the values read from the step response are $K_s = 8, L = 1$, and $T = 9$. The response of the system is presented in Figure 6.3. Table 6.2 shows the final values of the parameters of the three PID controllers.

Parameter	PID 1 (fast)	PID 2 (slow)	PID 3 (Z–N)
b	1	0	1
K	15	1	1.35
M	10	0	10
T_D	10	10	0.5
T_I	0.1	15	2
T_T	0.1	1	0.5
u_{min}	0	0	0
u_{max}	1	1	1
Δt	0.1	0.1	0.1

Table 6.2: Parameters of the Controllers.

6.3.2 Design of the Fuzzy Coordinator

Two logic processors are required to build the coordinator. They were designed as outlined in Chapter 2. Using the windowing technique, step-response experiments were carried out for each PID individually. The step size used was 4 m, with $h(0) = 8$ m, for a time interval from 0 to 100 sec. The plots were subdivided into windows of 5 seconds each, and samples were taken at every second.

The membership functions for error and change of error were triangular as shown in Figure 3.4. The selected values of the parameters were $a = 0.5$ and $b = 0.05$. The average membership values were calculated for each fuzzy set of error and its change for the first 10 windows. The hidden layers of each logic processor consisted of eight elements, and the selected learning rate was $\eta = 0.2$. The training data set was collected from two step-response graphs, one for each controller. The simulation results are presented in the following section.

6.4 Simulation Results

The simulation results presented show the performance of the controllers under different conditions. Initially, a fixed output flow was considered with a value of $a_{out} = 0.05$ m^3/sec. The simulations were repeated with a_{out} random with a rectangular distribution over $[0, 0.125]$. Figures 6.5 to 6.7 and 6.8 show the results for the three profiles of the tank. Notice that the response of the coordinator is fast with a good regulation. The performance is better than that of the Ziegler-Nichols PID.

A typical state trajectory produced by the fuzzy hierarchical controller is shown in Figure 6.9. Notice that this state plane resembles the characteristic of well-tuned PID controllers. The state trajectory move from an initial condition to the origin following a direct path.

Notice also that the quality of response of the fuzzy hierarchical controller is better than that of the optimal PID_3, and that the best features of both PID_1 and PID_2 are combined. It is fast with good regulation, and the side effects caused by each controller are minimized.

The dynamic switching between PID_1 and PID_2 is shown in Figure 6.10. The figure shows the values of λ_1 and λ_2 in the time interval from 6 to 8 *sec*. It also shown the values of the output h and the reference h_{ref}. Notice that as h approaches h_{ref}, the tendency of λ_1 is to decrease, while λ_2 tends to increase. This means that PID_1 is becoming less active while PID_2 is taking over the control task.

In the following section, some measurements of the performance are presented.

6.4.1 Performance Measurements

A quantitative characterization of response can be represented in terms of the following performance indices

$$Q = \sum_{t=1}^{\infty} e^2(t)$$
$$Q' = \sum_{t=1}^{\infty} t|e(t)|. \tag{6.21}$$

The first performance index summarizes a total error in discrete time moments $e(t)$. The second includes the time factor, which is useful in measuring the error that remains nonzero when a long period of time has passed. The values of these performance indices for the simulations are shown in Tables 6.3 and 6.4. Notice that the

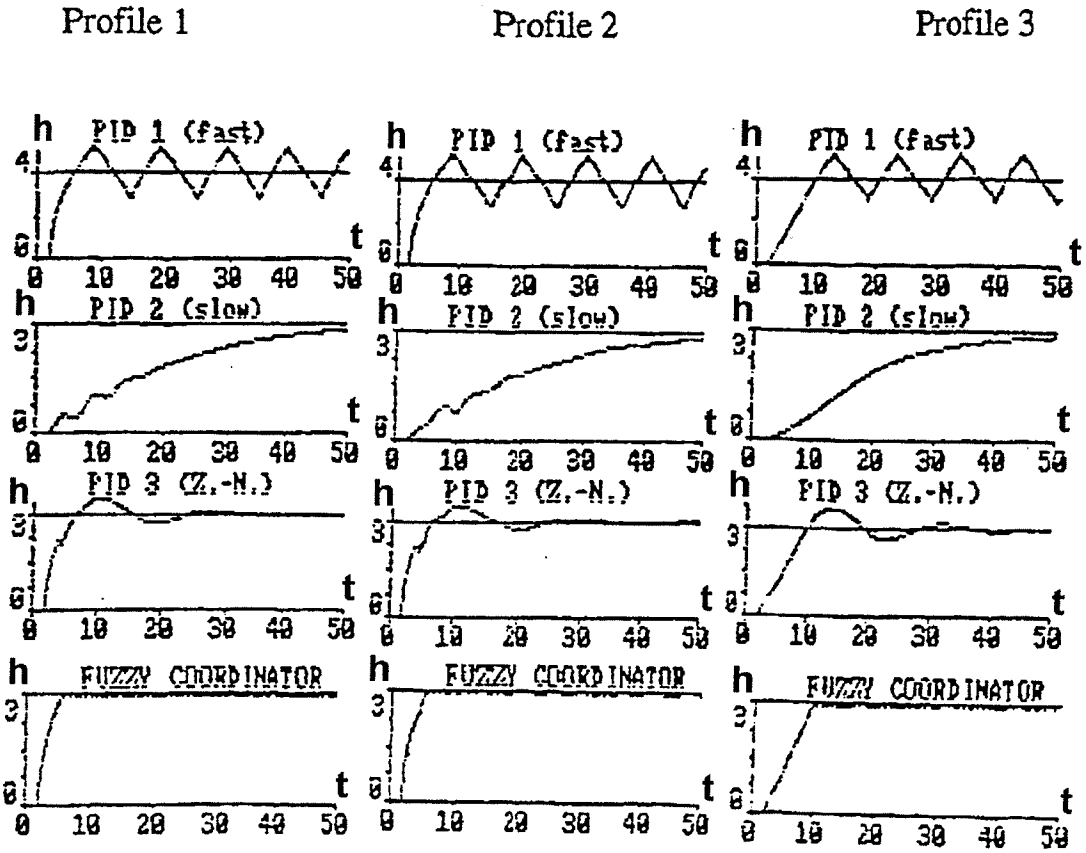

Figure 6.5: Simulations: $a_{out} = 0.05m^3/sec$, Input: Step.

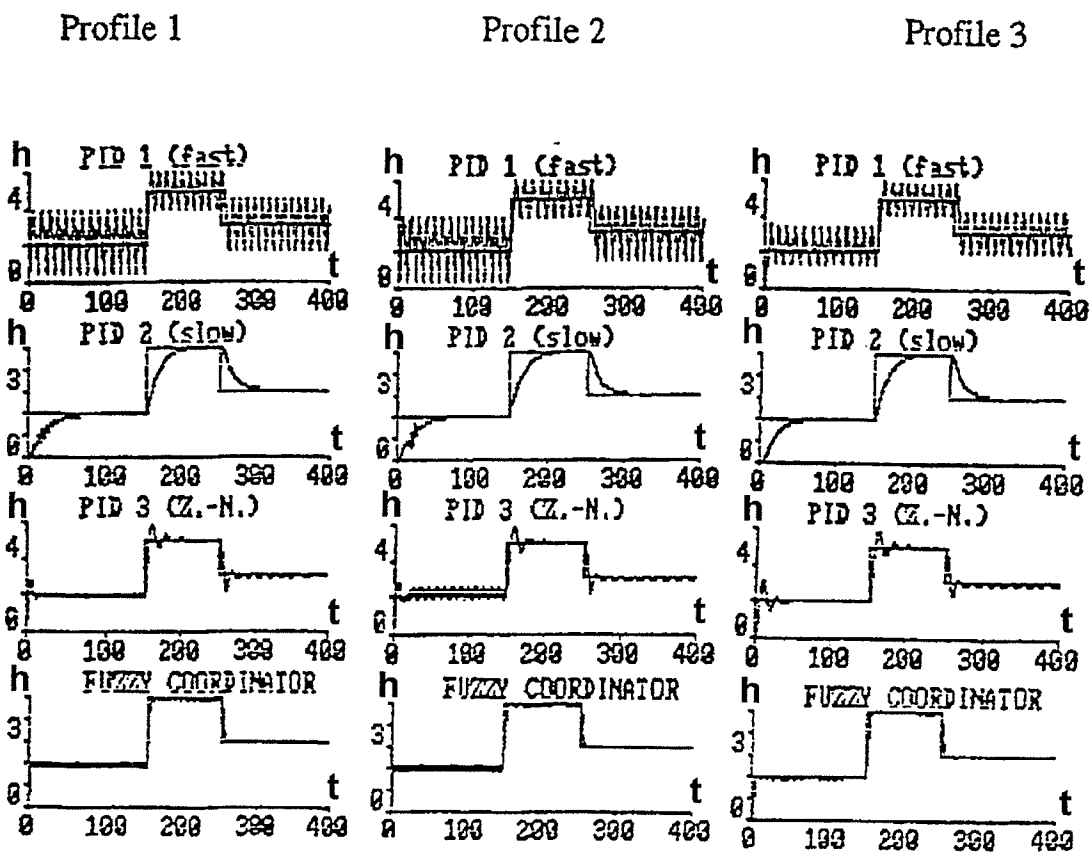

Figure 6.6: Simulations: $a_{out} = 0.05m^3/scc$, Input: Steps.

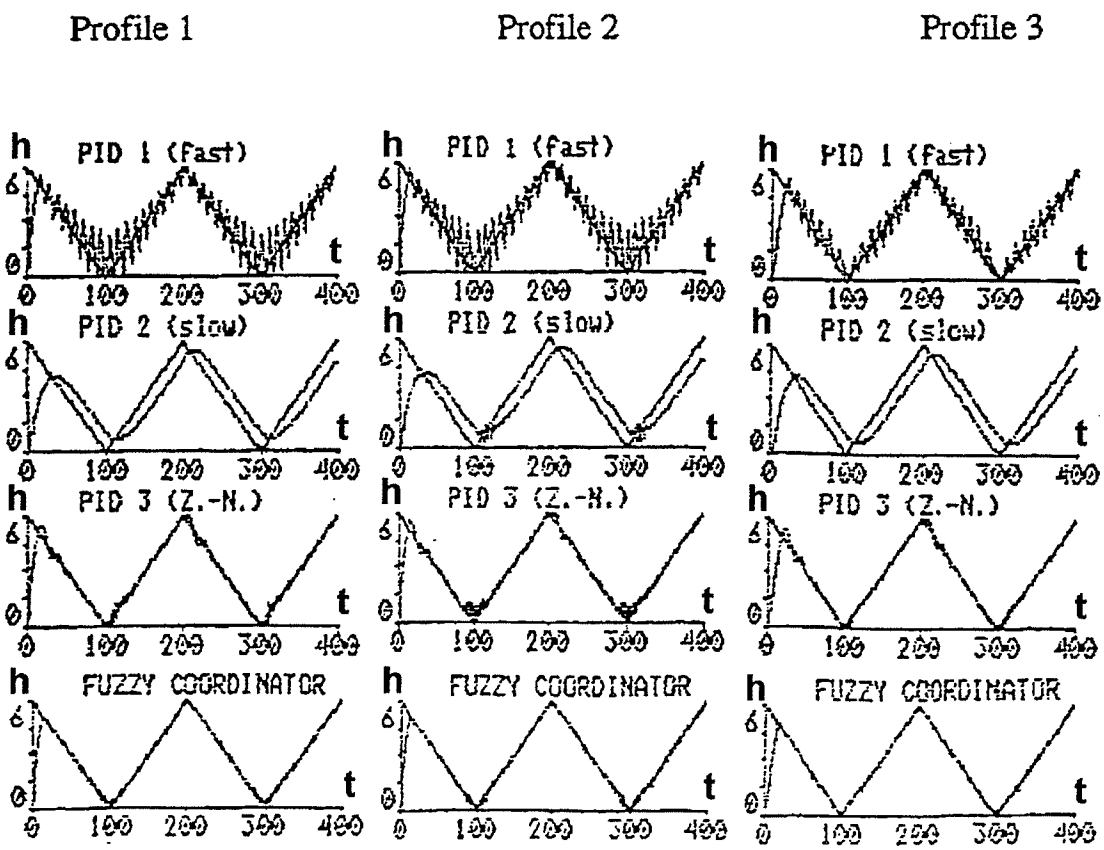

Figure 6.7: Simulations: $a_{out} = 0.05m^3/sec$, Input: Triangular.

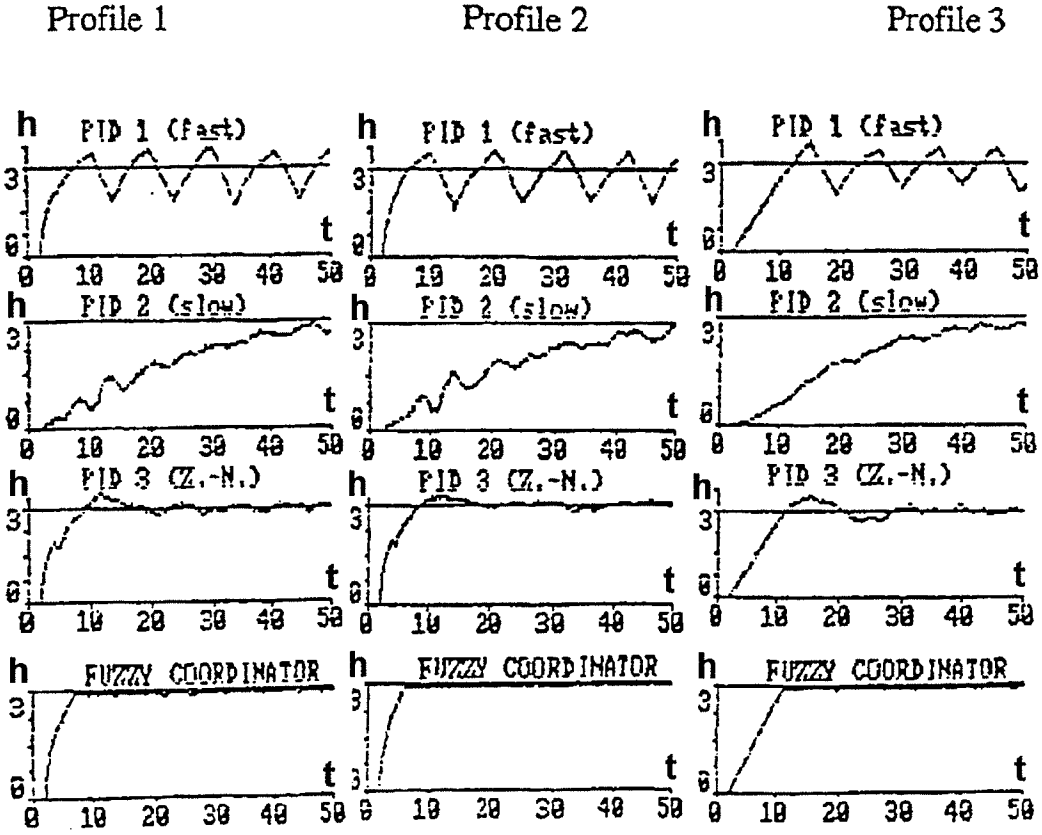

Figure 6.8: Simulations: a_{out} Random, Input: Step.

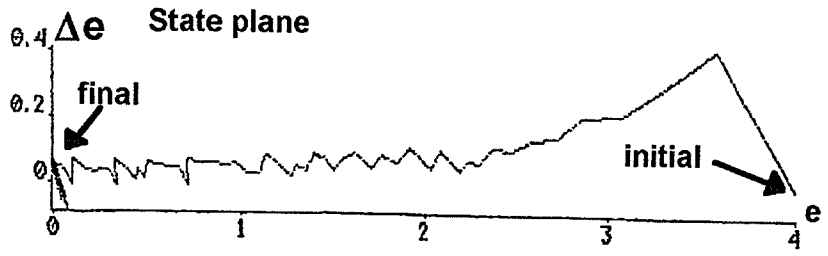

Figure 6.9: Typical State Trajectory.

entries corresponding to the fuzzy hierarchical controller are smaller than the entries of the other controllers.

Input	Profile	PID_1, Q	PID_2, Q	PID_3, Q	Fuzzy, Q	PID_1, Q'	PID_2, Q'	PID_3, Q'	F. C., Q'
Step	1	646.22	1843.5	471.7	439.08	7511.9	11115	1431.4	903.66
	2	665.83	1846.2	479.98	447.38	7478.3	11057	1479.3	788.79
	3	929.81	2050.9	823.44	775.0	7298.9	10384	2900.3	1396.1
Sine	1	690.36	2449.9	538.55	492.94	7716.2	12321	1940.3	697.24
	2	699.18	2450.9	553.7	503.62	7716.7	12219	2218.1	724.77
	3	1122.2	2759.4	1008.7	965.87	1981.1	12820	3566.4	1584.3
Steps	1	3418.6	1847.7	615.11	338.67	577020	188520	790950	504460
	2	3505.2	1847.4	668.31	332.3	581180	188500	989060	520490
	3	2210.9	1951.2	782.77	453.5	473870	187040	839280	529180
Triangular	1	5684.4	10656	2771.4	2624.4	555710	940510	74167	51311
	2	5687.5	10585	2922.4	2658.1	552740	930730	116040	53807
	3	5455.8	11442	4272.2	4148.2	382580	959440	78482	44402

Table 6.3: Performance Index, $a_{out} = 0.05m^3/sec.$

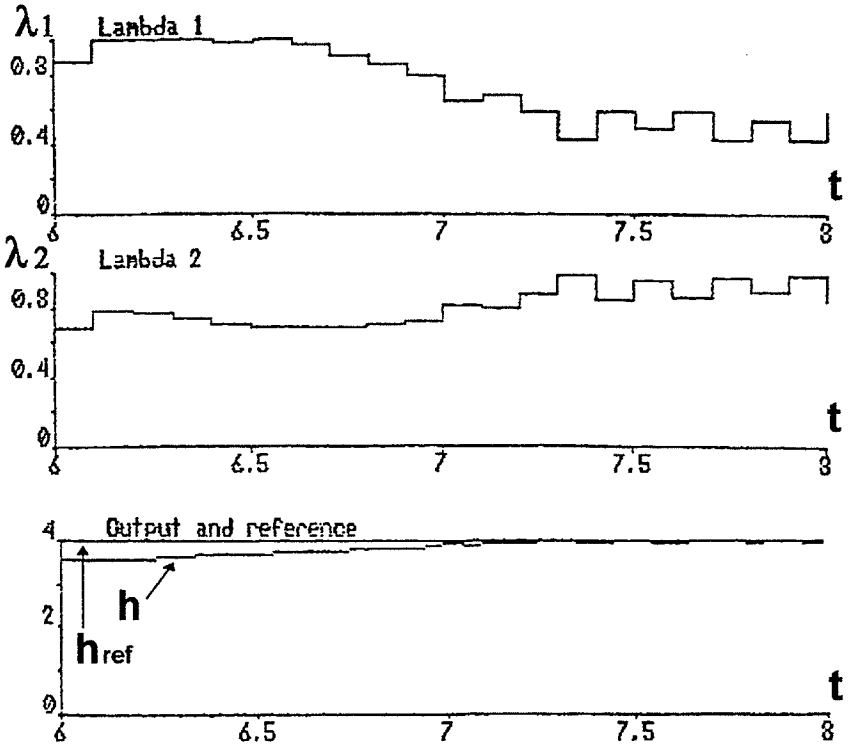

Figure 6.10: Coordinator Action.

Input	Profile	PID$_1$, Q	PID$_2$, Q	PID$_3$, Q	Fuzzy, Q	PID$_1$, Q'	PID$_2$, Q'	PID$_{3Q'}$	F. C., Q'
Step	1	685.32	2088.9	500.67	459.77	7699.0	12772	1899.0	1270.6
	2	690.68	2081.6	500.74	464.42	7497.8	12897	1559.2	1050.7
	3	995.58	2242.6	856.4	827.59	7253.3	12359	2898.4	1869.4
Sine	1	747.02	2700.5	586.61	518.08	8031.0	12792	2443.5	1346.5
	2	775.19	2719.8	592.95	558.71	8232.7	12690	2568.4	1289.6
	3	1216.2	3006.3	1141.2	1074.9	7832.6	13429	4306.9	2365.6
Steps	1	3343.2	2052.7	685.63	448.2	569390	250830	139330	114800
	2	3587.2	2085.0	713.18	439.56	588210	251360	148590	97413
	3	2124.2	2110.2	884.89	573.89	464120	238760	138790	93009
Triangular	1	5924.2	11531	3392.2	2987.5	590440	977530	163750	90515
	2	5798.0	11697	3289.0	3027.1	578530	982960	1.69410	87964
	3	6177.7	12330	4829.9	4573.7	442980	998440	112050	76446

Table 6.4: Performance Index, $a_{out} = random$.

6.4.2 Performance Statistics under Disturbances

The response of the control system is statistically studied in the presence of noise in the control system. In applications, the effects caused by measurement instruments may alter the quality of control. A Gaussian noise of zero mean is being added to the measured level h, for different values of the standard deviation. For these simulations, a_{out} is kept constant (0.05 m^3/sec). Figure 6.11 shows the performance indices Q and Q', averaged over 40 independent experiments for a step of 4 m (half of the maximal level of the tank), versus the standard deviation of the noise signal.

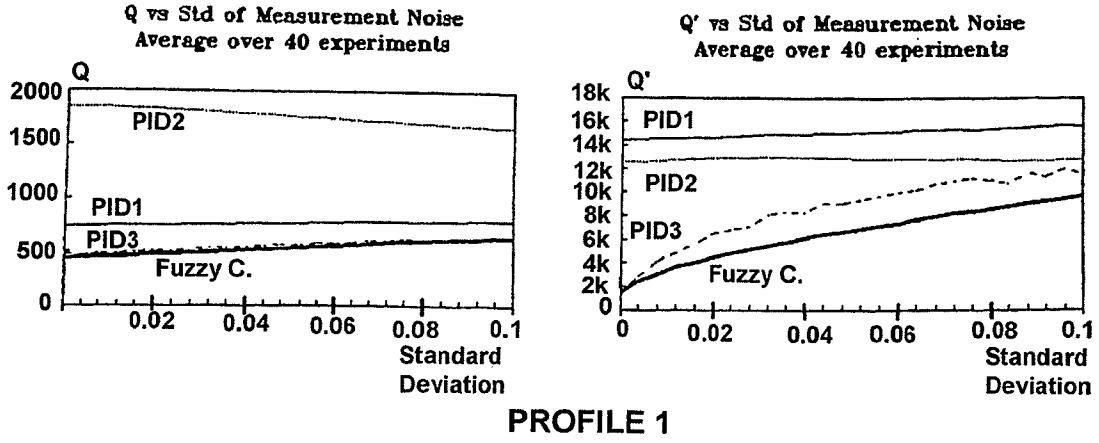

Figure 6.11: Performance Statistics under Disturbances.

As expected, the values of the performance indices increase as the level of noise (standard deviation) increases. Nevertheless, the fuzzy hierarchical controller shows an improved performance over the PID controllers. Its index Q is just slightly smaller than that of the Ziegler-Nichols PID, and as shown in Figure 6.12, the standard deviation of Q is the smallest. The values of Q' are also smaller than those of the optimal PID, and the standard deviation of Q', as shown in Figure 6.13, is the smallest. Considering the statistics of the two performance indices the fuzzy hierarchical controller provides the best response.

6.4.3 Parameters of the Equivalent State-Dependent PID

Consider again Equation 5.9, where $n = 2$ for the simulations. The overall controller resembles a single PID with variable coefficients, and these coefficients can be found experimentally using the fuzzy switching controller. Figure 6.14 shows the values of the parameters K_P and K_I as a function of e for different Δe.

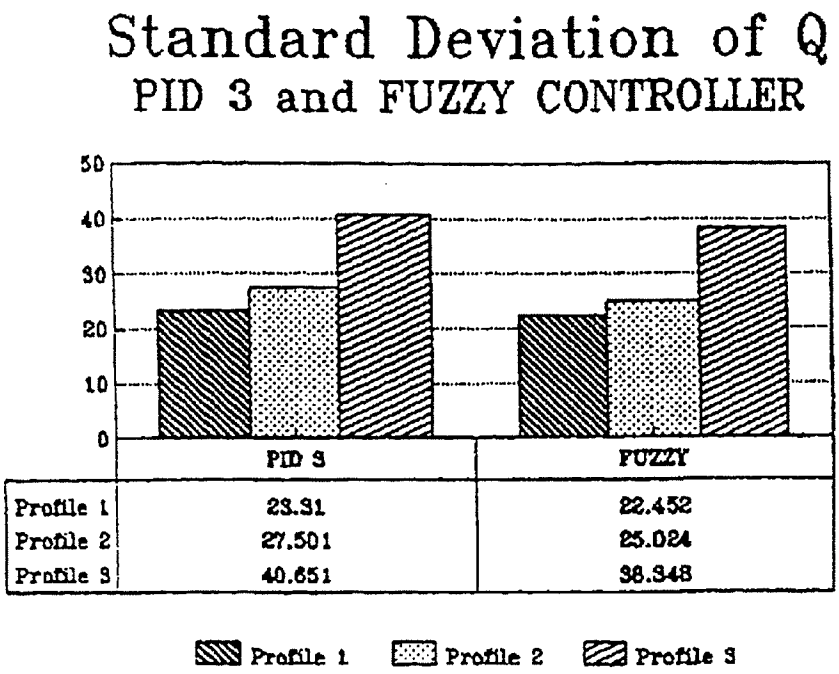

Figure 6.12: Standard Deviation of Q.

Notice that for the values of the parameters of PID_1 and PID_2 in Table 6.4, K_D is the same for both of the controllers. Therefore it can be seen from Equation 5.9 that the combined parameter remains unchanged.

The control signal u of the fuzzy hierarchical controller can also be expressed in terms of e and Δe. This is shown in Figure 6.15. These graphs represent the scheduled control action that the fuzzy coordinator implements.

Summary

This chapter presented the application of the proposed architecture to the control of a nonlinear process. Discrete PID controllers were chosen, since they are very popular in applications. The system was capable of tracking a reference signal by combining the responses of two local PID controllers. The system exhibits a superior performance over that of an optimal PID. The fuzzy coordinator controller brings the response of a the system to the reference in a very short time, and follows the reference even under disturbances. Statistics of the performance of the controller under disturbances were also presented.

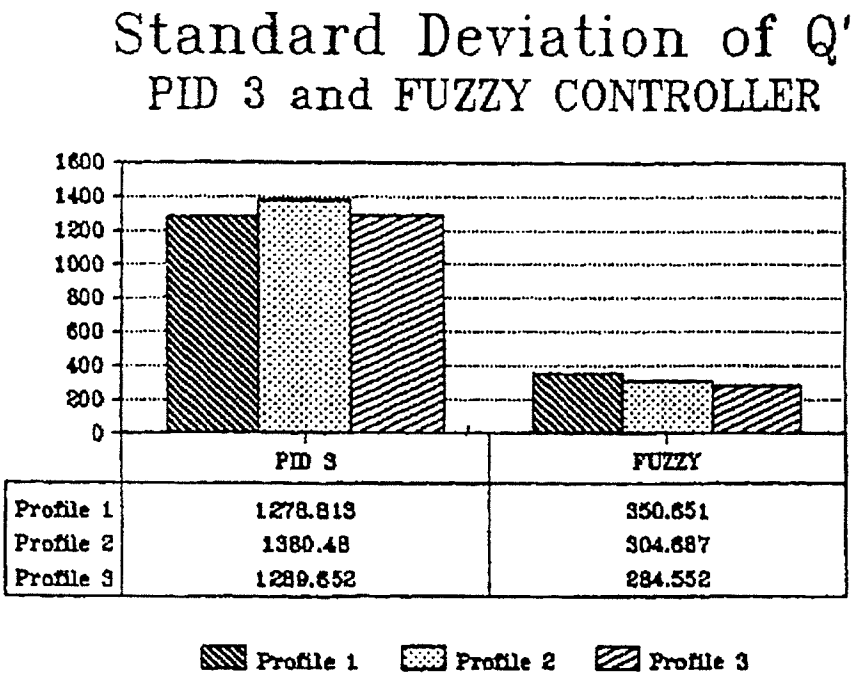

Figure 6.13: Standard Deviation of Q'.

The next chapter presents an application in control of a robot manipulator. The application includes the extension of the architecture to the multivariable case.

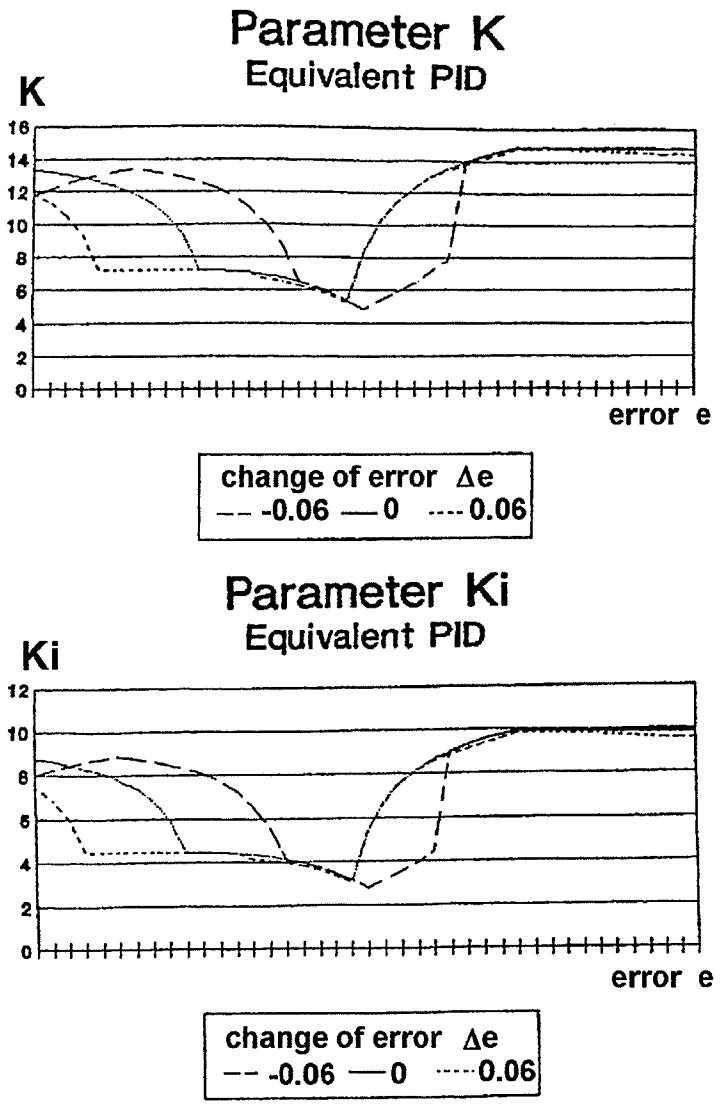

Figure 6.14: Scheduled Parameters of the Equivalent PID Controller.

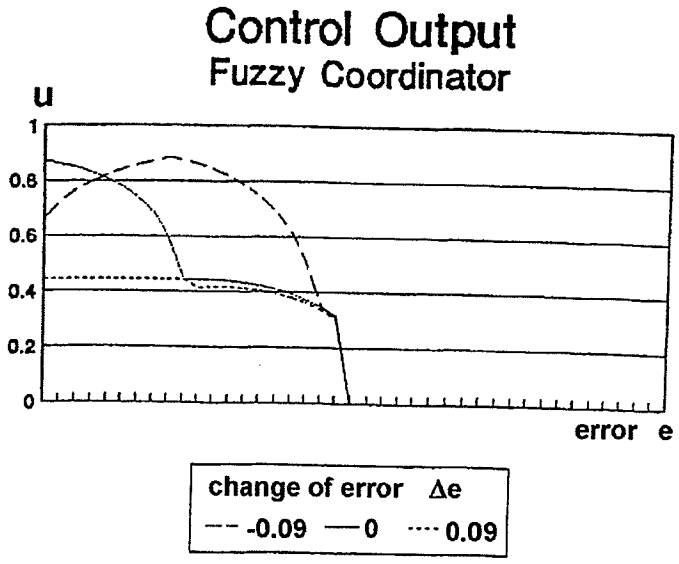

Figure 6.15: Scheduled Control Signal.

Chapter 7

Control of a Robot Manipulator

This chapter extends, the hierarchical controller, and applies it to the control of a multivariable system. The application is the task control of an N degrees-of-freedom robot manipulator. A trajectory generation subsystem provides the controller with a desired trajectory to complete the task, and the role of the controller is to provide the necessary control commands to the actuators of the robot.

A coordinator composed of a fuzzy neural network and local regulators of the PD type are used. The proposed controller is compared to the relay-type control strategy used in some industrial robot controllers. The relay strategy is used in the PUMA robot controller and consists of a PD for gross motion and a PID for fine motion, as explained by Wen [81]. There exist results which show that different types of PD and PID controllers are effective for set point control (Arimoto [2], Kelly [30], Spong [69], and Wen [81]), including several rules for the tuning of the gains. The stability study for the present application example is based on these results. Sufficient conditions are presented for asymptotic stability of the fuzzy-neural-PD controller applied to the control of a nonlinear system.

The architecture considered is the one presented in Figures 2.1 and 2.2, with the difference that all the signals shown are now considered as vectors. The fuzzy coordinator is implemented by a fuzzy-neural network, the local controllers are PD, and the aggregation is a multivariable center of gravity method, similar component-wise to the one presented before.

7.1 Dynamic Model of the Robot

The dynamics of a serial, unconstrained N-link robot manipulator with an actuator at each joint is described by (Spong [69])

$$H(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) = \tau \tag{7.1}$$

where $\mathbf{q}(t), \tau \in R^N$ are the stacked vectors of all joint displacements and applied joint torques, $H(\mathbf{q})$ is a symmetric positive definite (uniformly) inertia matrix, $C(\mathbf{q}, \dot{\mathbf{q}})$ is the matrix of centrifugal and Coriolis torques, defined from the Christoffel symbols, and $g(\mathbf{q})$ is the stacked vector of gravitational torques. The matrices H and C and the vector g are of the appropriate dimensions. A robot manipulator with two degrees-of-freedom, $N = 2$ is considered for this application example. The modeling procedure for the robot under consideration is readily available in the literature (see Spong [69]). The following section states the control problem.

7.1.1 Statement of the Control Problem

The problem under consideration is the tracking of a desired trajectory specified by $\mathbf{q_d}(t), \dot{\mathbf{q}}_\mathbf{d}(t)$,and $\ddot{\mathbf{q}}_\mathbf{d}(t)$. The control problem is then to calculate the torques $\tau \in R^N$ that have to be applied to the joints so that

$$\lim_{t \to \infty} \mathbf{q}(t) = \mathbf{q}_d(t) \tag{7.2}$$

The position error is defined as $\tilde{\mathbf{q}}(t) = \mathbf{q_d}(t) - \mathbf{q}(t)$. The following sections detail the multivariable architecture of the proposed controller.

7.2 The Logic Processor

The coordinator is implemented by a series of fuzzy logic processors, which are driven by the fuzzy sets of error and change of error. Their outputs are the levels of activation of each local controller. The degrees of activation are represented by a static mapping $\lambda = \lambda(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}) \in [0, 1]$. In this case, N logic processors are required, each capable of coordinating one local controller. The hidden layers are considered with eight units, and the weights are calculated following the learning procedure based on step responses of the actuators as in the single variable case. For the present application, step-response graphs are obtained for each of the joints. The

design of the logic processor is reduced to the design of two systems like the ones used in the tank application. The details on the design of the controller used in the example are presented in a later section. The structure of the local controllers is presented next.

7.3 PD Controller for Robots

The present application example considers the standard PD controller for robots. The control law is given by (Arimoto [2], Spong [69])

$$\tau = K_P\tilde{\mathbf{q}} + K_V\dot{\tilde{\mathbf{q}}} \tag{7.3}$$

where K_P and K_V are the constant proportional and derivative gain matrices. It has been shown in Spong [69], that a positioning control objective is achieved, provided that K_P and K_V are positive definite matrices, and that the desired trajectory $\mathbf{q_d}$ is constant. The proposed controller is equivalent to a single PD where the gains are functions of the state $[\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}]$, as shown next.

7.3.1 The Equivalent State-Dependent PD Controller

Let λ_{ik} be the output of the kth logic processor, corresponding to the activation level of the ith local controller for joint k. The final control command for joint k is calculated according to the following center of gravity-like method:

$$\tau_k = \frac{\sum_{i=1}^{n} \tau_{ik}\lambda_{ik}}{\sum_{j=1}^{n} \lambda_{jk}}. \tag{7.4}$$

The final control command vector τ is then given by

$$\tau = \Gamma\left\{\sum_{i=1}^{n} \Lambda_i(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}})\left(K_{Pi}\tilde{\mathbf{q}} + K_{Vi}\dot{\tilde{\mathbf{q}}}\right)\right\} \tag{7.5}$$

where the activation values for controller i are contained in the matrix

$$\Lambda_i(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}) = diag_j(\lambda_{ij}). \tag{7.6}$$

The matrix Γ is given by

$$\Gamma = diag_k\left((\sum_{j=1}^{n} \lambda_{jk})^{-1}\right) \tag{7.7}$$

for $\sum_{j=1}^{n} \lambda_{jk} \neq 0$. Grouping the P and D terms, the state-varying gains of the equivalent PD controller are obtained

$$K_P(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}) = \Gamma \left\{ \sum_{i=1}^{n} \Lambda_i(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}) K_{Pi} \right\} > 0$$
$$K_V(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}) = \Gamma \left\{ \sum_{i=1}^{n} \Lambda_i(\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}) K_{Vi} \right\} > 0. \tag{7.8}$$

The functions K_P and K_V can be generated implicitly by the hierarchical architecture to achieve the control objective. These functions can be stored and implemented in a scheduling control architecture.

7.3.2 Stability

The stability analysis of the system is summarized by the following statement.

The robot described by Equation 7.1 in closed-loop with the controller of Equation 7.8 has the origin as a stable equilibrium. ◇

The stability of the system is determined by the stability properties of the local controllers, given the result presented for the switching operator, and the stability studies for PID-type controllers presented by Arimoto [2], Kelly [30], Spong [69], and Wen [81]. Therefore, the closed-loop system is stable. It is straight forward to show that the state-dependent gains satisfy the stability requirements. □

The details on the implementation of the robot and the controller are presented in the following section.

7.4 Design of the Controller

This application example considers a robot of two degrees-of-freedom ($N = 2$) as shown in Figure 7.1. The method presented, however, applies to the general case of N degrees-of-freedom.

The model of the robot was implemented as a continuous-time system with the values of the parameters shown in Table 7.1. The task assigned to the robot is explained in the following section.

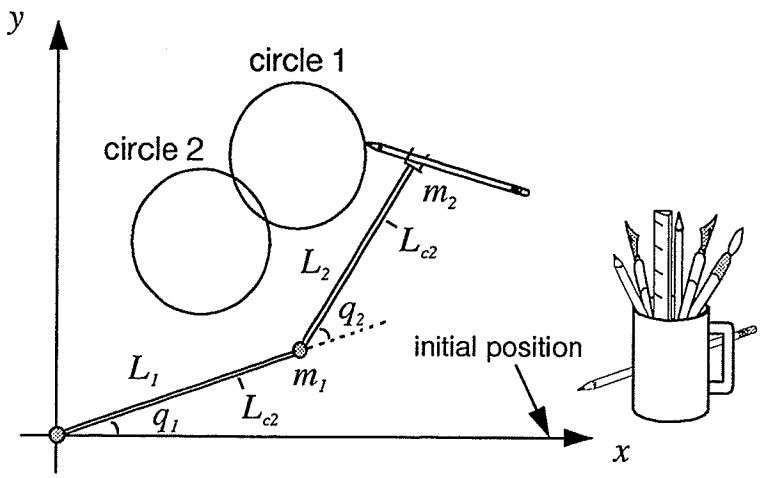

Figure 7.1: Robot Manipulator.

7.4.1 Robot Task

The task for the robot is to draw two circles of radius 7.5 cm in a Cartesian plane, centered at (22 cm, 22 cm), and (12 cm, 12 cm), with a frequency of 1 circle/second. The desired trajectory is shown in Figure 7.2.

The desired joint positions, velocities, and accelerations required to draw a circle are provided by a trajectory generation system that solves the inverse kinematic problem. This system transforms the specifications from Cartesian coordinates to

Parameter	*Value*	*Description*
L_1	*0.25m*	*Length of Link 1*
L_2	*0.16m*	*Length of Link 2*
L_{c1}	*0.20m*	*Length to Center of Gravity 1*
L_{c2}	*0.14m*	*Length to Center of Gravity 2*
m_1	*9.5kg*	*Mass of Link 1*
m_2	*5.0kg*	*Mass of Link 2*
I_1	$0.0043 kgm^2$	*Moment of Inertia of Link 1*
I_2	$0.0061 kgm^2$	*Moment of Inertia of Link 2*

Table 7.1: Parameters of the Robot.

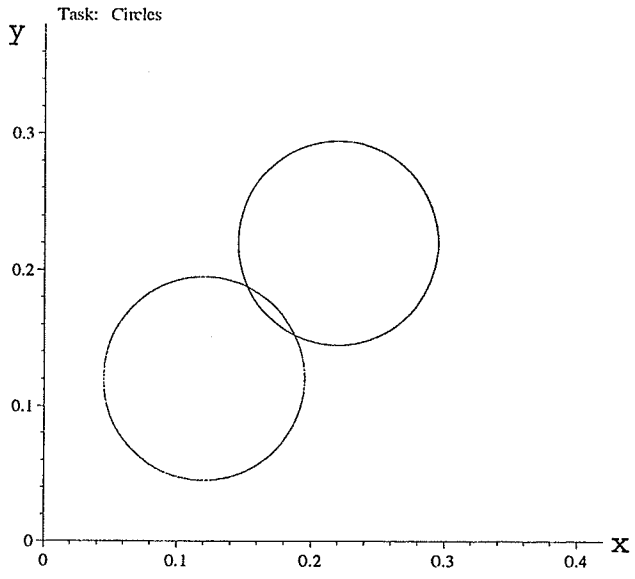

Figure 7.2: Target.

the joint variables[1].

The task is provided to the controllers as a collection of functions representing the angular positions, velocities and accelerations of the joints. The robot must accurately follow these functions. The following section explains the details on the implementation of the logic processors.

7.4.2 Construction of the Logic Processor

For the present application three fuzzy sets for error $\tilde{q}_k$ and three for change of error $\dot{\tilde{q}}_k$ in joint k are considered for two logic processors of two outputs each. The hidden layers have eight units each. The learning is completed from step-responses of the links as in the single-variable case. The learning stage was completed in a supervised mode from a training set consisting of values of error and desired values of the output. The training set was formed by using the data acquisition technique introduced before. A data set was formed from the graphs of the responses of the

[1]The author thanks Dr. R. Kelly from CICESE, Mexico, for providing the source code of the inverse kinematic problem for the simulation included in his book on Control of Robot Manipulators [30]. This allowed the repetition of his results for comparison, which was very valuable in the development of the code for the simulations included in this chapter.

controllers PD_1 and PD_2. The overall fuzzy neural network was implemented as a discrete-time system.

7.4.3 Local Control Algorithms

For the tracking problem under consideration two local PD algorithms ($n = 2$) were needed. These local controllers were selected as in the single variable case, PD_1 was fast for operation far from the operating point, and PD_2 for fine motion.

The independent responses of the controllers PD_1 and PD_2 are shown in Figures 7.3, 7.4, 7.5, and 7.6. Notice that PD_1 drives the system to an initial contact with the reference signal in a very short time, and its speed of response determines the speed of the switching controller. The error in the response of PD_2 is slowly approaching the reference. This slow response is good for regulation and disturbance rejection.

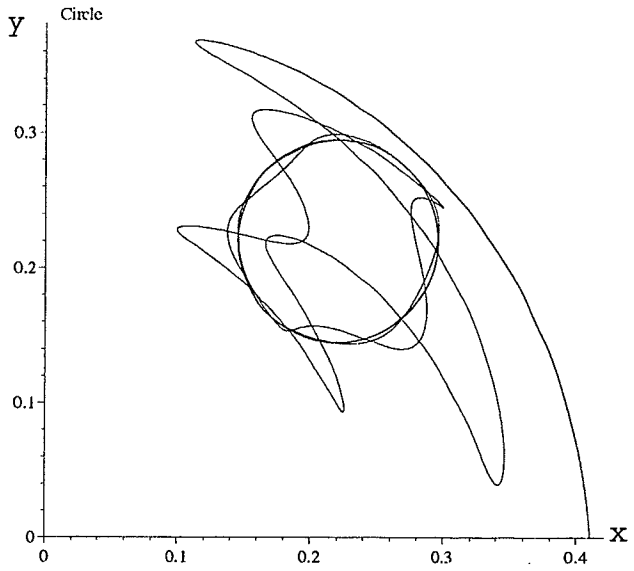

Figure 7.3: Response with PD_1.

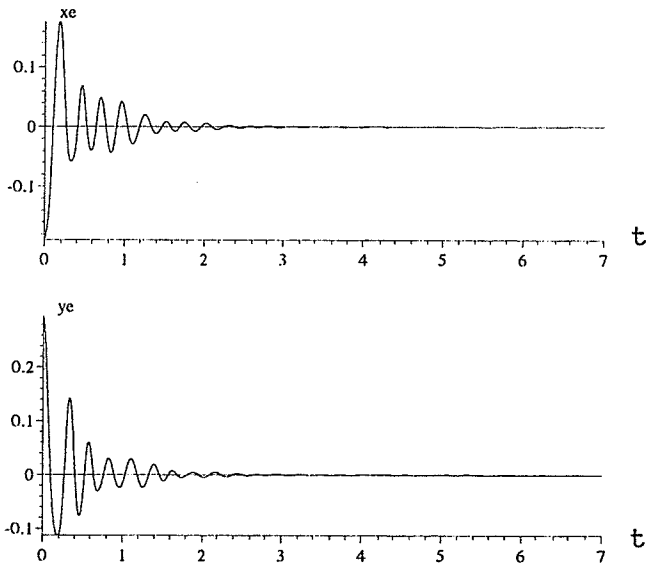

Figure 7.4: Joint errors with PD_1.

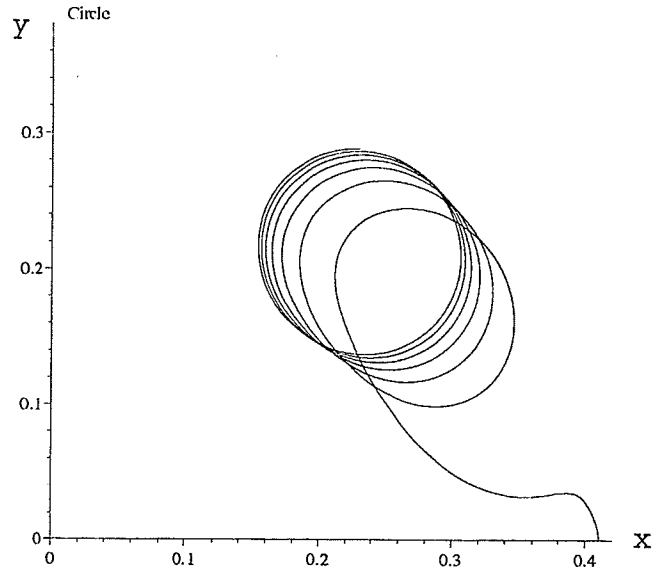

Figure 7.5: Response with PD_2.

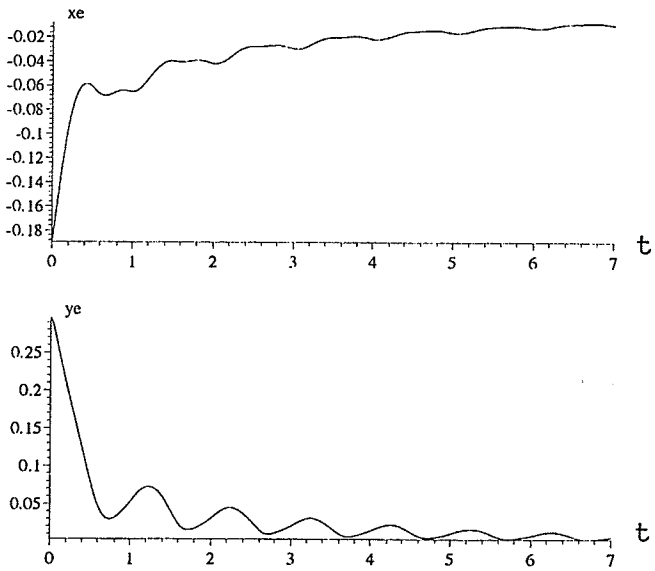

Figure 7.6: Joint errors with PD_2.

7.5 Simulation Results

The initial condition for the robot manipulator was the resting position with zero joint displacements, that is $q_1 = q_2 = 0$. The total simulation time was 7 seconds. The robot was to draw the first circle three times (3 seconds) and the rest of the time the second circle (4 seconds). The results are presented in Figures 7.7 and 7.8. The joint errors are shown in Figure 7.9.

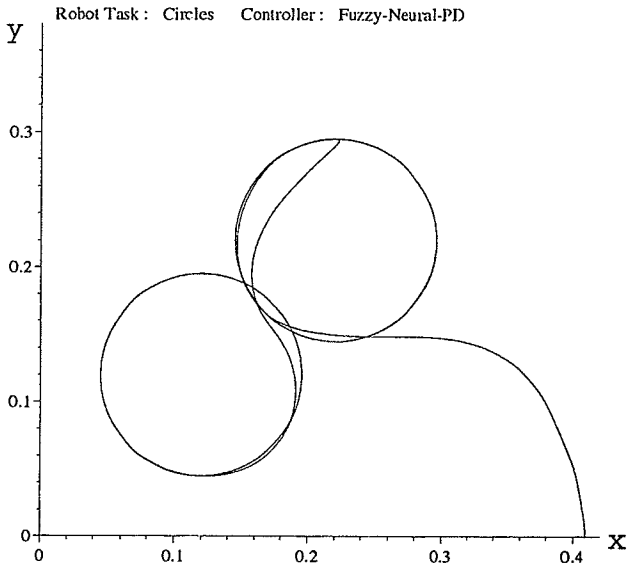

Figure 7.7: Fuzzy-Neural-PD Controller.

It can be observed that the task has been accomplished and that the performance is very good. There is no overshoot and the error vanishes in about 1/4 of a second. The values of the membership functions during the first second of operation are shown in Figure 7.10. The labels of the fuzzy sets are NE, ZE, PE, NCE, ZCE, and PCE, which respectively stand for negative error, zero error, positive error, negative change of error, zero change of error, and positive change of error. The values of the selection variables and the joint control signals are shown in Figure 7.11. The selection variables λ_{ij} are represented by muij in the figure.

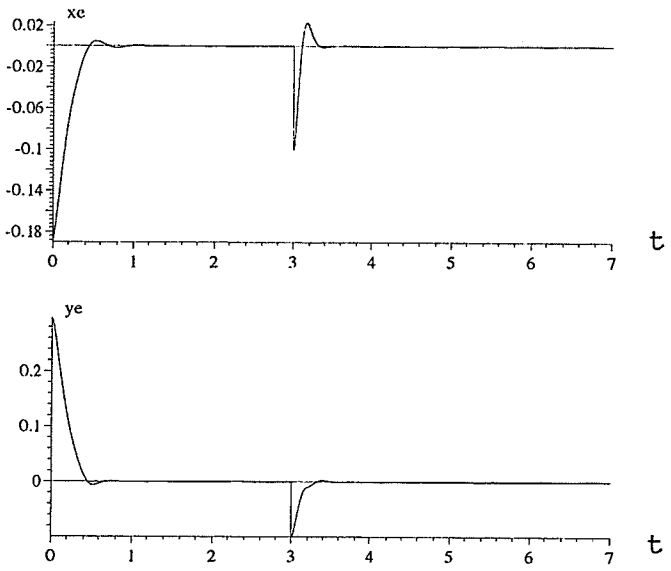

Figure 7.8: End-Effector Errors.

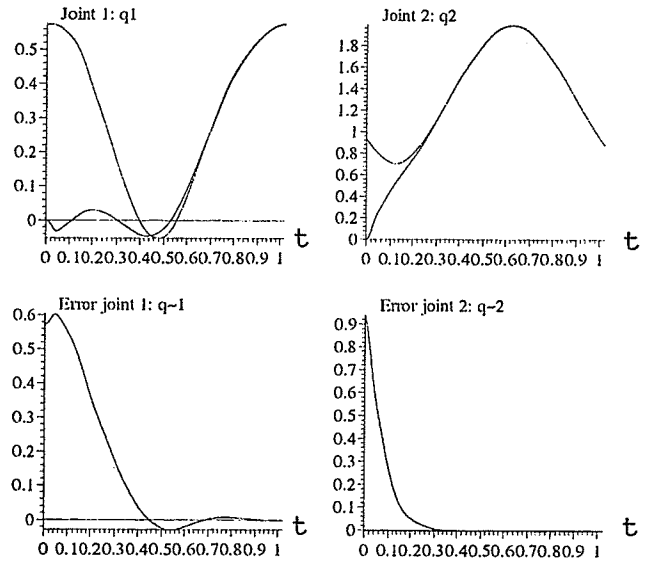

Figure 7.9: Joint Errors.

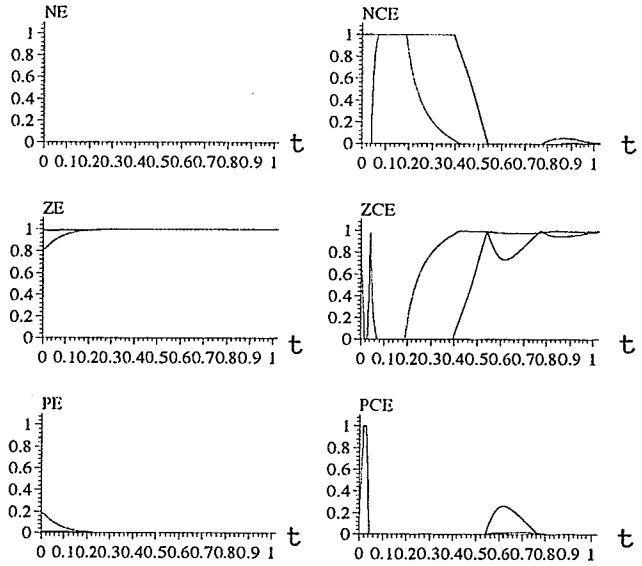

Figure 7.10: Values of the Membership Functions.

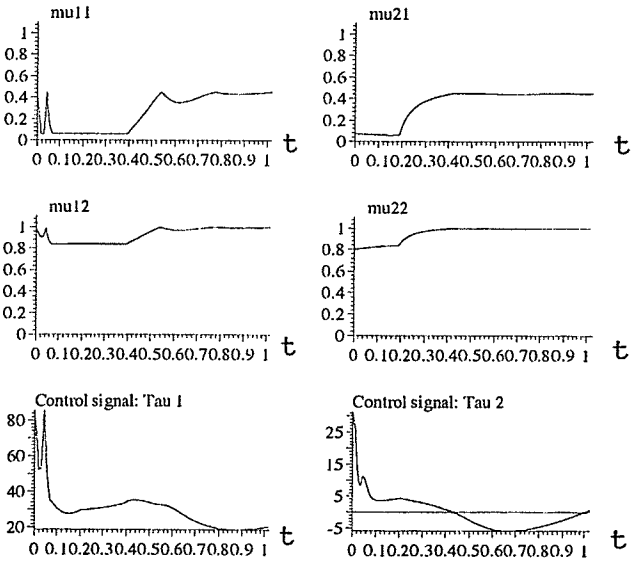

Figure 7.11: Selection Variables.

7.5.1 Operation under Disturbances

The operation of the control system under a disturbance in the control signal τ was considered. The disturbance was modeled by a random signal with a rectangular distribution and an amplitude of 10 units which is approximately 10% of the average control signal τ. Figures 7.12 and 7.13 show the response of the system. Notice that the response deteriorates but nevertheless the task is accomplished.

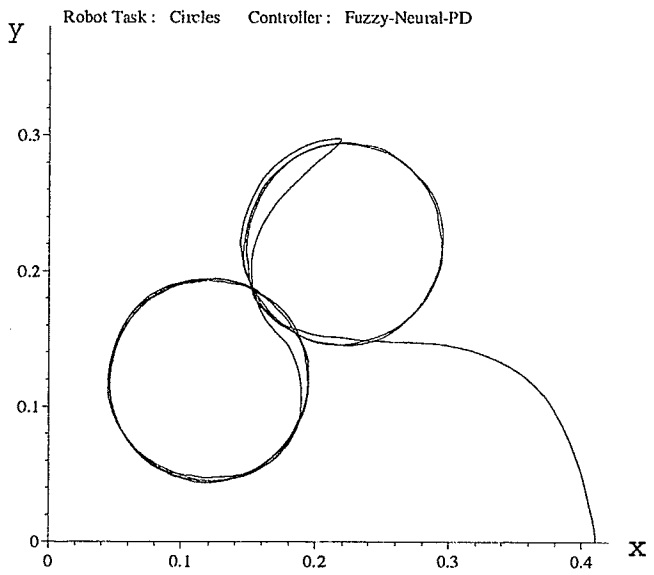

Figure 7.12: Fuzzy-Neural-PD Controller.

The error signal is shown in Figure 7.14 during the first second of operation. The values of the membership functions also during the first second of operation are shown in Figure 7.15. Figure 7.16 displays the values of the selection variables and the joint control signals. The level of noise affecting the control signal can be observed in this figure.

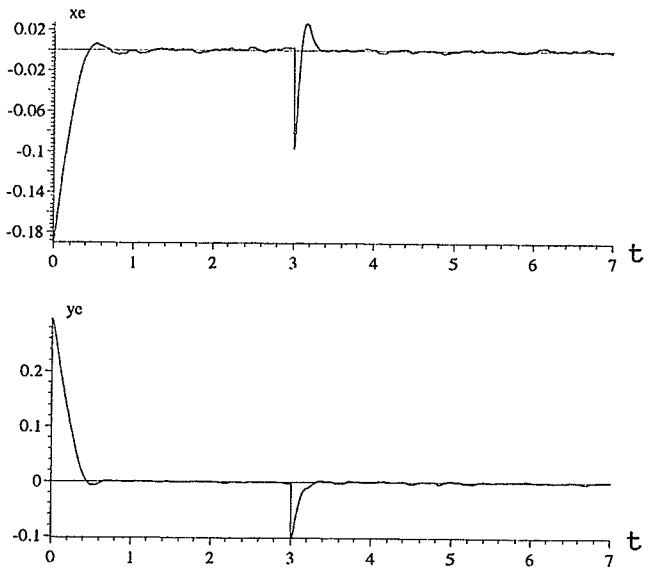

Figure 7.13: End-Effector Errors.

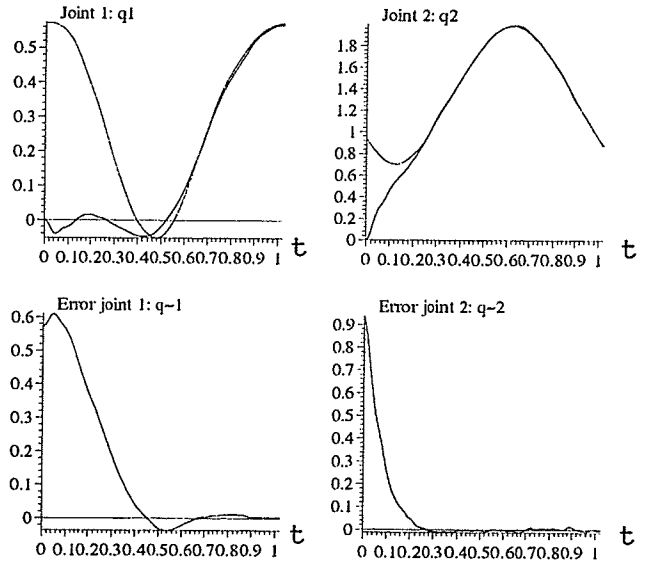

Figure 7.14: Joint Errors.

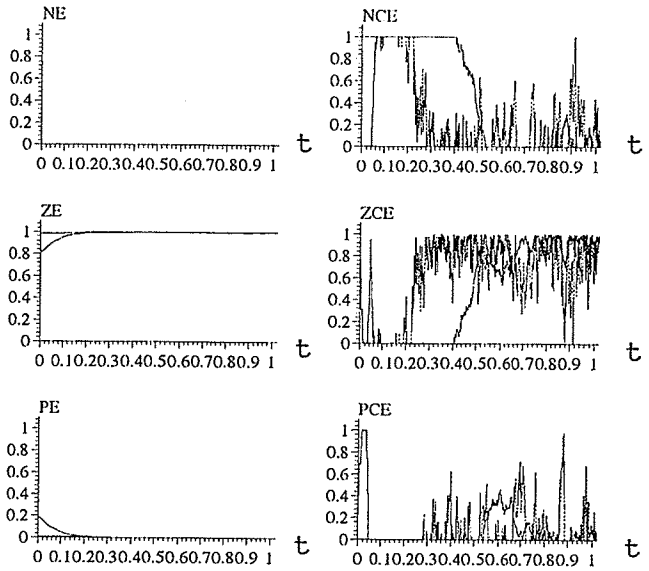

Figure 7.15: Values of the Membership Functions.

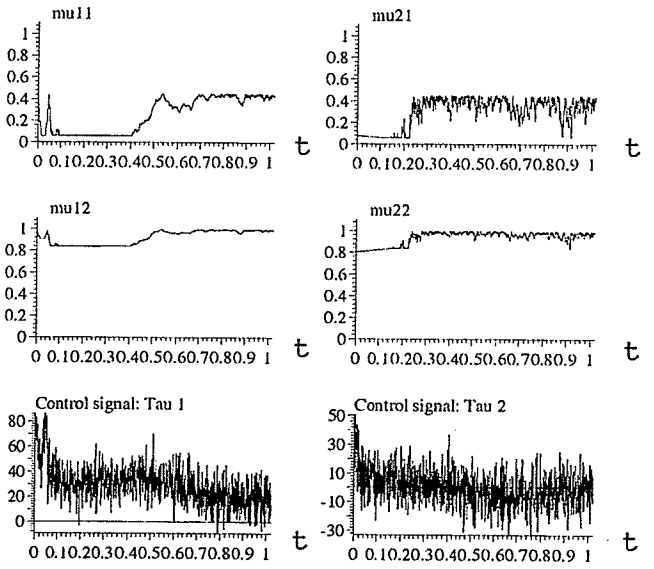

Figure 7.16: Selection Variables.

7.5.2 The Relay-Type Controller

A simulation of the relay-type algorithm is included for comparison. An OR-type relay has been chosen. The switching occurs if the error in joint i or the change error in joint i is greater than the threshold values of 0.05 and 0.01, respectively. The response is presented in Figures 7.17 and 7.18.

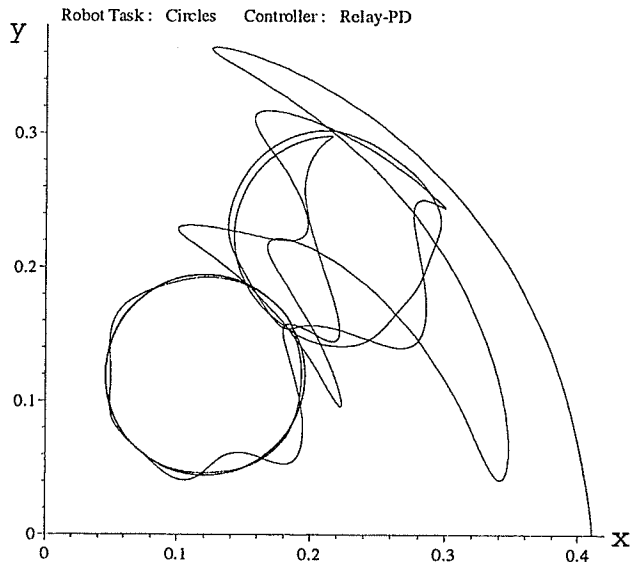

Figure 7.17: Task Simulation, Relay(OR)-PD Controller.

Notice that the oscillations in the transient response during the first 2 seconds cause the deterioration of the first two circles. Also notice that the robot is far from the reference during the transient. This could damage the robot or its environment in a real application, and for this reason the response is not acceptable.

The membership functions and the control signals have been included in Figure 7.19. The on-off operation of the relay can be observed in this figure.

7.5.3 Operation of the Relay under Disturbances

An AND relay is included for comparison with the OR. It was found that the performance of the AND relay is better than that of the OR relay in terms of the quality of the response. In Figures 7.20 and 7.21 the response of the AND relay is shown. The same disturbance as before is added and the same threshold values as in the OR relay are used.

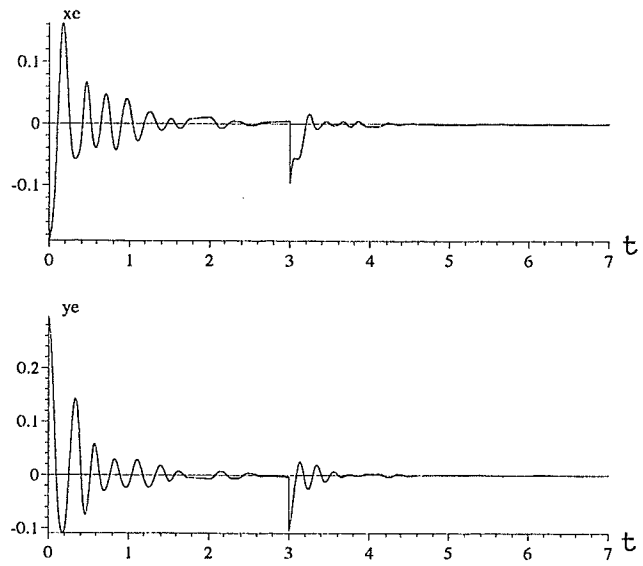

Figure 7.18: Relay(OR)-PD Errors.

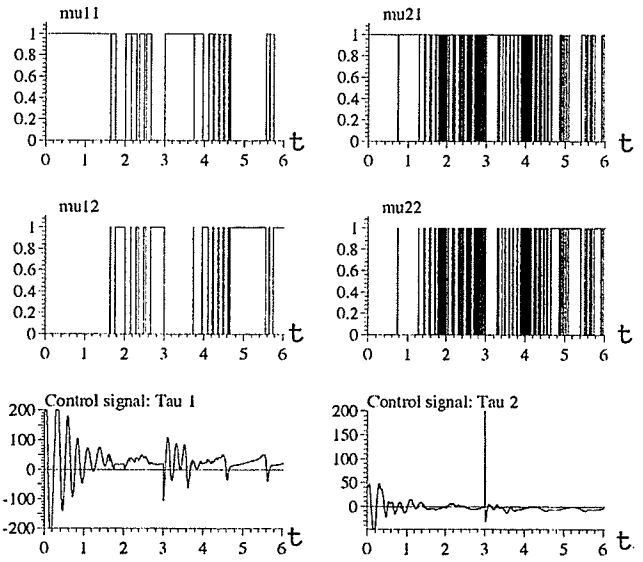

Figure 7.19: Relay(OR)-PD Switching.

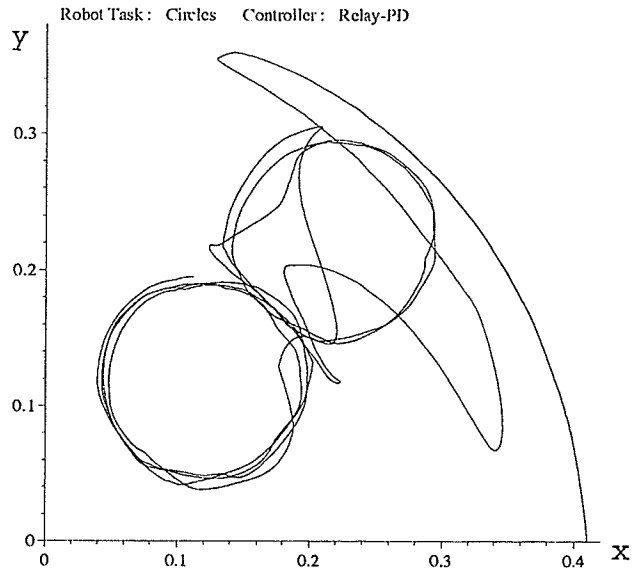

Figure 7.20: Task Simulation, Relay(AND)-PD Controller with Noise.

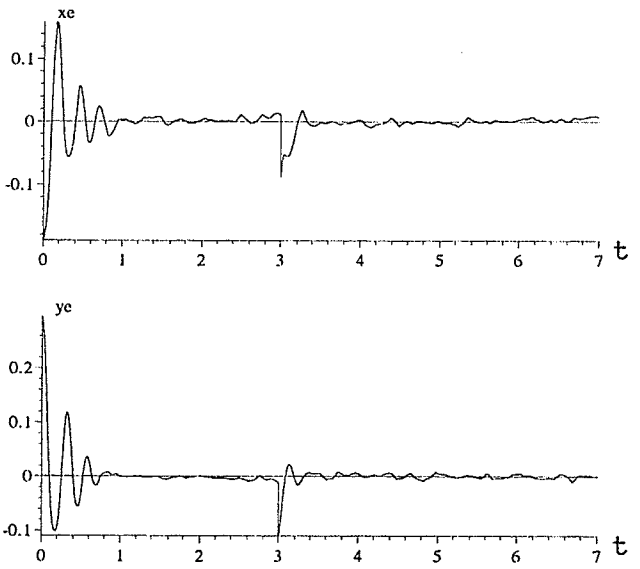

Figure 7.21: Relay(AND)-PD Errors with Noise.

Notice that the task is accomplished after a large overshoot and 1 second of oscillations. The first circle is deteriorated and again, the robot or its environment could be damaged by the overshoot. This makes the response unacceptable.

In a practical application the overshoot can be reduced by the use of filters, however this can reduce the speed of response. The proposed controller is a better choice than a relay since the overshoot is almost insignificant and the response is faster.

7.5.4 Summary

An application of the proposed controller to the control of robot manipulators has been presented in this chapter. The simulations showed that a very good performance can be achieved even under disturbances. A comparison was made with the relay algorithm. The quality of the response provided by the proposed controller was better than that of the relay. The response obtained is a satisfactory solution to the control tracking problem.

A quantitative comparison has been made in Table 7.2, which shows the values of the performance indices Q and Qp described in the following equations for each joint k

$$Q_k = \sum_{t=1}^{T} \tilde{q}_k(t)^2 \qquad Qp_k = \sum_{t=1}^{T} t|\tilde{q}_k(t)| \tag{7.9}$$

It can be observed that taking into consideration the quantitative and qualitative properties of the response, the proposed architecture provides the best response overall.

Robot Controller	*Distributed Controller*		*Relay Switch*		*PID 1 Fast*		*PID 2 Regulator*	
Index	*Q*	*Qp*	*Q*	*Qp*	*Q*	*Qp*	*Q*	*Qp*
Index Joint 1	*68.83*	*143.24*	*82.29*	*450.04*	*80.77*	*158.54*	*210.98*	*1638.70*
Index Joint 2	*92.50*	*232.36*	*70.33*	*246.50*	*31.81*	*18.38*	*78.83*	*836.27*
Joint 1+Noise	*73.05*	*266.15*	*81.41*	*396.59*	*81.41*	*236.14*	*208.18*	*1436.00*
Joint 2+Noise	*87.84*	*349.93*	*70.13*	*254.21*	*31.75*	*51.28*	*84.57*	*1331.80*
Sum	*322.22*	*991.68*	*304.16*	*1347.34*	*225.74*	*464.34*	*582.56*	*5242.77*
Total Index	*1313.90*		*1651.50*		*690.08*		*5825.33*	
Quality	*Good*		*Not Good*		*Not Good*		*Not Good*	
Best Overall	**X**							

Table 7.2: Performance Evaluation.

In the following chapter the proposed architecture is used with a different perspective. A solution to the problem of distributed modeling is introduced.

Chapter 8

Distributed Modeling

In this chapter, the problem of distributed modeling is considered. This problem refers to the use of several models to approximate the model of a complicated plant for control purposes. The issue of linearization of systems is initially discussed. Several techniques for linearization that have been developed for this thesis have been included in the appendices, as well as a survey of known results. The choice of a technique depends on the data available about the plant.

Distributed modeling is considered as the problem of calculating linearizations of a complicated system around several operating points to cover a whole range of operation for control purposes. The models are then combined by a coordinator to approximate the original model. The coordinator smoothly switch bettwen the local models to approximate a complicated system.

A similar architecture has been presented by Johansen and Foss [28]. They use fixed gaussian functions to switch between the models, however, no qualitative information is used during the design. The approach presented in this thesis includes a system capable of learning the relationships between the models based on qualitative knowledge. Thus, the proposed system is more flexible and a systematic design method is available.

The first step is to calculate the number and parameters of the models. Once the models have been identified, the same hierarchical architecture presented for the distributed control problem is used. Several linearization methods have been developed for this thesis including algorithms to calculate the number and parameters of the linear models. A brief overview of linearization methods is presented in the following section.

8.1 Overview of Linearization Methods

This section briefly describes methods for linearization of functions and equations that can be used in the distributed modeling architecture. The details of the methods are included in Appendices B to H. A brief description of the methods is presented next.

Appendix B includes an algorithm developed for the calculation of a piece-wise approximation of a continuous function in a plane. Two methods were developed for linearization of functions in the space. The first method, included in Appendix C, is based on the Frenet frame of a curve (see O'Neill [46]). The second method, presented in Appendix D, is based on the curvature. A method was also developed for linearization of models described by data in a multi-dimensional space. This method is presented in Appendix E.

Difference equation models have also been considered. Appendix F presents a method developed based on the curve representation of trajectories. In the case of differential equations, a collection of available techniques has been included in Appendix G. A technique known as exact linearization for nonlinear systems has been included in Appendix H.

The following section addresses the concept of distributed modeling.

8.2 Distributed Modeling

Distributed modeling refers to the calculation of several local models to approximate the model of a complicated system for control purposes. As mentioned before, for different problems there are different techniques to calculate the models, depending on the data available. The proposed distributed modeling architecture is based on the idea of smooth switching introduced in this thesis. The distributed model consists of the local models and a fuzzy coordinator. The idea is the same as in the case of distributed control. The coordinator calculates degrees of activation of the models depending on the point of operation. It is very important to determine the number of models required to cover a specified range of operation.

In this discussion, models described by linear systems are considered. The discussion can easily be adapted for any of the other linear models described. The

following continuous-time linear system is considered in the discussion

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= C\,x \end{aligned} \tag{8.1}$$

where x, y, $u \in R$ are the state, the output, and the control signal, respectively. The matrices A, B, and C are constant matrices of appropriate dimensions. Since the purpose of this analysis is to improve the performance of an existing controller, it can be assumed that the pair (A, B) is controllable, and that (A, C) is observable. These concepts can be found in Chen [12].

Given a nonlinear system and the range of operation, a set of n linear models can be obtained. Consider the states $\mathbf{x}_o, \cdots, \mathbf{x}_{n-1} \in R^m$ where m is the dimension of the state space of the nonlinear system. The linear models can be represented as follows

$$\begin{array}{llll} \dot{\mathbf{x}}_o = A_o\,\dot{\mathbf{x}}_o + B_o\,u_o & & \cdots & \dot{\mathbf{x}}_{n-1} = A_{n-1}\,\dot{\mathbf{x}}_{n-1} + B_{n-1}\,u_{n-1} \\ y_o = C_o\,x_o & & & y_{n-1} = C_{n-1}\,x_{n-1} \\ \Downarrow & & & \Downarrow \\ G_o(s) = C_o\,(sI - A_o)^{-1}\,B_o & & \cdots & G_{n-1}(s) = C_{n-1}\,(sI - A_{n-1})^{-1}\,B_{n-1} \end{array} \tag{8.2}$$

where $G_o(s), \cdots, G_{n-1}(s)$ are the corresponding transfer functions in the Laplace domain s, represented by

$$\begin{aligned} G_o(s) &= \frac{a_{oo}s^m + a_{o1}s^{m-1} + \cdots + a_{om}}{s^m + b_{o1}s^{m-1} + \cdots + b_{om}} \\ &\vdots \\ G_{n-1}(s) &= \frac{a_{(n-1)o}s^m + a_{(n-1)1}s^{m-1} + \cdots + a_{(n-1)m}}{s^m + b_{(n-1)1}s^{m-1} + \cdots + b_{(n-1)m}} \end{aligned} \tag{8.3}$$

Once the linear models have been determined, different approaches can be taken for control of the distributed model. The following section describes the approaches considered.

8.2.1 Control Methodologies

The control methodology used with the distributed model depends on the configuration of the local model, specifically on the input/output configuration. The first approach consists of the design of a local controller for each local model; thus, the overall control system is made of n local closed-loop models. The second approach consists of the design of a single controller for the distributed model. The two methodologies are described in the following section.

8.3 Distributed Model with n Model-Controller Pairs

The approximated model consists of n local models and n controllers. The overall model can be represented as follows, where the symbol '$\rightarrow$' is read 'produce(s)'

$$e \rightarrow \left\{ \begin{array}{ccccc} u_o & \rightarrow & G_o(s) & \rightarrow & y_o \\ u_1 & \rightarrow & G_1(s) & \rightarrow & y_1 \\ & & \vdots & & \vdots \\ u_{n-1} & \rightarrow & G_{n-1}(s) & \rightarrow & y_{n-1} \end{array} \right\} \rightarrow y. \tag{8.4}$$

In this model, each controller u_i receives the same value of error as the other controllers, and calculates the correcting action for $G_i(s)$ which, in turn, generates y_i. The output of the model y is generated using the degrees of activation calculated by the fuzzy system and the output of the local model-controller pairs. If model j is far from its point of operation, it will produce an output which does not represent the output of the original system; thus, the fuzzy system will assign a low value of activation.

If it is guaranteed that each of the model-controller pairs is stable, then the stability of the overall system depends on the properties of each local loop and the fuzzy switching. Again, PID and optimal controllers are considered.

8.4 Distributed Model with a Single Controller

In this approach, the distributed model is composed of n local models which are activated depending on the point of operation. One controller is designed which calculates the control command for the overall system as follows, where the symbol '$\rightarrow$' is read 'produce(s)'

$$e \rightarrow u \rightarrow \left\{ \begin{array}{ccc} G_o(s) & \rightarrow & y_o \\ G_1(s) & \rightarrow & y_1 \\ & \vdots & \\ G_{n-1}(s) & \rightarrow & y_{n-1} \end{array} \right\} \rightarrow y. \tag{8.5}$$

Since the input to each of the models is the same and the output is a combination of the individual outputs, the model can be represented by a single model with varying

coefficients. As before, the center of gravity method is considered. The aggregation of the outputs is given by

$$y = \frac{\sum_{i=1}^{n} y_{i-1}\lambda_i}{\sum_{j=1}^{n} \lambda_j} \tag{8.6}$$

which can be written as

$$y = \frac{\sum_{i=1}^{n} C_{i-1}\mathbf{x}_i\,\lambda_i}{\sum_{j=1}^{n} \lambda_j}. \tag{8.7}$$

Since the state is the same for every model, the output equation can be rewritten to obtain

$$y = \sum_{i=1}^{n} C_{xi}\mathbf{x}_i \tag{8.8}$$

where

$$C_{xi} = \frac{C_{i-1}\,\lambda_i}{\sum_{j=1}^{n} \lambda_j} \tag{8.9}$$

which represents the output of the approximated model. Each local model calculates the new state according to its parameters, and this produces a state that is an aggregation of all the states. Thus, the output is an aggregation of all the outputs. The following section addresses the robustness of the proposed distributed model.

8.5 Robustness of the Distributed Model

Robustness refers to how well the system maintains its behaviour under variations of its parameters. Robustness can provide information about the accuracy of a model. To determine the accuracy of the proposed distributed modeling system a study of how robust the system is to changes in the parameters is necessary.

The problem of linear systems with uncertain coefficients has been extensively studied (see Barmish [7]). There exists a method to study the robustness of a linear system based on the study of its characteristic equation, that is, on the properties of the poles of the denominator of the transfer function.

Given a state equation model, it is well-known that the corresponding transfer function can be calculated as follows

$$G(s) = C(sI - A)^{-1}B. \tag{8.10}$$

It is considered that the linear models are in the transfer function form.

Consider a well-ordered set of linear systems, where, in order to get to the $(n-1)$th linearization, it is necessary to pass through the others. In any well-ordered set there is a minimum element. The distributed model is a well-ordered set of linear systems, indexed by the point of operation. Thus, there is a minimum linear system in the set. This system is minimum in terms of the initial conditions in state space, and as the system moves in this space the other linear systems become active. Let the distributed model be $G^*(s)$ and the minimum system be $G_o(s)$. The following sequence represents $G^*(s)$, where the symbol '$\rightsquigarrow$' is read 'is followed by'

$$\overbrace{G_1(s) \rightsquigarrow G_2(s) \rightsquigarrow \cdots \rightsquigarrow G_n(s)}^{G^*(s)} \tag{8.11}$$

The static distributed model is equivalent to the transfer function with unknown constant coefficients given by

$$G^*(s) = \frac{a_m^* s^m + a_{m-1}^* s^{m-1} + \cdots + a_o^*}{b_m s^m + b_{m-1}^* s^{m-1} + \cdots + b_o^*} \tag{8.12}$$

with

$$min_i(a_{ij}) \leq a_j^* \leq max_i(a_{ij}) \tag{8.13}$$

and

$$min_i(b_{ij}) \leq b_j^* \leq max_i(b_{ij}) \tag{8.14}$$

where $i \in [1, n]$, n is the number of models, and $j \in [0, m], m$ is the order of the equivalent system. Thus, the coefficients of the varying system $G^*(s)$ are determined by the coefficients of the local models. The following theorem is the main tool in analyzing the robustness of a system with uncertainties. Given the structure of the distributed model, the theorem can be used to study robustness in applications. In the proposed architecture it is necessary to study the stability of each of the local models. The stability can be easily determined by studying the bounds of the parameters using Kharitonov's Theorem. This is specially useful when the number of models is large, since it is not necessary to study each model individually.

Kharitonov's Theorem: An interval polynomial

$$p(s) = q_o + q_1 s + \cdots + q_{m-1} s^{m-1} + q_m s^m \tag{8.15}$$

with invariant degree and coefficients bounded by

$$x_j \leq q_j \leq y_j \tag{8.16}$$

is robustly stable if and only if the following four polynomials are Hurwitz

$$\begin{aligned} p_1(s) &= y_o + x_1 s + x_2 s^2 + y_3 s^3 + y_4 s^4 + x_5 s^5 + x_6 s^6 + \cdots \\ p_2(s) &= y_o + y_1 s + x_2 s^2 + x_3 s^3 + y_4 s^4 + y_5 s^5 + x_6 s^6 + \cdots \\ p_3(s) &= x_o + x_1 s + y_2 s^2 + y_3 s^3 + x_4 s^4 + x_5 s^5 + y_6 s^6 + \cdots \\ p_4(s) &= x_o + y_1 s + y_2 s^2 + x_3 s^3 + x_4 s^4 + y_5 s^5 + y_6 s^6 + \cdots \end{aligned} \tag{8.17}$$

$\diamond$.

Notice that the bounds on the parameters of the distributed model given by Equation 8.14 are of the form required by the theorem in Equation 8.16. That is

$$\begin{aligned} x_j &= min_i(b_{ij}) \\ q_j &= b_j^* \\ yj &= max_i(b_{ij}). \end{aligned} \tag{8.18}$$

Thus, for a specific application, the robustness of the system can be determined by constructing the four polynomials of Equation 8.17 and examining their roots. If all the roots are in the left-hand side of the complex plane, then the system is robustly stable by the theorem.

This result can also be used for validation of the distributed model. The region of stability of the model can also be determined by examining the roots of the four polynomials. Based on this, the appropriate local linear systems can be redesigned.

The stability results obtained by using Kharitonov's theorem are for a more general case than the proposed system. Notice that the theorem considers stability in the whole range given by Equation 8.16. The theorem assumes that the parameters take constant values within those bounds, but no particular rules are followed. The parameters of the proposed distributed model do not cover the whole range, since the switching system follows a certain trajectory while changing from model to model.

Next, an application to the control of a nonlinear single-link robot is presented.

8.6 Distributed Modeling of a Single-Link Robot

A simple one-link robot without friction given by the following equation is considered

$$ml^2\ddot{q} + mgl\sin(q) = \tau \tag{8.19}$$

where m, l, g are the mass, length of the link and gravity, respectively, q is the angular position, and τ the input torque. The idea is to obtain a pair of linear models valid

in specific regions, and then apply the hierarchical structure to switch between the models (see Figure 8.1). The switching of the models must behave as the nonlinear system. Thus, given an initial condition the distributed model must oscillate at the same amplitude and frequency as the original model.

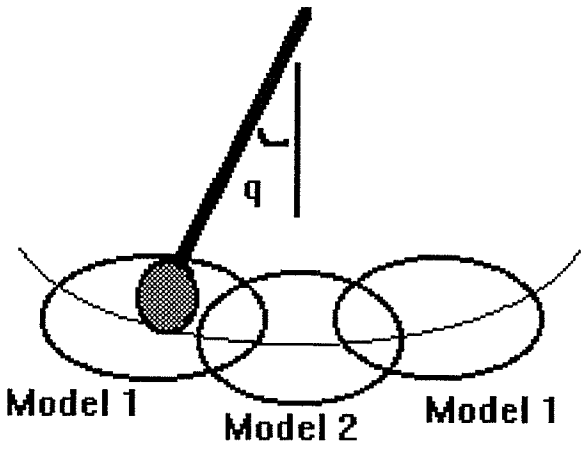

Figure 8.1: Distributed Modeling of a Single-Link Robot.

The code for the simulations presented here was written in the Simnon simulation language [15], [16].

8.6.1 Linearization of the Model

The linearized models were obtained by using Taylor's expansion method for control systems (presented in Appendix G). The points of linearization chosen are

$$\begin{aligned} q_{o1} &= \pi/4 \; rad \\ q_{o2} &= 0 \; rad. \end{aligned} \tag{8.20}$$

The matrices of the linearized model are given by

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ -g\cos(q_o)/l & 0 \end{bmatrix} \\ B &= \begin{bmatrix} 0 \\ 1/ml^2 \end{bmatrix}. \end{aligned} \tag{8.21}$$

The system was discretized using the formulas presented in Chapter 5, in the section on optimal control. The discretization formulas were implemented in Matlab

[42]. The values of the parameters considered are $m = g = l = 1$, with the corresponding units. After substituting the values of the parameters and the points of linearization, the discrete models are given by

$$A_{d1} = \begin{bmatrix} 9.9996\text{x}10^{-1} & 9.9999\text{x}10^{-3} \\ -7.071\text{x}10^{-3} & 9.9996\text{x}10^{-1} \end{bmatrix}$$
$$B_{d1} = \begin{bmatrix} 5\text{x}10^{-5} \\ 9.9999\text{x}10^{-3} \end{bmatrix} \tag{8.22}$$

and

$$A_{d2} = \begin{bmatrix} 9.9995\text{x}10^{-1} & 9.9998\text{x}10^{-3} \\ -9.9998\text{x}10^{-3} & 9.9995\text{x}10^{-1} \end{bmatrix}$$
$$B_{d2} = \begin{bmatrix} 5\text{x}10^{-5} \\ 9.9998\text{x}10^{-3} \end{bmatrix}. \tag{8.23}$$

The responses of the two linear and the original nonlinear models are presented in Figure 8.2. The initial condition was $q(0) = 1\ rad$ at rest. Notice that the two linear models produce slightly different responses due to linearization. The nonlinear system was implemented as a continuous system in Simnon.

The details on the construction of the fuzzy coordinator are presented in the following section.

8.6.2 Construction of the Fuzzy Coordinator

During the design of the fuzzy coordinator it was noticed that the distribution of the models in terms of the fuzzy sets of inputs was exactly the same as in the tank example, that is, the table of fuzzy sets and selection variables was the same. The scaling factors for the selection of the membership functions were also the same. For these reasons, the fuzzy coordinator used was the same as in the tank example. The only difference was that the inputs were the position and velocity of the distributed model.

There are many modeling and control situations that utilize two local models, or controllers, as in the tank and the present one-link robot example. It is considered an advantage that the same fuzzy coordinator can be used due to the similarity in the decision-making process. This avoids going through the process of training. It is not surprising that a trained fuzzy coordinator can be used in different applications

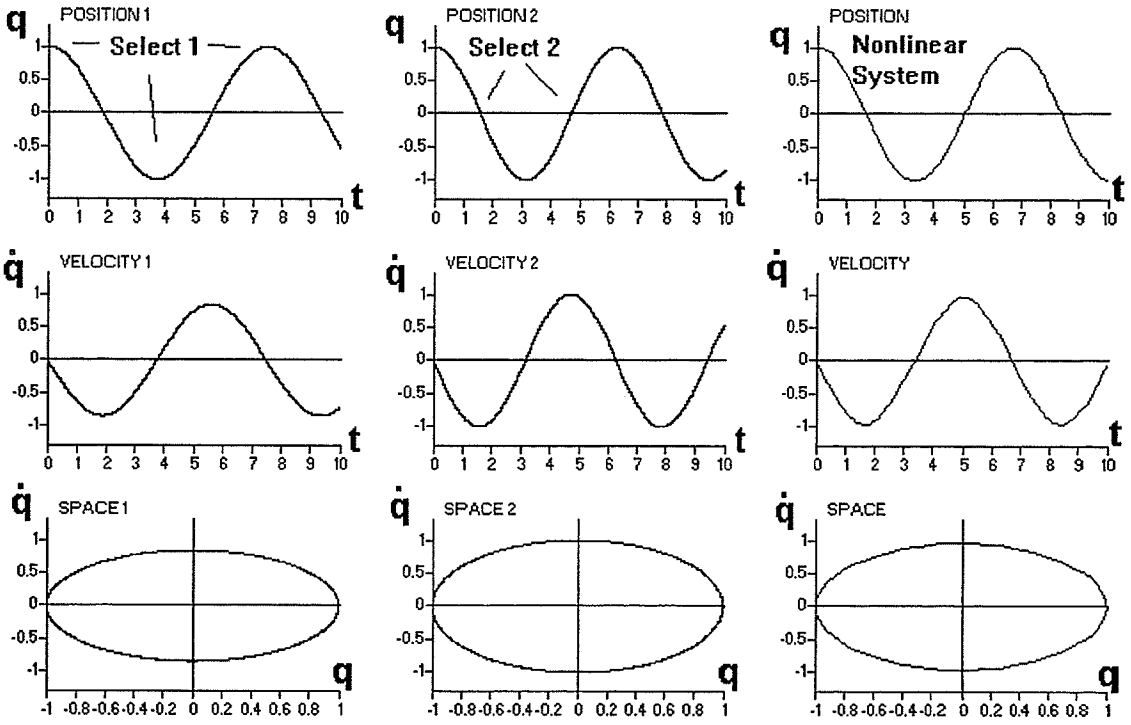

Figure 8.2: Models 1, 2 and Nonlinear.

since it is a nonlinear static mapping trained to emulate the decision-making process of selection between local elements. Thus, a fuzzy coordinator calculated for one application can be used in a different application, as long as the criteria to choose between the different local elements and the scaling factors of the variables are the same.

This does not mean that the fuzzy coordinator is unique, in fact, the same training algorithm with the same data can produce different sets of weights. Nevertheless, it is an advantage that a decision-making system can be used in different applications.

The following sections presents the results of the simulations of the distributed model.

8.6.3 Response of the Distributed Model

The response of the distributed model for the same initial condition as in Figure 8.2 is presented in Figure 8.3. The regions of operation of the two linear models have been indicated. The selection variables have also been plotted. Notice that the selection variables follow a sine-like function, and that the two variables move in opposite directions. This behaviour was expected since when one model is closer to its point of operation, the selection variable has a higher value, and the other model becomes less active. Comparing this response to the response of the nonlinear system in Figure 8.2, it can be seen that the distributed model presents a delay due to the discretization process, but the amplitude and frequency are the same.

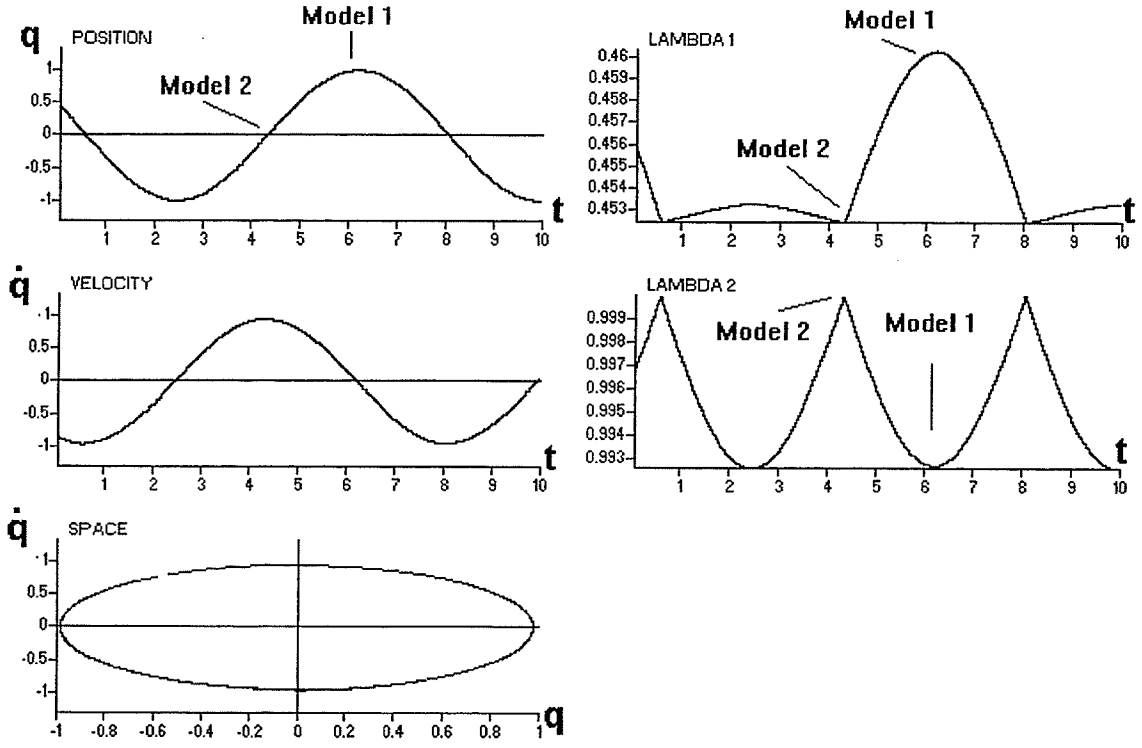

Figure 8.3: Distributed Model Response.

In the following section, the distributed model is controlled with two optimal controllers following the approach represented by Equation 8.4.

8.6.4 Control of the Distributed Model

Two discrete LQR optimal controllers were designed to control the two models. The approach of two model-controller pairs was used. The gains of the controllers were calculated using the algorithm of Chapter 5 with a command that was written in Matlab. The two gain vectors were

$$\begin{aligned} F_1 &= [-5.0893\text{x}10^{-1} \quad -1.4191] \\ F_2 &= [-4.0468\text{x}10^{-1} \quad -1.3451]\,. \end{aligned} \tag{8.24}$$

The step response of the system is presented in Figure 8.4. The initial condition and the reference step were

$$\begin{aligned} q(0) &= 1\ rad \\ q_{ref} &= -0.5\ rad. \end{aligned} \tag{8.25}$$

The values of the velocity and selection variables have also been plotted. Note the pattern of motion of the selection variables. They behave in a complementary way, as expected. Notice that the curve in the error space goes to zero.

8.6.5 Summary

The application of the distributed modeling technique has been presented. A nonlinear single-link robot was considered and two linear models were calculated for specific operating points. The two models were implemented as discrete systems.

The distributed model behaved as expected. The response presented a delay due to the discretization process but with the same frequency and amplitude as the nonlinear system. It was also shown that this model can be controlled with two very simple controllers applied to each of the local models.

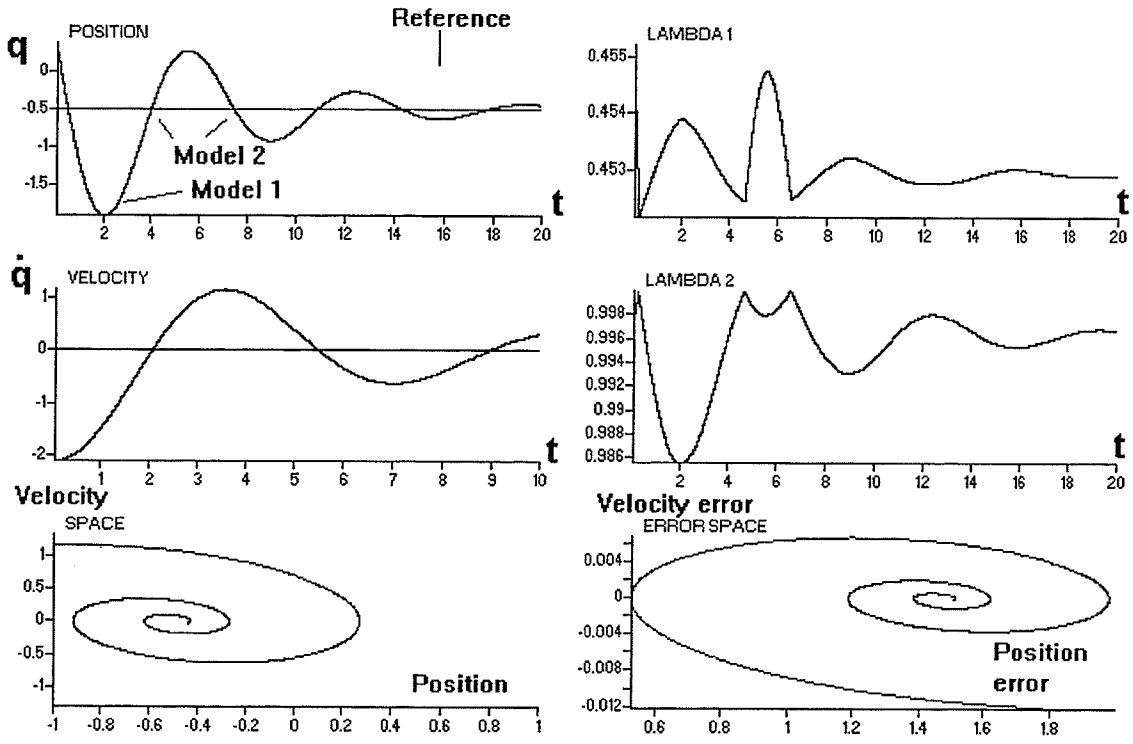

Figure 8.4: Control of the Distributed Model.

Chapter 9

Conclusions

In this thesis, the problem of modeling and control of complex systems was studied. The motivation was to explore the application of techniques from artificial intelligence in control loops. Several architectures have been reported in the past, where, the intelligent system was used directly as a controller, or it was used to tune the parameters of a standard controller. These architectures work well for simple applications. However, there are problems when the plant is nonlinear, and, in general, the couplings between the parameters of a simple controller cause deterioration of the performance, when the parameters are adjusted online.

A new approach was considered in this thesis. The approach consisted of using an intelligent system to coordinate the operation of a number of local modules. This is a more natural application of artificial intelligence in control, since, in order to coordinate the operation of several modules, it is required to make decisions based on measurements of the performance of the system. The switching of the local modules was based on the concept of dynamic fuzzy singletons, which consists of encoding the local modules as fuzzy sets.

A hierarchical structure designed using qualitative information was proposed. It was used to coordinate the operation of a number of local modules. The architecture was applied in control and modeling problems with good results, as showed in the application examples. It was shown that the architecture can be used in both, single and multivariable systems. Particularly, the proposed system can be applied in dynamic control of robots in tracking problems. Future work can include the application of the architecture to a real process. This might result in improvements of the learning and switching algorithms.

There were several options for the design of the coordinator. A fuzzy logic proces-

sor was chosen for its capacity to learn nonlinear mappings, and because qualitative information can be encoded using fuzzy logic. Several analytical results were presented for the logic processor. The most important being a Lemma to calculate the number of single processing elements for an implementation using analog electronics. A design method was proposed which consisted of the encoding of qualitative knowledge about the selection of the modules. The design method was based on experimentation and proper knowledge of selection policies. Future work can be devoted to the exploration or development of new learning and encoding techniques for logic processors, and for other systems.

The hierarchical architecture was also applied to the design of distributed models for control applications. These models are useful when a mathematical model that covers the whole range of operation of the system is not available. The modeling architecture was designed based on a coordinator and a number of local models. In some cases, the system is described by data sequences or functions. Several methods were developed for piece-wise linearization of systems, described by functions, data sequences, differential and difference equations. An application example was presented, which showed that a robot manipulator can be controlled successfully using a distributed modeling and control architecture.

The incorporation of qualitative information is a key factor in the success of the hierarchical system. Future work can include the development of new techniques. It is important to encode both, qualitative and quantitative information. Traditional control systems only use quantitative information, and as this thesis shows, there are ways to make use of qualitative information for a better performance.

The implementation in simulation using standard control algorithms makes the resulting systems modular and easy to adapt to existing control loops. In this thesis, PID and optimal controllers were considered. They were chosen since they are the most common algorithms in applications. Future work can include the inclusion of other control algorithms.

This thesis shows that techniques from artificial intelligence are a valuable resource in the design of control systems. The proposed architecture presents an excellent performance, which cannot be achieved with simple controllers alone, nor intelligent systems alone. The thesis shows that a balanced approach can simplify the design and improve the performance.

Appendix A

Defuzzification Methods

Some of the existing defuzzification methods are included in this appendix. These methods have been taken from Le Coeur [34].

For all the methods, consider a generic element u in a fuzzy set which exists in the universe of discourse U.

Method of the Surface

The method of the surface is the most utilized method. It is convenient in fuzzy inferences of the type Mamdani or Larsen [34], [49]. The defuzzified value u_o is given by

$$u_o = \frac{\sum \mu_i S_i}{S_i} \tag{A.1}$$

where μ_i is the membership function of the ith fuzzy set, and S_i its surface.

Center of Gravity Method

This method is more difficult to implement, in practice, it is often approximated by the method of the surface. It is considered that all the fuzzy sets i are contained in the envelope μ', and the defuzzified value is given by

$$u_o = \frac{\int u\mu'(u)\ du}{\int \mu'(u)\ du} \tag{A.2}$$

where u is the element in the fuzzy set. The following approximation for this method will be considered for this thesis. The output singleton is calculated by taking the

weighted average of c_j with respect to the degrees of activation λ^i in R^i

$$u_j = \frac{\sum_{i=1}^{n} \lambda^i c_j^i}{\sum_{i=1}^{n} \lambda^i}. \tag{A.3}$$

Method of the Height

This method does not take into account the shape of the fuzzy sets chosen by the inference engine for the output. It makes direct use of the degrees of certainty associated with each fuzzy set. This method is used with the inferences of Mamdani and Larsen. Notice the similarity with Sugeno's method below. The defuzzified value is given by the following equation

$$u_o = \frac{\sum u_i h_i}{\sum h_i} \tag{A.4}$$

where h_i is the maximum height of fuzzy set i.

Method of the Maximum Height

This method considers the values of u that maximize $\mu'(u)$ for all i. The output is given by

$$u_o = \frac{\sum u_i h_i}{h_{max}} \tag{A.5}$$

where h_{max} is the maximum height of all the fuzzy sets chosen for the output. Again this method is used in inferences of Mamdani and Larsen.

Method of the Length

This method is used with inferences like Lukasiewicz, Goguen, and Rescher [34], which use fuzzy algebraic operators instead of the conventional algebraic product. It is derived from the method of the height. Let l_i be the length of the maximum of fuzzy set i, in other terms, $l_i = u_2 - u_1$, where u_2 and u_1 are such that $\mu'(u) = 1$, for all $u \in [u_1, u_2]$. The defuzzified value is given by

$$u_o = \frac{\sum u_i(1 - l_i)}{\sum(1 - l_i)}. \tag{A.6}$$

Method of the Maximum Length

This defuzzification method is also used with fuzzy inferences. It is derived from the second method of the maximum height. The output is calculated as follows

$$u_o = l_{max} \frac{\sum u_i}{l_i} \tag{A.7}$$

where l_{max} is the maximum length of the fuzzy sets.

Second Method of the Height

This method is also used with fuzzy inferences. The output value is given by

$$u_o = \frac{\sum u_i(1 - h_i)}{\sum(1 - h_i)} \tag{A.8}$$

where h_i represents the minimum value of the membership function.

Sugeno's Method

In this method the idea is to suppress the inference stage by replacing the table of rules by real numbers. These numbers are typically the centers of the output fuzzy sets. The output is given by

$$u_o = \sum u_i a_i \tag{A.9}$$

where a_i is the degree of certainty associated with fuzzy set i.

Appendix B

Functions in the Plane

The algorithm presented here was developed to provide a mechanism for finding the parameters and the number of straight lines that are required to linearize a function $x(t)$ with a given margin of error in a given interval of its domain. The following assumptions are considered throughout the discussion

- $x(t) = f(h(t))$ is a continuous function, where $h(t)$, the domain function of f, is a diffeomorphism (differentiable mapping with differentiable inverse) on an open set of R.
- If f is parameterized by h, other than $h(t) = t$, then the discussion applies for a reparameterization f' (not a derivative) by t. It is well known that f' can always be found (O'Neill [46]).
- f is a diffeomorphism $f : R \to R$.

The linearization problem can be stated as follows: find the 'n' straight lines ($n \in R$, that is, finite) that form a piece-wise linearization (approximation) of $x = f(t)$ over the interval $t \in [t_{min}, t_{max}]$ with a maximum error $\epsilon > 0$.

The following algorithm is proposed as a solution. It is a recursive algorithm that starts at t_{min}, calculates a tangent, and then finds the next tangent intersecting the previous at a point where the error of approximation is ϵ. This algorithm produces a set of n straight lines with the corresponding intervals of application.

Algorithm 1

- 1. Initial linearization at t_{min}: let $n = 1$, $x_1(t) = f'(t_{min})(t - t_{min}) + f(t_{min})$, set step size to $\xi = c\epsilon$.

- 2. Next linearization at $t_{min}+\xi$: $x_n(t) = f'(t_{min}+\xi)(t-t_{min}-\xi)+f(t_{min}+\xi)$. Is $t_{min}+\xi > t_{max}$?:

  Yes, go to End.

  No, go to 3.

- 3. Calculating intersection: If $f'(t_{min}) = 0$ and $f'(t_{min}+\xi) = 0$, then go to 5 (since intersection does not exist; a plateau).

$$\begin{aligned} t_I &= \frac{-f'(t_{min}+\xi)g(t_{min}) + f'(t_{min})g(t_{min}+\xi)}{-f'(t_{min}) + f'(t_{min}+\xi)} \\ x_I &= \frac{-g(t_{min}) + g(t_{min}+\xi)}{-f'(t_{min}) + f'(t_{min}+\xi)} \end{aligned} \tag{B.1}$$

  where $g(t) = f'(t)t - f(t)$.

- 4. Calculate error: $e = x_I - x(t)$. Let $d = \epsilon/c$.

- 5. Loop: Is $e < \epsilon$?

  Yes: $t_{min}+\xi = t_{min}+\xi+\Delta$, go to 2.

  No: If $e > \epsilon$, then $t_{min}+\xi = t_{min}+\xi-\Delta$, go to 2; else store $x_n(t)$ and $t_{min}+\xi$, let $t_{min} = t_{min}+\xi$, $n = n+1$, and go to 2.

- End. The number of straight lines $x_n(t)$ that approximate the function $x(t)$ over the interval $[t_{min}, t_{max}]$ is n.

Appendix C

Geometric Linearization in the Space

This algorithm presented here was developed as an extension of the one presented for two-dimensional functions in Appendix B. The problem is to find a piece-wise approximation of a curve α in a three-dimensional space. The following considerations are made.

- $\alpha(t) = (x(\bar{z}), y(\bar{z}), \bar{z})$ is a continuous function of z, where $\bar{z} = h(z)$, the domain function of α, is a diffeomorphism (differentiable mapping with differentiable inverse) on an open set of R.
- α is re-parameterized by $z = h^{-1}(t)$, thus $\bar{z}$ becomes the independent variable.
- The discussion applies to an α', the re-parameterization without loss of generality.
- The approximation must be piece-wise continuous.

The linearization problem can be stated as follows: find the 'n' straight lines ($n \in R$, that is, finite) that form a piece-wise linearization (approximation) of $\alpha(t)$ over the interval $t \in [t_{min}, t_{max}]$ with a maximum error $\epsilon > 0$.

The following algorithm is proposed as a solution. It is a recursive algorithm that exploits differential geometrical properties of the curve in calculating the piece-wise continuous approximation.

The algorithm starts at t_{min}, calculating the Frenet Frame (see O'Neill [46]) at $\alpha(t_{min})$, then the first straight line is calculated in the direction of the unit tangent.

The next tangent at $t_{min} + \xi$ and the corresponding Frenet frame are calculated. If the two lines do not intersect, the second is rotated in the direction of the difference between the binormal vector fields until they do intersect. The error is then calculated at the intersection point, which is the maximum error given the smoothness of the curve. If the error is not ϵ, then the independent variable is incremented or decremented, recalculating the second linearization until the error is ϵ. The algorithm produces a set of n straight lines with the corresponding intervals of application.

Algorithm 2

- 1. Unit speed curve: Reparameterize the curve $\alpha(t)$ to $\beta(s) = \alpha(h(s))$, where $\|\beta'\| = 1$. Without loss of generality it is assumed that $\alpha(t)$ is a unit speed curve.

- 2. Frenet Frame at t:

$$\begin{array}{ll} \text{Unit Tangent:} & T = \alpha' \\ \text{Principal Normal Vector Field:} & N = T'/k \\ \text{Binormal Vector Field:} & B = T \text{x} N \\ \text{Curvature } k: & k = \|T'\| \\ \text{Torsion } \tau: & B' = -\tau N \end{array} \qquad \text{(C.1)}$$

  where T is the direction of linearization, N is the direction of turning of the frame, B is the direction of turning of the tangent plane, k measures the deviation from straightness, and τ measures how α leaves the tangent plane.

- 3. First linearization at t_{min}:

  $\lambda_1(t) = \alpha'(t_{min})\,(t - t_{min}) + \alpha(t_{min})$.

  Frenet Frame: $T(t_{min})$, $N(t_{min})$, $B(t_{min})$.

- 4. Next linearization at $t_{min} + \xi$:

  $\lambda_n(t) = \alpha'(t_{min} + \xi)\,(t - t_{min} - \xi) + \alpha(t_{min} + \xi)$.

  Frenet Frame: $T(t_{min} + \xi)$, $N(t_{min} + \xi)$, $B(t_{min} + \xi)$.

- 5. Intersection: Calculating point of intersection between λ_{n-1} and λ_n.
  a) Rewrite the equations of the straight lines as follows:

  $\lambda_{n-1}(t) = (a_1 t + c_1, a_2 t + d_1, t)$, and $\lambda_n(t) = (b_1 t + c_2, b_2 t + d_2, t)$.

Form the systems of equations:

$$A\begin{bmatrix} x \\ t \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \qquad B\begin{bmatrix} y \\ t \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{C.2}$$

where $A = \begin{bmatrix} 1 & -a_1 \\ 1 & -b_1 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & -a_2 \\ 1 & -b_2 \end{bmatrix}$.

Next, check solvability of the system of equations:

Is A or B singular (det()=0)?

Yes: go to b) to rotate Frame n towards Frame $n-1$;

No: solving equations:

$$\begin{bmatrix} x \\ t \end{bmatrix} = \frac{1}{a_1 - b_1}\begin{bmatrix} -b_1 & a_1 \\ -1 & b_1 \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{C.3}$$

$$\begin{bmatrix} y \\ t \end{bmatrix} = \frac{1}{a_2 - b_2}\begin{bmatrix} -b_2 & a_2 \\ -1 & b_2 \end{bmatrix}\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{C.4}$$

The intersection must occur at the same point in the domain for the two projections:

Is $\dfrac{b_1c_2 - c_1}{a_1 - b_1} = \dfrac{b_2d_2 - d_1}{a_2 - b_2}$?

Yes: The intersection point is:

$$(x, y, t)_I = \left(\frac{a_1c_2 - b_1c_1}{a_1 - b_1}, \frac{a_2d_1 - b_2d_2}{a_1 - b_1}, \frac{b_1c_2 - c_1}{a_1 - b_1}\right) \tag{C.5}$$

go to 6.

No: go to rotation of the Frame in b).

b) Calculating direction of rotation and the new Frame $\bar{T}_n$, $\bar{B}_n$ around N_n.

Direction: $B_{(n,n-1)} = B_n - B_{n-1}$

Projection on tangent plane n: $B_p = \|B_{(n,n-1)}\| \sin\left(\dfrac{< B_{(n,n-1)}, N_n >}{\|B_{(n,n-1)}\|}\right) B_n$.

Rotated Frame: $\bar{T}_n = T_n + B_p$ and $\bar{B}_n = \bar{T}_n \mathrm{x} N_n$.

c) Calculating rotated linearization: The equation of the line in the rotated Frame $\bar{T}_n$, N_n, $\bar{B}_n$ is given by the following expression:

$$\bar{\lambda}_n(t) = \bar{T}_n\,(t - t_{min} - \xi) + \int_o^{t_{min}+\xi} \bar{T}_n \mathrm{dt} \tag{C.6}$$

Let $\lambda_n(t) = \bar{\lambda}_n(t)$ and go to a).

- 6. Error: $e = \|(x, y, t)_I - \alpha(t_I)\|$.
- 7. Loop: Is $e < \epsilon$? Yes: $t_{min} + \xi = t_{min} + \xi + d$, go to 4. No: If $e > \epsilon$, then $t_{min} + \xi = t_{min} + \xi - d$, go to 4; else store $\alpha_n(t)$ and $t_{min} + \xi$, $t_{min} = t_{min} + \xi$, $n = n + 1$, go to 4.
- End. n is the final number of straight lines $\alpha_n(t)$ that approximate the curve $\alpha(t)$ on the interval $[t_{min}, t_{max}]$.

Appendix D

Geometric Linearization via Curvature

The method presented here was developed as a modification of the one introduced for functions in the plane in Appendix C. The geometric concept of curvature is introduced in order to measure the deviation of straightness of the function. A given tangent line is valid as an approximation until the curvature of the function is greater than a certain pre-specified value.

The algorithm calculates a tangent line, then moves along the domain of the function, checking the value of the curvature function. At this point, a new tangent line is calculated. The algorithm is introduced by example.

Example: Sine Function

The problem is to linearize the function $y(t) = \sin(t)$ over the domain $t \in [0, \pi]$ for a maximum deviation in curvature of ξ. A tangent line at t_o is given by

$$y_o(t) = y'(t_o)\,(t - t_o) + y(t_o) \tag{D.1}$$

for $y'(t) = \cos(t)$. The curvature is given by $k_o = \|y'(t_o)\|$. Let $t_{min} = 0$ and $t_{max} = \pi$.

Algorithm 3

- 1. Initial linearization at t_o: $y_1(t) = y'(0)\,(t - 0) + y(0)$, for $y'(0) = 1$. The curvature at t_o is given by $k_1 = \|y'(0)\|$. The following represents the initial

linearization.

$$t_o = 0 \qquad k_1 = 1 \qquad y_1(t) = t \tag{D.2}$$

Let $t_{old} = t_o$. The next step is to find the relation curvature-increment on the independent variable. Define: $\Delta k = k_1 - k(t_{new}) = \gamma\xi$, for a constant $\gamma > 0$, where t_{new} is the point of next linearization. Let $\gamma = 10$. Since K_1 and Δk are known and $k >= 0$, then $k(t_{new}) = |k_1 - \gamma\xi|$. Next, calculate the increment Δt to find $t_{new} = t_{old} + \Delta t$. Let $m = 0$.

Algorithm 4 Calculation of Δt.

a) $m = m + 1$, increment t_{old} by $\Delta t = \delta\xi m$, for a constant $\delta > 0$.

Let $\delta = 2$, then $\Delta t = 2\xi$. Calculate $k(t_{new}) = \|y(t_{new})\|$.

Is $k(t_{new}) >= k_1 - \gamma\xi$?

Yes: $t_{new} = t_o + m\delta\xi$. Let $t_o = t_{new}$ and go to next step.

No: go to a).

- 2. Next curvature: The maximum acceptable value of curvature is $k_1 - \gamma\xi = 0.5$, and combined with $\delta = 2$ then $\Delta t = m0.1$. The curvature at the point of next linearization is then $k(t_{old} + m0.1) = \|\cos(t_{old} + m0.1)\| = 0.5$, also written as $k_2 = \|\cos(t_{new}\| = 0.5$. The value of t_{new} can be estimated from k_2, and this is $t_{new} = t_{old} + m0.1 = \pi/3$, and therefore $m = (\pi/3 - 0)/0.1 = 10$. This value could have also been found after 10 iterations of Algorithm 4. The value can be easily verified since 10 sections are required to cover the whole interval.

  Let $t_{old} = t_o$. The linearization is $y_n(t) = \cos(\pi/3)\,(t - \pi/3) + y(\pi/3)$, which gives $y_n = 0.5t + 0.342$. It is represented by the following:

$$t_o = \pi/3 \qquad k_2 = 0.5 \qquad y_2(t) = \frac{1}{2}t + \frac{3\sqrt{3} - \pi}{6} \tag{D.3}$$

- 3. Next linearization: Using the same reasoning as in 1a) and 2, yields $k_3 = |k_2 - \delta\xi| = 0$, and $t_{new} = t_{old} + m0.1 = \pi/2$, for $m = 10$ iterations. Let $t_o = t_{new}$, $t_{old} = \pi/2$, and the linearization is $y_n(t) = \cos(\pi/2)\,(t - \pi/2) + \sin(\pi/2)$. It is represented by the following:

$$t_o = \pi/2 \qquad k_3 = 0 \qquad y_3(t) = 1 \tag{D.4}$$

- 4. Next linearization: As before $k_4 = |k_3 - \delta\xi| = 0.5$ and the next value is given by $t_{new} = t_{old} + m0.1 = 2\pi/3$. Let $t_o = t_{new}$. The resulting approximating function is $y_4(t) = \cos(2\pi/3)\,(t - 2\pi/3) + \sin(2\pi/3)$ $(= -0.5t + 1.913)$. Thus

$$t_o = 2\pi/3 \qquad k_4 = 0.5 \qquad y_4(t) = -\frac{1}{2}t + \frac{2\pi + \sqrt{3}}{6} \tag{D.5}$$

- 5. Next linearization: $k_5 = |k_4 - \delta\xi| = \|cos(t_{new})\| = 1$, for $t_{new} = \pi$. Therefore

$$t_o = \pi \qquad k_5 = 1 \qquad y_5(t) = -t + \pi \tag{D.6}$$

- End. The solution is the set $\{y_1, \cdots, y_5\}$ at $t_o \in \{0, \pi/3, \pi/2, 2\pi/3, \pi\}$

Appendix E

Linearization of Data Sequences

The method presented here was developed to generate a piece-wise linear approximation of a data sequence. Data sequences are often used in statistical control. The linearization problem is to find the set of 'n' straight lines ($n \in R$) in the space that form a piece-wise linear approximation of the data sequence $\{p_1, p_2, \cdots\}$, where $p_i \in R^3$. The following algorithm is proposed as a solution. The data sequence can be infinite, although in applications, a way to stop the algorithm must be specified.

Define an angle of tolerance ψ representing how smooth the approximation must be. For large values of ψ the algorithm will determine a smaller number of straight lines than for a small value. Let $a = 1$ be a constant that initializes the algorithm, $u_1 = 0$, and p_1 be the first point of the sequence.

Algorithm 5

- 1. Let p_2 be the next point of the sequence.

  If p_3 =*empty* then go to *End* else proceed as follows. Let $p_1 = (y_1, z_1, u_1)$ and $p_2 = (y_2, z_2, u_2)$.

- 2. Calculate the line passing by p_1 and p_2:

$$\lambda_{12} = (m_{y_{12}}(u - u_1) + y_1, m_{z_{12}}(u - u_1) + z_1, u) \tag{E.1}$$

  where

$$m_{y_{12}} = \frac{y_2 - y_1}{u_2 - u_1}$$
$$m_{z_{12}} = \frac{z_2 - z_1}{u_2 - u_1} \tag{E.2}$$

Shift to the origin and calculate a unit vector as follows.

Shifted Equation and Unit Vector:

$$\bar{\lambda}_{12} = (m_{y_{12}}u, m_{z_{12}}u, u)$$
$$v_{12} = \left(\frac{m_{y_{12}}}{d_{12}}, \frac{m_{z_{12}}}{d_{12}}, \frac{1}{d_{12}}\right) \tag{E.3}$$

where $d_{12} = \sqrt{m_{y_{12}}^2 + m_{z_{12}}^2 + 1}$ for $((m_{y_{12}}u)^2 + (m_{y_{12}}u)^2 + u^2)^{1/2} = 1$.

Store v_{12}.

- 3. Let the next point be $p_3 = (y_3, z_3, u_3)$.
- 4. Calculate the line passing by p_1 and p_3:

$$\lambda_{13} = (m_{y_{13}}(u - u_1) + y_1, m_{z_{13}}(u - u_1) + z_1, u) \tag{E.4}$$

for

$$m_{y_{13}} = \frac{y_3 - y_1}{u_3 - u_1}$$
$$m_{z_{13}} = \frac{z_3 - z_1}{u_3 - u_1}. \tag{E.5}$$

Calculate $\bar{\lambda}_{13}$ and v_{13} as in step *3*.

- 5. Calculate the angle between λ_{12} and λ_{13}. Let $\theta_{23} = \cos^{-1}(v_{12} \cdot v_{13})$.

The angle is $\theta = \theta_{23}$, let $m = 3$.

- 6. Check the angle. Do the following while the end of the sequence is not reached.

Is $\theta < \psi$?

No: go to step *7*.

Yes: check other angles until the end of the file of stored vectors

$\theta_{n(m+1)} = \cos^{-1}(v_{1(n-1)} \cdot v_{1(m+1)})$, let $\theta = \theta_{n(m+1)}$.

At EOF: take p_1 and $p_3 = p_{m+1}$, and go to step *4*.

- 7. Calculate the line that satisfies the *least squares* criterion to approximate the following points

$$\begin{aligned}
&(m_y u_1 + b_y, m_z u_1 + b_z, u_1) = (y_1, z_1, u_1) \\
&(m_y u_2 + b_y, m_z u_2 + b_z, u_2) = (y_2, z_2, u_2) \\
&\qquad\vdots \\
&(m_y u_m + b_y, m_z u_m + b_z, u_m) = (y_m, z_m, u_m)
\end{aligned} \tag{E.6}$$

The first linear approximation is the *least squares* solution of these equations given by

$$\begin{bmatrix} m_y \\ b_y \end{bmatrix} = \left(U_m^T U_m\right)^{-1} U_m^T Y_m \tag{E.7}$$

$$\begin{bmatrix} m_z \\ b_z \end{bmatrix} = \left(U_m^T U_m\right)^{-1} U_m^T Z_m \tag{E.8}$$

where

$$U_m = \begin{bmatrix} u_1 & 1 \\ u_2 & 1 \\ \vdots & \vdots \\ u_m & 1 \end{bmatrix} \qquad Y_m = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \qquad Z_m = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} \tag{E.9}$$

Store (u_1, u_m, λ_1). The linear approximation is

$\lambda_1 = (m_y u + b_y, m_z u + b_z, u)$,

Reset p_1 and consider the previous p_m: let $p_1 = p_m$, $a = a + 1$.

Go to step *1*.

- End. The result is a collection of 'a' triplets (u_1, u_m, λ_1).

Appendix F

Linearization of Difference Equations

Consider a nonlinear autonomous difference equation (*nlade*)

$$x_{k+1} = f(x_k) \qquad x_{k_o} = x_o \in R \tag{F.1}$$

where $k \in Z$. Notice that the domain of f can be interpreted as $f : R \rightarrow R$ or $f : Z \rightarrow R$. In fact, this can be expressed as $f : Z \rightarrow R \rightarrow R$. This interpretation results in two cases of linearization:

- *Case 1:* Linearization around a given value of the independent variable.
- *Case 2:* Linearization around a given value of the state variable.

Next, the developed solutions to these problems are presented. For the first case, the algorithm presented for curves in the space is used, and for the second case, the solution is rather trivial.

Case 1:

The *nlade* can be expressed as a curve $\alpha(k)$ in the three-dimensional space

$$dom(x_k) \text{x} dom(x_{k+1}) \text{x} dom(k)$$

(which is equivalent to $R\text{x}R\text{x}Z$) as follows:

$$\alpha(k) = (x_k, k_{k+1}, k) = (x_k, f(x_k), k)\,. \tag{F.2}$$

The derivative $\alpha'(k)$ can be a backward, forward, or rectangular approximation (Astrom [4]). A backward approximation is recommended for this algorithm. It is given by

$$\alpha'(k) = \frac{\alpha(k) - \alpha(k-1)}{k}. \tag{F.3}$$

The proposed linear approximation of the difference equation around k_o is then

$$\lambda(k) = \alpha'(k_o)\,(k - k_o) + \alpha(k_o). \tag{F.4}$$

From this, the piece-wise linearization can be calculated using Algorithm 2 for curves in the space of Appendix C.

Example: $x_{k+1} = \sin(x_k) + b$

Let $x_o \in R$ be the initial condition and $b \in R$ a constant. The curve and its derivative are given by

$$\begin{aligned} \alpha(k) &= (x_k, \sin(x_k) + b, k) \\ \alpha'(k) &= \left(\frac{x_k - x_{k-1}}{k}, \frac{\sin(x_k) - \sin(x_{k-1})}{k}, \frac{1}{k} \right). \end{aligned} \tag{F.5}$$

The linear approximation at k_o is given by

$$\lambda(k) = (\lambda_x, \lambda_y, \lambda_z) \tag{F.6}$$

where

$$\begin{aligned} \lambda_x &= \left(\frac{x_{k_o} - x_{k_o-1}}{k_o} \right)(k - k_o) + x_{k_o} \\ \lambda_y &= \left(\frac{\sin(x_{k_o}) - \sin(x_{k_o-1})}{k_o} \right)(k - k_o) + \sin(x_{k_o}) + b \\ \lambda_z &= \frac{k - k_o}{k_o} + k_o. \end{aligned} \tag{F.7}$$

The linearized equation is then

$$y(k) = \left(\frac{\sin(x_{k_o}) - \sin(x_{k_o-1})}{k_o} \right)(k - k_o) + \sin(x_{k_o}) + b. \tag{F.8}$$

Let $c_o + a_o k_o = \sin(x_{k_o}) + b$, for

$$a_o = \left(\frac{\sin(x_{k_o}) - \sin(x_{k_o-1})}{k_o} \right). \tag{F.9}$$

Then the equation of the straight line is

$$y(k) = a_o k + c_o. \tag{F.10}$$

This equation is a linear approximation of the difference equation around a point of linearization k_o. Notice that $k_o \geq 1$, since the nonlinear difference equation is of first-order.

The difference equation can then be linearized in a range of operating points by calculating a set of approximations than can be activated by a coordinating system.

Case 2:

Since the linearization is around a value of the state variable x we need to consider a curve in the two-dimensional state plane, given by

$$\alpha(x) = (x_k, x_{k+1}). \tag{F.11}$$

Its first derivative is given by

$$\alpha'(x) = \frac{d}{dx_k}\alpha(x) = \left(1, \frac{d}{dx_k}x_{k+1}\right). \tag{F.12}$$

The linear approximation around x_o is then given by

$$\lambda(x) = \alpha'(x)\,(x - x_o) + \alpha(x_o). \tag{F.13}$$

The point of linearization x_o must be a valid value; in other words, the curve must pass by or tend to it, otherwise the linearization will not make sense. This is represented by $x_o \in \alpha(x)$.

Notice that x_o is the point of linearization, not the initial condition. For example, consider that $\alpha(x)$ is a feedback system, where it is required that $x \to 0$, then let $x_o = 0$.

Example: $x_{k+1} = \sin(x_k) + b$

Let $x(0) \in R$ be the initial condition and $b \in R$ a constant. The curve and its derivative are given by

$$\begin{aligned} \alpha(k) &= (x_k, \sin(x_k) + b) \\ \alpha'(k) &= (1, \cos(x_k)). \end{aligned} \tag{F.14}$$

The linear approximation at x_o is

$$\lambda(x) = (x, \cos(x_o)\,(x - x_o) + \sin(x_o) + b). \tag{F.15}$$

The linearization is then

$$y(x) = \cos(x_o)\,(x - x_o) + \sin(x_o) + b. \tag{F.16}$$

Let $a_o = \cos(x_o)$ and $c_o + a_o x_o = \sin(x_o) + b$. The straight line can be written as

$$y(x) = a_o x + c_o. \tag{F.17}$$

An important observation is that the approximation $\lambda(x)$ is very sensitive to the point of linearization x_o.

Appendix G

Linearization of Differential Equations

In this appendix, a survey of existing methods for linearization of differential equations is presented.

- *1. TAYLOR'S EXPANSION & CONTINUITY:* (Kailath [29]) Consider the system $\dot{x}(t) = f(x(t))$, where x_o is an equilibrium point of the system, that is, $f(x_{e\leq}) = 0$. Consider $z(t)$ as a small deviation from the equilibrium point, then the Taylor's expansion around x_o is given by the following equation for $x(t) = z(t) + x_o$

$$f(z(t) + x_o) = f(x_o) + \frac{\delta}{\delta x} f(\xi) \mid_{\xi = x_o} z(t) + g(z(t)). \tag{G.1}$$

  By continuous differentiability (which implies Lipchitz continuity), then

$$\frac{\|g(z(t))\|}{\|z(t)\|} \to 0 \qquad \text{as } \|z(t)\| \to 0. \tag{G.2}$$

  The linearized system is then

$$\dot{z}(t) = A\, z(t) + g(z(t)) \tag{G.3}$$

  where $A = \dfrac{\delta}{\delta x} f(x(t))$ at $x = x_o$.

- *2. GRAPHICAL EULER METHOD:* (Vidyasagar [78]) This method is used for 2nd order autonomous systems. The method consists of the calculation of

the slope s of the phase portrait diagram at every point of interest. The linear system is given by

$$\dot{x}(t) = s\,x(t). \tag{G.4}$$

- *3. TAYLOR'S EXPANSION FOR CONTROL SYSTEMS:* (Kailath [29]) This method is an extension of the Method presented in *1* for control systems. Consider the system $\dot{x}(t) = f(x(t), u(t), t)$. Let x_o be an equilibrium of the system, and $z(t)$ the state around x_o. The linearized equation around x_o is given by

$$\dot{z}(t) = A(t)\,z(t) + B(t)\,u(t) \tag{G.5}$$

where $A(t) = \dfrac{\delta}{\delta x} f(x,u,t)$ and $B(t) = \dfrac{\delta}{\delta u} f(x,u,t)$ at $x = x_o$ and $u = u_o$

- *4. MEAN VALUE THEOREM:* (Khalil [31]) This method is valid for autonomous and nonautonomous systems. Consider $\dot{x}(t) = f(x(t))$. Let the origin be an equilibrium and z be a point on the line segment connecting x to the origin. At this point, the following equation can be written

$$f(x) = \frac{\delta}{\delta x} f(x)\mid_{x=0} + \left[\frac{\delta}{\delta x} f(x)\mid_{x=z} - \frac{\delta}{\delta x} f(x)\mid_{x=0}\right] x = A\,x + g(x). \tag{G.6}$$

By Lipchitz continuity the following equation is obtained

$$\|g(x)\| \leq \|\frac{\delta}{\delta x} f(x)\mid_{x=z} - \frac{\delta}{\delta x} f(x)\mid_{x=0}\| \,\|x\| \tag{G.7}$$

and $\dfrac{\|g(x)\|}{\|x\|} \to 0$ as $\|x\| \to 0$. The linearized system is then

$$\dot{x}(t) = A\,x \tag{G.8}$$

where $A = \dfrac{\delta}{\delta x} f(x)\mid_{x=0}$.

- *5. CENTER MANIFOLD THEOREM:* (Khalil [31]) This method is used when standard linearization fails due to eigenvalues with positive or zero real parts. The method consists of a partial linearization of the equation via a linearization of a lower order system. The order of this system is equal to the number of eigenvalues of the standard linearization matrix A with zero real parts. The system can be decomposed by a homomorphism T as follows. Let $[y,z]^T = Tx$ be the new coordinates where the order of y is equal to the number of eigenvalues of A with zero real part. Consider a diagonalization A as follows

$$T\,A\,T^{-1} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}. \tag{G.9}$$

The system in the new coordinates is given by

$$\begin{bmatrix} \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} + \begin{bmatrix} g_1(y,z) \\ g_2(y,z) \end{bmatrix} \tag{G.10}$$

The theorem states that there exists an interval of y for which $z = h(y)$ is a center manifold, that is, $h(0) = 0$, $\frac{\delta}{\delta y} h \mid_0 = 0$. Thus it is only required to study the following equation by standard methods:

$$\dot{y} = A_1 y + g_1(y, h(y)). \tag{G.11}$$

- *6. DESCRIBING FUNCTION METHOD:* (Khalil [31]) This method consists of applying a change of variable to the system in order to express it as a linear system and the nonlinearities by their describing function. The Nyquist criterion can then be used to study the system.

- *7. EXACT LINEARIZATION:* (Khalil [31], Isidori [26], Nijmeijer [44]) This method is applied to multivariable systems and yields a canonical controllable form. The method is based on a dynamic feedback, where the nonlinear terms are cancelled by a proper change of coordinates. Consider the system

$$\dot{x} = f(x) + g(x)u \tag{G.12}$$

where f and g are nonlinear functions of x, and $g(x)^{-1}$ exists. Consider the feedback law

$$u = g(x)^{-1}[v - f(x)]. \tag{G.13}$$

The system is then reduced to $\dot{z} = Az + v$, where v is the new control input.

Appendix H

Exact Linearization of Nonlinear Systems

This appendix presents the method of feedback linearization. This method produces an exact linearization of a nonlinear system via a nonlinear change of coordinates.

For certain classes of nonlinear systems, it is possible to follow elegant procedures in order to simplify the equations without loosing accuracy. The technique considered in this appendix is a combination of differential geometry and nonlinear systems in the area of control theory known as geometric control (Isidori [26], Nijmeijer [44]).

The main idea is that a certain class of nonlinear systems can be exactly linearized around an operating point in the state space, via a local nonlinear change of coordinates. The method is based on the Frobenius Theorem and the concept of state-feedback. The resulting linear system is controllable (Kailath [29]), which allows the designer to arbitrarily place the closed-loop eigenvalues.

The resulting linear system can be controlled with any kind of algorithm locally, and by Lyapunov's First Theorem (Khalil [31]), the dynamic properties of the linear system are the same as the original nonlinear system.

Feedback Linearization

Consider a nonlinear system of the form

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}) + g(\mathbf{x})u(\mathbf{x}) \qquad\qquad \mathbf{x}(0) = \mathbf{x_0} \in R^n \\ y &= h(\mathbf{x}). \end{aligned} \tag{H.1}$$

The control tracking problem with qualitative and quantitative requirements can be solved by a fuzzy distributed control architecture based on a fuzzy system, and

local controllers based on the exact linearization of the nonlinear system around appropriate points of operation.

Before the general design method is introduced, the following definitions from Isidori [26] and Nijmeijer [44] are presented. These concepts are important since they serve as an introduction to the theory of feedback linearization in nonlinear control.

Definition 1: Lie Derivative.

Given a scalar-valued function $\lambda(\mathbf{x}) \in R$ and a n-vector-valued function $\mathbf{f}(\mathbf{x}) \in R^n$, the *Lie derivative* of $\lambda(\mathbf{x})$ along $\mathbf{f}(\mathbf{x})$ is given by

$$L_f \lambda(\mathbf{x}) =< \nabla\lambda(\mathbf{x}), \mathbf{f}(\mathbf{x}) > \tag{H.2}$$

where $\nabla \lambda(\mathbf{x})$ is the gradient of $\lambda(\mathbf{x})$.

Remarks: Repeated use of this derivative is possible, for instance

$$L_g \, L_f \, \lambda(\mathbf{x}) =< \nabla L_f \, \lambda(\mathbf{x}), \mathbf{g}(\mathbf{x}) > . \tag{H.3}$$

This operation can also be defined recursively as follows

$$L_f^k \, \lambda(\mathbf{x}) =< \nabla L_f^{k-1} \, \lambda(\mathbf{x}), \mathbf{f}(\mathbf{x}) > . \tag{H.4}$$

This operator is very useful to simplify notation.

Definition 2: Relative Degree.

The single-input single-output system H.1 has a relative degree r at $\mathbf{x_o}$ if

$$\begin{aligned} &L_g \, L_f^k \, h(\mathbf{x}) = 0 \\ &L_g \, L_f^{r-1} \, h(\mathbf{x_o}) \neq 0. \end{aligned} \tag{H.5}$$

where the first condition must be satisfied for all $\mathbf{x} \in B(\mathbf{x_o})$ and for all $k < r - 1$, $B(\mathbf{x_o})$ is a neighborhood of $\mathbf{x_o}$.

Definition 3: Lie Bracket.

Given two n-vector-valued functions $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$, the *Lie Bracket* of $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ is defined by

$$[\mathbf{f}, \mathbf{g}](\mathbf{x}) =< J(\mathbf{g}(\mathbf{x})), \mathbf{f}(\mathbf{x}) > - < J(\mathbf{f}(\mathbf{x})), \mathbf{g}(\mathbf{x}) > \tag{H.6}$$

where J is the jacobian operator.

Remarks: The following properties hold

$$\begin{aligned} &[\mathbf{f},\mathbf{g}]\,(\mathbf{x}) = -\,[\mathbf{g},\mathbf{f}]\,(\mathbf{x}) \\ &[\mathbf{f},\mathbf{f}]\,(\mathbf{x}) = 0 \\ &L_{[f,g]}\,\lambda(\mathbf{x}) = L_f\,L_g\,\lambda(\mathbf{x}) - L_g\,L_f\,\lambda(\mathbf{x}). \end{aligned} \tag{H.7}$$

The following notation will be used whenever a $\mathbf{g}(\mathbf{x})$ is repeatedly *bracketed* with $\mathbf{f}(\mathbf{x})$

$$\begin{aligned} &ad_f\,\mathbf{g}(\mathbf{x}) = [\mathbf{f},\mathbf{g}]\,(\mathbf{x}) \\ &ad_f^2\,\mathbf{g}(\mathbf{x}) = [\mathbf{f},[\mathbf{f},\mathbf{g}]]\,(\mathbf{x}) \\ &ad_f^k\,\mathbf{g}(\mathbf{x}) = \left[\mathbf{f}, ad_f^{k-1}\,\mathbf{g}\right](\mathbf{x}). \end{aligned} \tag{H.8}$$

Definition 4: Involutive Set.

A set of k n-vector-valued functions $(\mathbf{f_1}(\mathbf{x}),\cdots,\mathbf{f_k}(\mathbf{x}))$ such that the matrix given by $[\mathbf{f_1}(\mathbf{x})\cdots\mathbf{f_k}(\mathbf{x})]$ has rank k at a point $\mathbf{x}=\mathbf{x_o}$, is said to be *involutive* near x_o if for each pair of integers (i,j), $1\le i,j\le k$, the matrix $\left[\mathbf{f_1}(\mathbf{x})\cdots\mathbf{f_k}(\mathbf{x})\;[\mathbf{f_i}(\mathbf{x}),\mathbf{f_j}(\mathbf{x})]\right]$ has still rank k $\forall\mathbf{x}\in B(\mathbf{x_o})$.

Lemma 1: (Isidori [26])

For the nonlinear system given in Equation H.1, there exists an output function $h(\mathbf{x})$ for which the system has a relative degree n at a point $\mathbf{x_o}$, if and only if the following conditions are satisfied:

$$\begin{aligned} &i)\ rank\left([\mathbf{g}(\mathbf{x_o})\;ad_f\mathbf{g}(\mathbf{x_o})\cdots ad_f^{n-2}\mathbf{g}(\mathbf{x_o})\;ad_f^{n-1}\mathbf{g}(\mathbf{x_o})]\right) = n \\ &ii)\ \left(\mathbf{g}(\mathbf{x_o}), ad_f\mathbf{g}(\mathbf{x_o}),\cdots, ad_f^{n-2}\mathbf{g}(\mathbf{x_o}), ad_f^{n-1}\mathbf{g}(\mathbf{x_o})\right) \text{ is involutive near } \mathbf{x_o}. \end{aligned} \tag{H.9}$$

Comments on the proof: The proof of this Lemma makes use of Frobenius' Theorem, from the theory of partial differential equations, which states conditions for existence of solutions of the equation

$$\left(\frac{\delta}{\delta\mathbf{x}}h(\mathbf{x})\right)\left[\mathbf{f_1}\cdots\mathbf{f_k}\,[\mathbf{f_i}(\mathbf{x}),\mathbf{f_j}(\mathbf{x})]\right] = 0. \tag{H.10}$$

A system of the form H.1 that has a relative degree n, that is $r=n$ at a point $\mathbf{x_o}$, can be transformed into a linear system by the nonlinear change of coordinates

$$\mathbf{z} = \phi(\mathbf{x}) = \begin{bmatrix} h(\mathbf{x}) \\ L_f h(\mathbf{x}) \\ \vdots \\ L_f^{n-1}h(\mathbf{x}) \end{bmatrix} \tag{H.11}$$

and a state feedback

$$u = \frac{1}{a(\mathbf{z})}\left(-b(\mathbf{z}) + v\right) \tag{H.12}$$

where $a(\mathbf{z}) = L_g L_f^{r-1} h(\phi^{-1}(\mathbf{z}))$ and $b(\mathbf{z}) = L_f^r h(\phi^{-1}(\mathbf{z}))$. The resulting linear system is of the form

$$\dot{\mathbf{z}} = A\,\mathbf{z} + B\,v \qquad\qquad \mathbf{z}(0) = \mathbf{z_o} \in R^n \tag{H.13}$$

where v is the new control input and the new state is given by

$$\dot{\mathbf{z}} = \begin{bmatrix} \dot{\mathbf{z}}_1 \\ \dot{\mathbf{z}}_2 \\ \vdots \\ \dot{\mathbf{z}}_n \end{bmatrix} = \begin{bmatrix} \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_n \\ b(\mathbf{z}) + a(\mathbf{z})u \end{bmatrix} \tag{H.14}$$

and the matrices are given by

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{H.15}$$

The system H.13 is an exact linearization of H.1 around $\mathbf{x_o}$.

Remarks: Notice that $a(\mathbf{z}) \neq 0$, since $\mathbf{z_o} = \phi(\mathbf{x_o}) \neq 0$ by Definition 2, since there are no dynamics if $\mathbf{z_o} = \mathbf{0}$. Also notice that Equation H.13 is controllable since $rank\left([B\ AB\ AB^2 \cdots AB^{n-1}]\right) = n$.

Controllability

A controllable system can be transformed by an equivalence transformation into the *controllability canonical form* (Chen [12]), where the matrix A is in the Frobenius form (that is, the companion matrix of the characteristic polynomial), and B remains unchanged. It is also a fact that the controllability properties of a system remain unchanged under a state feedback $\mathbf{v} = F\,\mathbf{x} + \mathbf{v}'$, and that the eigenvalues of this closed-loop system can be arbitrarily placed. With these facts in mind, it can be considered, without loss of generality, that H.13 is in the *controllability canonical form.*

Bibliography

[1] P. Arabshahi, J. J. Choi, J. J. Mark II and T. P. Caudell, "Fuzzy Control of Backpropagation", in *Proc. IEEE 1st Int. Conf. on Fuzzy Systems*, San Diego, CA, pp. 967-972, 1992.

[2] T. Arimoto, "A New Feedback Method for Dynamic Control of Manipulators", *J. Dynamical Systems, Measurement and Control*, vol. 103, 1981.

[3] K. J. Astrom, U. Borisson, L. Ljung and B. Wittenmark, "Theory and Applications of Self-Tuning Regulators", *Automatica*, vol. 13, no. 6, pp. 235-242, 1977.

[4] K. J. Astrom and B. Wittenmark, *Computer Controlled Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[5] K. J. Astrom and T. Hagglund, *PID Controllers*. Research Triangle Park, NC: Instrument Society of America, 1988.

[6] K. J. Astrom, "Assessment of Achievable Performance of Simple Feedback Loops", in *Int. J. Adaptive Control and Signal Processing*, vol. 5, no. 1, pp. 3-20, 1991.

[7] B. R. Barmish, *New Tools for Robust Control Systems*, Course Notes. Monterrey, Mexico: University of Nuevo Leon, 1992.

[8] S. A. Billings and W. S. F. Voon, "Piecewise Linear Identification of Non-Linear Systems", in *Int. J. of Control*, vol. 46, no. 1, pp. 215-235, 1987.

[9] M. Brown, C. S. Coleman and D. A. Drew (Editors) "Differential Equation Models". Berlin: Springer-Verlag, 1983.

[10] J. J. Buckley, Y. Hayashi, and E. Czogala, "On the Equivalence of Neural Networks and Fuzzy Expert Systems", in *Proc. of the Int. Joint Conf. on Neural Networks*, vol. II, pp. 691-695, 1992.

[11] R. Castro and J. Alvarez, *Control of Nonlinear Systems*, Course Notes, *Latinoamerican Congress on Automatic Control*, Puebla, Mexico, 1990.

[12] C. T. Chen, *Linear System Theory and Design*. New York: Hold, Rinehart and Winston, 1984.

[13] G. Cybenko, "Approximations by Superpositions of a Sigmoidal Function", in *Mathematical Control, Signals and Systems*, vol. 2, pp. 303-314, 1989.

[14] R. C. Dorf, "Modern Control Systems", 7th Edition. New York: Addison-Wesley, 1995.

[15] H. Elmqvist, "SIMNON-An Interactive Simulation Program for Nonlinear Systems", in *Proc. Simulation*, Montreux, France, 1977.

[16] H. Elmqvist, K. J. Astrom, T. Schonthal and B. Wittenmark, *SIMNON, Simulation of Nonlinear Systems*. Goteborg: SSPA Systems, 1993.

[17] B. Francis and T. Chen, *Digital Control*, course notes. The Fields Institute for Research in Mathematical Sciences: Waterloo, Ontario, 1992.

[18] T. Fukuda and T. Shibata, "Hierarchical Intelligent Control for Robotic Motion by Using Fuzy, Artificial Intelligence, and Neural Networks", in *Proc. Int. J. Conf. on Neural Networks*, Baltimore, MD, vol. I, pp. 269-274, 1992.

[19] M. M. Gupta and D. H. Rao, "Dynamic Neural Units with Applications to the Control of Unknown Nonlinear Systems", *J. Intelligent & Fuzzy Systems*, vol. 1, no. 1, pp. 73-92, 1993.

[20] Y. Hayashi, J. Buckley and E. Czogala, "Fuzy Neural Network with Fuzzy Signals and Weights", *Int. J. of Intelligent Systems*, vol. 8, pp. 527-537, 1993.

[21] R. J. Hathaway and J. C. Bezdek, "Switching Regression Models and Fuzzy Clustering", in *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 3, pp. 195-204, 1993.

[22] S. Z. He, et al., "PID Self-Tuning Control Using a Fuzzy Adaptive Mechanism", in *Proc. 2nd IEEE Int. Conf. on Fuzzy Systems*, San Francisco, CA, pp. 708-713, 1993.

[23] K. Hirota and W. Pedrycz, "Referential Modes of Reasoning", in *Proc. 2nd Int. Conf. on Fuzzy Systems*, San Francisco, CA, pp. 559-563, 1993.

[24] K. Hirota and W. Pedrycz, "Directional Fuzzy Clustering", to appear in *Fuzzy Sets and Systems*, 1994.

[25] K. Hornik, "Approximation Capabilities of Multilayer Feedforward Networks", in *Neural Networks*, vol. 4, pp. 251-257, 1991.

[26] A. Isidori, *Nonlinear Control Systems*, 2nd. Ed. Berlin: Springer-Verlag, 1989.

[27] E. A. Jackson, *Perspectives of Nonlinear Dynamics*, vol. I. Cambridge: Cambridge University Press, 1991.

[28] T. A. Johansen and B. A. Foss, "Constructing NARMAX Models Using ARMAX Models", in *Int. J. on Control*, vol. 58, no. 5, pp. 1125-1153, 1993.

[29] T. Kailath, *Linear System Theory*. Englewood Cliffs, N.J.: Prentice Hall, 1980.

[30] R. Kelly, *Control de Movimiento de Robots Manipuladores*, in Spanish. CONACYT-CICESE: Ensenada, Mexico, 1994.

[31] H. K. Khalil, *Nonlinear Systems*. New York: Macmillan, 1992.

[32] W. Kickert and E. Mamdani, "Analysis of a Fuzzy Logic Controller", *Fuzzy Sets and Systems*, vol. 1, pp. 29-44, 1978.

[33] B. Kuipers and K. J. Astrom, "The Composition of Heterogeneous Control Laws", report AI90-138, Artificial Intelligence Laboratory, *Univ. of Texas at Austin*, Austin, TX, 1990.

[34] M. Le Coeur, "La Commande Floue", report. Saint-Martin-d'Heres, France: Laboratoire d'Automatique de Grenoble, 1992.

[35] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller", Parts I and II, *IEEE Trans. on Syst. Man and Cybern.*, vol. 20, no. 2, pp. 404-435, 1990.

[36] J. Lee, "On Methods for Improving Performance of PI-Type Fuzzy Logic Controllers", *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 4, pp. 298-301, 1993.

[37] C. C. Ma, "Rapid Tracking of Complex Trajectories in Short-Duration Processes", in *Automatica*, vol. 27, no. 1, pp. 173-176, 1991.

[38] J. Maiers and Y. Sherif, "Applications of Fuzzy Set Theory", in *IEEE Trans. on Syst. Man and Cybern.*, vol. SMC-15, no. 1, pp. 175-189, 1985.

[39] H.A. Malki, H. Li and G. Chen, "New Design and Stability Analysis of Fuzzy Proportional-Derivative Control Systems", in *IEEE Trans. on Fuzzy Systems*, vol. 2, no. 4, pp. 245-254, 1994.

[40] E. Mamdani, "Applications of Fuzzy Algorithms for Control of Simple Dynamic Plants", in *Proc. IEEE*, vol. 121, pp. 1585-1588, 1976.

[41] E. Mamdani, P. King, "The Application of Fuzzy Control Systems to Industrial Processes", *Automatica*, vol. 13, pp. 235-242, 1977.

[42] The MATH WORKS Inc., *MATLAB*. New Jersey: Prentice Hall, 1992.

[43] J. M. Meslin, J. Zhou and P. Coiffet, "Fuzzy Dynamic Control of Manipulators: A Scheduling Approach", in *Proc. IEEE Int. Conf. on Syst. Man. and Cybern.*, Le Touquet, France, pp. 69-73, 1993.

[44] H. Nijmeijer and A. van der Schaft, *Nonlinear Dynamical Control Systems*. Berlin: Springer-Verlag, 1990.

[45] A. W. Naylor and G. R. Sell, *Linear Operator Theory in Engineering and Science*. Berlin: Springer-Verlag, 1982

[46] B. O'Neill, *Elementary Differential Geometry*. New York: Academic Press, 1969.

[47] R. Palm, "Tuning of Scaling Factors in Fuzzy Controllers Using Correlation Functions", in *Proc. 2nd IEEE Int. Conf. on Fuzzy Systems*, San Francisco, CA, pp. 691-696, 1993.

[48] W. Pedrycz, "Fuzzy Neural Networks with Reference Neurons as Pattern Classifiers", in *IEEE Trans. on Neural Networks*, vol. 3, no. 5, pp. 770-775, 1992.

[49] W. Pedrycz, *Fuzzy Control and Fuzzy Systems*, 2nd edition. Taunton: Research Studies Press, John Wiley, 1993.

[50] W. Pedrycz, "Fuzzy Neural Networks and Neurocomputations", *Fuzzy Sets and Systems*, vol. 56, pp. 1-28, 1993.

[51] W. Pedrycz and A. F. Rocha, "Fuzzy-Set Based Models of Neurons and Knowledge-Based Networks", *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 4, pp. 254-266, 1993.

[52] S. J. Qin and G. Borders, "A Multiregion Fuzzy Logic Controller for Nonlinear Process Control", in *IEEE Trans. on Fuzzy Systems*, vol. 2, no. 1, pp. 74-81, 1994.

[53] G. Raju and J. Zhou, "Fuzzy Rule Based Approach for Robot Motion Control", in *Proc. IEEE 1st Int. Conf. on Fuzzy Systems*, San Diego, CA, pp. 1349-1356, 1992.

[54] H. Rotstein and A. Sideris, "H_∞ Optimization with Time-Domain Constraints", in *IEEE Trans. on Aautomatic Control*, vol. 39, no. 4, pp. 762-779, 1994.

[55] G. A. Rovithakis, M. A. Christodoulou, "Adaptive Control of Unknown Plants Using Dynamical Neural Networks", in *IEEE Trans. on Syst. Man and Cybern.*, vol. 24, no. 3, pp. 400-412, 1994.

[56] A. Rueda, *Dynamics and Two New PD-Type Controllers for Robot Manipulators*, M. Sc. Thesis. Univ. of Manitoba: Dept. of Electrical and Computer Eng., Winnipeg, Canada, 1991.

[57] A. Rueda and W. Pedrycz, "Fuzzy Coordinator in Control Problems", in *NAFIPS Int. Conf. on Fuzzy Systems*, Puerto Vallarta, Mexico, vol. I, pp. 322-329, 1992.

[58] A. Rueda and W. Pedrycz, "A Design Method for a Class of Fuzzy Coordinator Controllers", in *Proc. 2nd IEEE Int. Conf. on Fuzzy Systems*, San Franciso, CA, pp. 196-199, 1993.

[59] A. Rueda, *Fuzzy Hierarchical Commuting Control*, Ph. D. Candidacy Examination Paper. Univ. of Manitoba: Dept. of Electrical and Computer Eng., Winnipeg, Canada, 1993.

[60] A. Rueda, W. Pedrycz, R. Kelly and R. Carelli, "Control of a Robot Manipulator via a Stable Fuzzy-Neuro Controller", in *Proc. Int. Conf. on Qualitative Information, Fuzzy Techniques and Neural Networks in Simulation*, Barcelona, Spain, 1994.

[61] A. Rueda and W. Pedrycz, "A Hierarchical Fuzzy-Neural-PD Controller for Robot Manipulators", in *Proc. 3rd IEEE Int. Conf. on Fuzzy Systems*, Orlando, FL, 1994.

[62] A. Rueda and W. Pedrycz, "On Fuzzy Switching Control and an Application to Robot Manipulators", in *Proc. VII Int. Symposium on Artificial Intelligence*, Monterrey, Mexico, 1994.

[63] E. N. Sanchez and V. Vega, "Stability of Neuro-Fuzzy Control", in *Proc. American Control Conference*, Seattle, WA, pp. 4251-4252, 1995.

[64] T. Shibata, T. Fukuda, K. Kosuge, F. Arai, M. Tokita and T. Mitsuoka, "Skill Based Control by Using Fuzzy Neural Network for Hierarchical intelligent Control", in *Proc. of the Int. J. Conf. on Neural Networks*, Baltimore, MD, vol. II, pp. 81-86, 1992.

[65] C. W. de Silva, "Simulation Studies of an Analytical Fuzzy Tuner for a PID Servo" , report, *Dept. of Mechanical Eng., Univ. of British Columbia*, Canada, 1992.

[66] P. K. Simpson, *Artificial Neural Systems*. New York: Pergamon Press, 1990.

[67] P. K. Simpson, "Fuzzy Min-Max Neural Networks–Part 1: Classification", in *IEEE Trans. on Neural Networks*, vol. 3, no. 5, pp. 776-785, 1992.

[68] P. K. Simpson, "Fuzzy Min-Max Neural Networks–Part 2: Clustering", in *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 32-45, 1993.

[69] M. Spong and M. Vidyasagar, *Robot Dynamics and Control*. New York: John Wiley & Sons, 1989.

[70] M. Sugeno, editor, *Industrial Applications of Fuzzy Control*. Amsterdam: North Holland, 1985.

[71] M. Sugeno and T. Yasukawa, "A Fuzzy-Logic-Based Approach to Qualitative Modeling", in *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, 1993.

[72] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modelling and Control", *IEEE Trans. on Syst. Man and Cybern.*, vol. 15, no. 1, pp. 116-132, 1985.

[73] M. Takegaki and S. Arimoto, "An Adaptive Trajectory Control of Manipulators", in *Int. J. of Control*, vol. 34, no. 2, pp. 219-230, 1981.

[74] S. Tan and S. He, "Hybrid Control of Nonlinear Dynamical Systems Using Neural Nets and Conventional Control Schemes", in *Proc. Int. J. Conf. on Neural Networks*, Baltimore, MD, vol. II, pp. 805-810, 1992.

[75] R. M. Tong, "A Control Engineering Review of Fuzzy Systems", *Automatica*, vol. 13, pp. 559-569, 1977.

[76] H. C. Tseng, V. H. Hwang and S. L. Lui, "Fuzzy Servo Controller: The Hierarchical Approach", in *Proc.IEEE Int. Conf. on Fuzzy Systems*, San Diego, CA, pp. 623-631, 1992.

[77] H. C. Tseng and V. H. Hwang, "Sevocontroller Tuning with Fuzzy Logic", in *IEEE Trans. on Control Systems Technology*, vol. 1, no. 4, pp. 262-269, 1993.

[78] M. Vidyasagar, *Nonlinear Systems Analysis*, 2nd edition. New York: Prentice-Hall, 1993.

[79] L. X. Wang and J. M. Mendel, "Back-Propagation Fuzzy System as Nonlinear Dynamic System Identifiers", in *Proc.IEEE Int. Conf. on Fuzzy Systems*, San Diego, CA, pp. 1409-1418, 1992.

[80] L. X. Wang, "Stable Adaptive Fuzzy Control of Nonlinear Systems", in *IEEE Trans. on Fuzzy Systems*, vol. 1, no. 2, pp. 146-155, 1993.

[81] J. Wen and S. Murphy, "PID Control for Robot Manipulators", CIRSSE Document 54, *Rensselaer Polytechnic Institute*, 1990.

[82] R. R. Yager and D. P. Filev, "Identification of Nonlinear Systems by Fuzzy Models", in *Proc. 2nd IEEE Int. Conf. on Fuzzy Systems*, San Diego, CA, pp. 1401-1408

[83] K. Young and C. Fan, "Control of Voluntary Limb Movements by Using a Fuzzy System", in *Proc. 32nd Conf. on Decision and Control*, pp. 1759-1764, 1993.

[84] L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes", *IEEE Trans. on Syst. Man and Cybern.*, vol. 3, no. 1, pp. 28-44, 1973.

[85] Z. Y. Zhao, M. Tomizuka and S. Isaka, "Fuzzy Gain Scheduling of PID Controllers", in *IEEE Trans. on Syst. Man and Cybern.*, vol. 23, no. 5, pp. 1392-1398, 1993.