DEVELOPMENT OF A KNOWLEDGE-BASED REGULATOR FOR A PWR-TYPE NUCLEAR POWER PLANT

by

H. Levent Akın

B.S. in A.E., Istanbul Technical University, 1982

M.S. in N.E, Boğaziçi University, 1984

Submitted to the Institute for Graduate Studies in Science and Engineering in partial fulfillment of the requirements for the degree of

Doctor

of

Philosophy

Fen

Boğaziçi University

1989

DEVELOPMENT OF A KNOWLEDGE-BASED REGULATOR

FOR A PWR-TYPE NUCLEAR POWER PLANT

APPROVED BY

Prof.Dr. Turan B. Enginol (Thesis Supervisor)	Turan B. Enginol
Doç. Dr. Vural Altın	V. Altın
Prof.Dr. Özer Çiftçioglu	Özer Çiftçioğlu
Prof.Dr. Melih Geçkinli	Melih Geçkinli
Doç. Dr. Selahattin Kuru	[illegible]

DATE OF APPROVAL July 3, 1989

To my parents

ACKNOWLEDGEMENTS

I would like to express my special thanks to Doç.Dr. Vural Altın for his invaluable and encouraging guidance throughout this work. Special thanks are extended to Prof.Dr.Turan B. Enginol, Prof.Dr. Özer Çiftçioğlu, Doç.Dr.Selahattin Kuru and Prof.Dr.Melih Geçkinli for their constructive criticisms and suggestions.

I would also like to thank my friends Dilek Kaptanoglu and Işıl Akmehmet for helping me in obtaining some of the reference material and to Onur Uzonur for helping in the preparation of the manuscript.

Levent Akın

ABSTRACT

In this study, a rule-based fuzzy logic controller for a PWR nuclear power plant has been developed in order to regulate the power around a full power setpoint.

In this artificial intelligence application, knowledge acquisition was performed through numerical simulation using a validated linear model of the H.B. Robinson power plant and production rules were used for knowledge representation. For comparison purposes broken-line and S-shaped fuzzy sets were investigated and broken-line fuzzy sets were preferred. The regulator was implemented on an IBM-compatible PC using the PASCAL language.

The performance of the rule-based controller was compared to that of an optimal controller and was found to be better in the sense that the overshoots were less. Also, the effect of noise in sensor data and variation in reactor parameters were investigated and their effect on the performance of the controller was found to be significant implying that the designed regulator is sufficiently robust.

ÖZET

Bu çalışmada PWR tipi bir nükleer güç santralı için tam güç etrafında regülasyon görevi yapacak ,kural tabanlı, bulanık mantık kullanan bir denetleyici geliştirilmiştir.

Bu yapay zeka uygulamasında bilgi, H. B. Robınson güç santralının doğrulanmış lineer bir modelinin sayısal simülasyonu yapılarak derlenmiş ve bilgi tasviri için üretim kuralları kullanılmıştır. Karşılaştırma amacıyla kırık çizgi ve S-biçimli bulanık kümeler incelenmiş ve kırık-çizgi tipinde olanlar tercih edilmiştir. Regülatör, bir IBM-uyumlu PC de PASCAL dili kullanılarak yazılmıştır.

Kural-tabanlı denetleyicinin performansı bir optimal denetleyicininkiyle karşılaştırılmış ve sapmaların azlığı açısından daha iyi olduğu tespit edilmiştir. Ayrıca algılayıcılardaki gürültü ve reaktör parametrelerindeki değişimlerin performans üzerindeki etkisi de araştırılmış ve çok az olduğu görülmüş, dolayısıyla denetleyicinin yeterince robust olduğu sonucuna varılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

A^2	heat transfer area, m
A_c	flow area of the channel, m^2
A_f	heat transfer area, m^2
δC_i	deviation of normalized precursor concentration from its steady state value
C_m	specific heat of the tube metal,
C_p	specific heat capacity of the fluid,
C_{pFW}	specific heat of the feedwater,
E	internal energy of secondary coolant, J
E_w	internal energy of water in the pressurizer, J
F	weighting factor
F_{ci}	reactivity importance for temperature changes in the i th coolant node.
F_{pci}	the fraction of total power released in coolant node i
F_{fi}	reactivity importance for temperature changes in the i th fuel node.
K	proportionality factor
M	mass of coolant in node i, kg
M_m	mass of tube metal, kg
M_p	mass of primary coolant in the steam generator,kg
M_s	mass of steam in the pressurizer, mass of steam in the steam generator, kg

M_w	mass of water in the pressurizer, mass of water in the steam generator, kg
P	the total power, MW
δP	deviation of reactor power from initial steady-state value, MW
P_h	heated perimeter of the channel, m
P_o	initial steady-state power level, MW
P_p	primary system pressure, MPa
P_s	steam pressure, MPa
Q	volumetric heat generation rate, W/m^3
Q_c	volumetric heat generation rate in the fluid, W/m
Q_{fi}	fraction of total reactor power generated in fuel node i
R	fuel-to-coolant heat transfer resistance, gas constant
T	fuel temperature, fluid temperature, C
T_{av}	average primary temperature, C
δT_{ci}	deviation of coolant temperature in the i th coolant node from its initial steady state value, C
δT_{cin}	deviation in inlet temperature of the first coolant node from its initial steady state value, C
δT_{cL}	deviation of cold-leg temperature from its steady state value, C
δT_{fi}	deviation of fuel temperature in the i th fuel node from its initial steady state value, C

δT_{HL}	deviation of hot-leg temperature from its steady state value, C
δT_{IP}	deviation of primary coolant temperature in the steam generator inlet plenum from its initial steady state value,C
T_{in}	fluid temperature at entrance,C
δT_{LP}	deviation of reactor lower plenum temperature from its initial steady state value, C
T_m	steam generator tube metal temperature, C
δT_{OP}	deviation of temperature of primary coolant in the steam generator outlet plenum, C
δT_p	primary coolant temperature in the steam generator, C
T_{ref}	reference temperature for load changes, C
T_s	fuel rod surface temperature, the average steam temperature, saturation temperature, C
δT_{UP}	deviation of reactor upper plenum temperature from its initial steady state value, C
U	fluid velocity, m/s, overall fuel to-coolant heat transfer coefficient
V_i	volume of i th coolant node, m^3
V_s	volume of steam in the pressurizer or steam generator, m^3
V_w	volume of water in the pressurizer or steam generator, m^3
W_{FW}	feedwater flow rate, kg/s
W_{wi}	mass flow of water into (or out of) the pressurizer, kg/s

W_s	flashing rate (or condensing rate) in the pressurizer, steam generation rate, kg/s
$W_{s,o}$	steam flow rate to the turbine, kg/s
X	integral control action variable
c	specific heat capacity,
h	film heat transfer coefficient,
h_{ms}	heat transfer coefficient for metal to secondary coolant
h_{pm}	heat transfer coefficient for primary coolant to metal
h_s	enthalpy of steam in the pressurizer,
h_{wi}	enthalpy of water entering the pressurizer,
k	thermal conductivity.
q	rate of heat addition to the pressurizer fluid with electric heater, W/s
r	fuel radius, m
x	distance along channel, m
	neutron generation time
α_c	coolant temperature coefficient of reactivity, 1/C
α_f	fuel temperature coefficient of reactivity, 1/C
α_p	coolant pressure coefficient of reactivity, 1/MPa
β	total delayed neutron fraction
β_i	delayed neutron fraction for the i th delayed neutron group

τ_1	slope of coolant density versus temperature curve
λ_i	delayed neutron decay constant for i th delayed neutron group
ρ	fluid density, density, kg/m
$\delta\rho_{rod}$	reactivity due to control rod movement
τ	residence time of fluid in node i, s
τ_{sg}	residence time of coolant in the steam generator, s

I. INTRODUCTION

As a means of control, the well developed analytic technologies, which require the accurate modeling of the process under control, have been used successfully for years. However, they are not applicable to the ill-defined processes not amenable to modeling, thus instead experienced persons are employed as operators who often perform satisfactorily despite the imprecision of the available information. The imprecision is generally due to the nonlinearity of the process, or to the time delays between the application of the control signal, or degraded sensors. The operator copes with lack of structure by employing heuristics, which are the rules of thumb that people use to solve problems when a lack of time or understanding prevents an analysis of the parameters involved(1). However, automatization of operation has its benefits as is apparent from the operating records of plants with process controllers. The crucial factors here are then, capturing the essence of expert behavior and implementing it in an automaton. Rule-based fuzzy logic control technique originally introduced by Mamdani and Assilian were applied to ill-defined processes successfully demonstrating that this approach is both possible and practical(2). However, for the operation of well-characterized systems the benefits of this approach has not been demonstrated clearly due to the scarcity of applications. Actually, the rationale for this application exists: in addition to some drawbacks such as the requirement of accurate modelling of the process under control, these systems are sensitive to failures in sensors, always inflexible, meaning a large drift in process variables renders the controller almost useless, and in more dramatic cases dangerous. The advantage of rule-based controllers is that they are generally more robust than their analytic counterparts but, there are no comprehensive guidelines for the design of rule-based controllers and such systems are quite difficult to calibrate. Therefore, the rule-based and analytic technologies should be used complementarily, with rule-based systems being employed both as backups to analytic controllers and as a means of improving the man machine interface by providing human operators with the rationale for automatic control actions(3).

The nuclear industry , especially that of U.S., had been very reluctant in using automatic control extensively. This is mainly due to the nuclear power plants(NPPs) being base load type not requiring load following, and also because some safety regulations require the designer to take into consideration the controller initiated abnormal conditions(4). Recently, however, as the percentage of electricity generated by the NPPs increased, the necessity of operating them in load following mode became apparent. Existing control systems were either not capable of performing the required tasks or were cost ineffective and clumsy. Introduction of multivariable control methods with the aim of improving the stability of interacting systems, thus permitting higher gains and better control, were initiated, mainly for the CANDU type NPPs(5) with a better chance of implementation due to previous experience with digital control. For PWR type plants, the main line of research has been directed towards load following control using approximate noninteractive control(6), optimal control(7,8), adaptive control(9). There has also been optimal control applications for disturbance control(10,11,12,13), with further recent research on such interesting methods as nonlinear multivariable control based on the unknown-but-bounded disturbance model(14). The "reactivity constrained approach"(15) which was successfully applied to research reactors is also worth mentioning as a promising automatic control method. The rule-based controllers have been designed and implemented for research reactors(3), and have been designed for load following operation for a PWR(16), and for BWR recirculation flow control system(17).

In this study, a rule-based fuzzy logic regulator for a PWR type NPP is developed in order to show the benefits of this application during normal operation, in the case of noise in sensor data, and drifts in process variables such as the moderator feedback coefficient.

In chapter 2 the mathematical model of the H. B. Robinson nuclear power plant is described. Rule-based fuzzy logic process control is reviewed in chapter 3 and in chapter 4, the development of the regulator constructed during this work is described. Finally, the results of this study and the conclusions derived are presented in chapter 5.

II. MODELING AND SIMULATION OF THE NUCLEAR POWER PLANT

In this work, a simple, fast and validated mathematical model of a PWR type nuclear power plant was chosen. Since the knowledge acquisition, development and testing stages of the knowledge-based controller necessitates extensive simulations[3] due to the fact that its structure is not suited to mathematical techniques used for conceiving the analytical controllers[18], a simple and fast model is necessary. On the other hand, the difficulty in obtaining relevant and most of the time proprietary plant data and more over testing its performance dictates the choice of a model already proven to be valid which was derived from the first principles and tested by using actual operating data. Thus, the multi-input mathematical model formulated and validated against full power experimental results by Kerlin et al.[19] for predicting the dynamic response of the H.B.Robinson power plant (HBR), during full power operation is used.

2.1 Model of the H.B. Robinson PWR Nuclear Power Plant

The model is based on the basic conservation laws for neutrons, mass and energy. It includes the representation for point kinetics, core heat transfer, pressurizer, piping and the steam generator components shown in Fig.2.1.

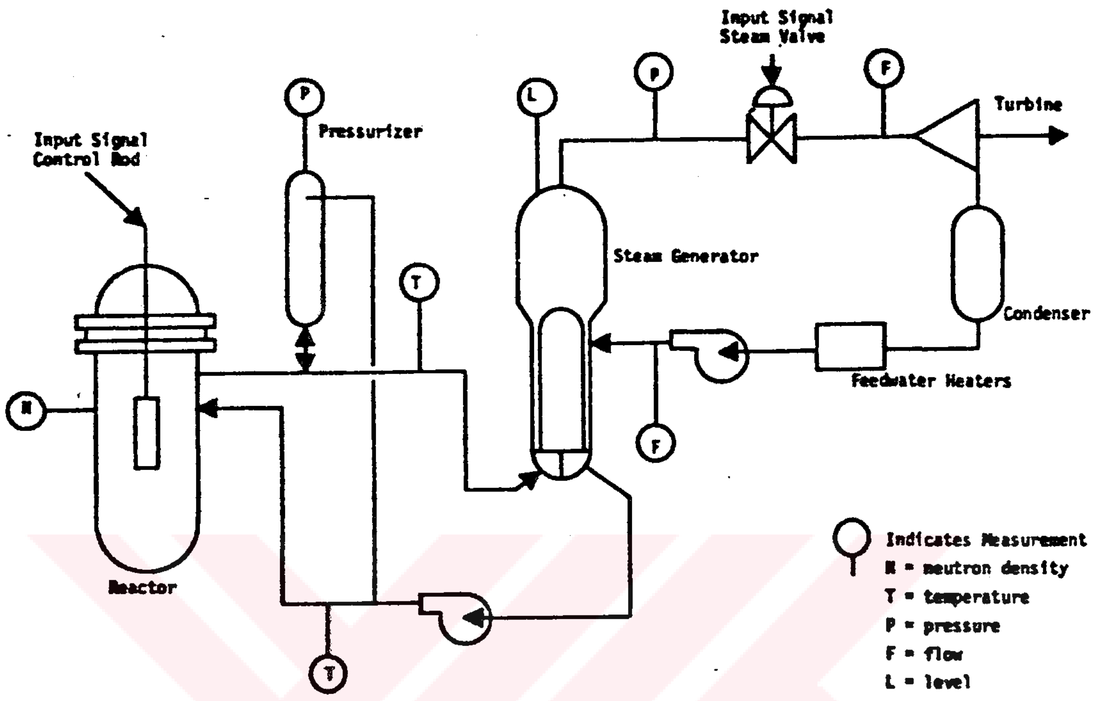

FIGURE 2.1. Schematic of the H.B. Robinson nuclear plant.

2.1.1 Reactor Core

<u>2.1.1.1. Neutronics.</u> The neutron population in a nuclear reactor is a function of time, position, energy, and direction of motion, and its most complete description is given by the Boltzmann transport equation(20). This model is extremely clumsy to implement for simulation purposes and is not used for models developed to simulate operational transients(21).

From the Boltzmann equation the point kinetics equations can be derived which can be used when the core is tightly coupled and spatial dependencies are not important(20).

In the H.B.Robinson model, the reactor power was modeled using the point kinetics equations with six groups of delayed neutrons and reactivity feedbacks due to changes in fuel temperature, coolant temperature, and primary coolant system pressure.

Since our study is concerned with a regulator design, the time scales are of the order of seconds, hence a description of xenon-135 build-up and decay is not necessary.

The feedback due to changes in moderator and fuel temperature and pressure of the primary coolant is handled using reactivity feedback coefficients. These coefficients give the proportionality that exists between temperature or pressure and reactivity. As these relationships are generally nonlinear, it is common to use the linear approximation around an operating point which was in our case the full power.

The linearized point kinetics equations, valid for small variations in reactivity and power, are:

$$\frac{d\delta P}{dt} = -\frac{\beta}{\Lambda}\delta P + \sum_i \lambda_i \delta C_i + \frac{\alpha_f P_o}{\Lambda} \sum_{\substack{\text{fuel}\\ \text{nodes}}} F_{fi}\ \delta T_{fi} + \frac{\alpha_p P_o}{\Lambda} \delta P_p$$
$$+ \frac{P_o}{\Lambda}\delta\rho_{rod} + \frac{\alpha_c P_o}{\Lambda} \sum_{\substack{\text{coolant}\\ \text{nodes}}} F_{ci}\ \delta T_{ci} \qquad (2.1)$$

$$\frac{d\delta C_i}{dt} = \frac{\beta_i}{\Lambda}\delta P - \lambda_i \delta C_i \qquad (2.2)$$

where δP is the deviation of reactor power from initial steady-state value, P_0 initial steady state power level, β_i is the delayed neutron constant for the i'the delayed neutron group, δC_i deviation of normalized precursor concentration from its steady state value, P_p primary pressure, α_f fuel temperature coefficient of reactivity, α_c coolant temperature coefficient of reactivity, α_p coolant pressure coefficient of reactivity, δT_{fi} deviation of fuel temperature in the i'th fuel node from its initial steady state value, δT_{ci} deviation of coolant temperature in the i'th coolant node from its initial steady state value, $\delta \rho_{rod}$ reactivity due to control rod movement, λ_i delayed neutron fraction for the i'th delayed neutron group, β total delayed neutron fraction, Λ neutron generation time, F_{fi} reactivity importance for temperature changes in the i'th fuel node, F_{ci} reactivity importance for temperature changes in the i'th coolant node.

Data from Tables 2.1 and 2.2 were used to evaluate the coefficients. For a model with one fuel node and two coolant nodes, the resulting equations taking F_{fi} = 1 and F_{ci} = 0.5 are:

$$\frac{d\delta P}{dt} = -400\ \delta P + 0.0125\ \delta C_1 + 0.0305\ \delta C_2 + 0.111\ \delta C_3 + 0.301\ \delta C_4 + 1.140\ \delta C_5 + 3.01\ \delta C_6 - 1781\ \delta T_f - 13700\ \delta T_{c1} - 13700\ \delta T_{c2} + 411\ \delta P_p \quad (2.3)$$

$$\frac{d\delta C_1}{dt} = 13.125\ \delta P - 0.0125\ \delta C_1 \quad (2.4)$$

$$\frac{d\delta C_2}{dt} = 87.5\ \delta P - 0.0305\ \delta C_2 \quad (2.5)$$

$$\frac{d\delta C_3}{dt} = 78.125\ \delta P - 0.111\ \delta C_3 \qquad (2.6)$$

$$\frac{d\delta C_4}{dt} = 158.125\ \delta P - 0.301\ \delta C_4 \qquad (2.7)$$

$$\frac{d\delta C_5}{dt} = 46.25\ \delta P - 1.140\ \delta C_5 \qquad (2.8)$$

$$\frac{d\delta C_6}{dt} = 16.875\ \delta P - 3.01\ \delta C_6 \qquad (2.9)$$

TABLE 2.1. Reactor Design Data.

Core Thermal and Hydraulic Characteristics	
Total primary heat output, MW (th)	2200
Nominal primary system pressure, psi (MPa)	2250 (15.82)
Total coolant flow rate, lb/h (kg/s)	101.5×10^6 (1.279×10^4)
Average coolant velocity along fuel rods, ft/sec (m/s)	14.3 (4.358)
Total mass of coolant in primary loop, lb (kg)	406 050 (184 347)
Nominal coolant inlet temperature, °F (C)	546.2 (285.66)
Nominal coolant outlet temperature, °F (C)	602.1 (316.72)
Active heat transfer surface area, ft^2 (m^2)	42460 (3944.7)
Average heat flux, Btu/(h ft^2) (W/m^2)	171600 (5.44×10^4)
Fuel-to-coolant heat transfer coefficient (includes resistance in fuel) Btu/(h ft^2-F) (W/m2-C)	176 (100.4)
Kinetic Characteristics	
Doppler coefficient, (Δk/k)/°F ((Δk/k)/°C)	-1.3×10^{-5} (-2.34×10^{-5})
Moderator temperature coefficient, (Δk/k)/°F ((Δk/k)/°C)	-2.0×10^{-4} (-3.6×10^{-4})
Moderator pressure coefficient, (Δk/k)/psi ((Δk/k)/MPa)	$+3.0 \times 10^{-4}$ (4.27×10^{-4})
Prompt neutron lifetime, sec	1.6×10^{-5}
Delayed neutron fraction	0.0064

TABLE 2.2. Delayed Neutron Constants.

Mean Life (sec)	Decay Constant ($_i$, sec^{-1})	Fraction
80.4	0.0124	0.00021
32.8	0.0305	0.00140
8.98	0.111	0.00125
3.32	0.301	0.00253
0.88	1.14	0.00074
0.332	3.01	0.00027

2.1.1.2. Core Heat Transfer. The core heat transfer model includes conduction in the fuel and heat transfer in the coolant.

Dynamic analysis of a power reactor must include calculation of fuel element temperature in the cylindrical rods. For this purpose the heat conduction equation must be used[22]

$$c \frac{\partial T}{\partial t} = Q + \nabla . k\nabla T, \tag{2.10}$$

where T is the temperature, Q heat generation rate, and k is the thermal conductivity.

~~Generally, radial conduction~~ dominates over axial or azimuthal conduction in a fuel rod, so that for constant k, Eq.(2.10) can be written as

$$c\frac{\partial T}{\partial t} = Q + k \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r}\right) \tag{2.11}$$

For modeling purposes, nodal approach is the most common method. In this approach, a single node can be used to represent the average condition in the fuel, gap, clad assembly. If a better representation is desired, the fuel can be divided into several sections. The clad is often represented by a separate node, but the gap is usually treated as a simple resistance (no heat capacity). However, gap conductance is very hard to determine, and depends closely on the operating conditions. In this work, the fuel is represented by a single node.

For the core heat transfer model, a heat balance equation for the coolant is also necessary. Assuming constant coolant density, one need not write a mass balance equation. Since, in normal PWR operation the flow is constant, a momentum balance is also not required. The heat balance for a single-phase, incompressible fluid flowing in one-dimensional slug flow is

$$\frac{\partial T}{\partial t} + U\frac{\partial T}{\partial x} = \frac{h\ P_h}{A_c \rho C_p}(T_s - T) + \frac{Q_c}{\rho C_p} \qquad (2.12)$$

where T is fluid temperature, U fluid velocity, x, distance along channel, T_s fuel rod surface temperature, Q_c volumetric heat generation rate in the fluid, ρ fluid density, C_p specific heat capacity of the fluid, h film heat transfer coefficient, P_h heated perimeter of the channel, and A_c flow area of the channel.

The nodal model for the coolant is

$$\frac{dT_{avi}}{dt} = \frac{F_{pci}}{(MC_p)_i} P_t + \frac{1}{R\ (MC_p)_i} (T_s - T_i)$$

$$+ \frac{1}{\tau} (T_{i-1} - T_i) \qquad (2.13)$$

where, F_{pci} is the fraction of total power released in coolant node i (assumed constant), M mass of coolant in node i, C specific heat of coolant, R fuel-to-coolant heat transfer resistance, T_{avi} average coolant temperature in node i, T_i outlet coolant temperature in node i, and τ residence time of fluid in node i. It is necessary at this stage to provide an equation relating T_{avi} and T_i for the system to be completely defined. Although this relation between the node average temperature and the node outlet temperature will vary during a transient, the relation between these variables is usually assumed to be constant and of the form

$$T_{avi} = FT_{i-1} + (1 - F)\ T_i \qquad (2.14)$$

where F is a weighing factor.

The common assumptions used for F are as follows:

a) Arithmetic average, (F = ½)

$$T_{avi} = \tfrac{1}{2}\ T_{i-1} + \tfrac{1}{2}\ T_i \qquad (2.15)$$

b) Well-stirred approximation (F = 0)

$$T_{avi} = T_i \qquad (2.16)$$

Also , sometimes a choice for F based on steady state temperature distribution is made.

The algebraic relation thus obtained can be substituted into Eq.(2.13) to eliminate T_i or T_{i-1} giving

$$\frac{dT_{avi}}{dt} = \frac{F_{pci}}{(MC_p)_i} P_t + \frac{1}{R\ (MC_p)_i} T_s - \left[\frac{1}{R\ (MC_p)_i} + \frac{1}{(1-F)} \right] T_{avi} + \frac{1}{(1-F)} T_{i-1} \qquad (2.17)$$

The model thus obtained has no explicit terms for nodal outlet temperature, therefore T_{i-1} can only be written as

$$T_{i-1} = \frac{1}{1-F} T_{i-1} - \frac{F}{1-F} T_{i-2} \qquad (2.18)$$

It can easily be seen that unless F = 0 the inlet temperature in each node of the series of fluid nodes is immediately affected by all upstream nodes. This is an unrealistic result born out of the assumptions made in deriving the model. Also, for F = 0, another unrealistic result , i.e., an initial decrease in outlet temperatures when inlet temperatures undergo a step increase, is implied by the model. Because of these deficiencies the well-mixed assumption was used widely. However, this choice has the drawback of implying the equality of average and outlet conditions in a finite-size region.

In order to overcome this flaw, two coolant nodes are used for each fuel node to obtain a good approximation to the average coolant temperature in HBR model. In the model with two coolant nodes for each fuel node, coolant node considerations are based on a well-mixed approximation. The average temperature of the first section is taken as the fluid temperature to determine the heat transfer driving force

$$q = \frac{1}{R} (T_s - T_{avi}) \qquad (2.19)$$

and half of this heat is transferred to each fluid section. The outlet temperature is taken as the average of the second section (see Fig.2.2). Although the model accuracy is increased the number of equations are also increased as a result of using more fluid sections. The resulting equations are:

$$\frac{d\delta T_{fi}}{dt} = \frac{Q_{fi}}{(MC_p)_{fi}} \delta P - \left(\frac{UA_f}{MC_p}\right)_{fi} (\delta T_{fi} - \delta T_{c1i}) \qquad (2.20)$$

$$\frac{d\delta T_{c1i}}{dt} = \left(\frac{UA_f}{MC_p}\right)_{ci} (\delta T_{fi} - \delta T_{c1i}) - \frac{2}{\tau} (\delta T_{c1i} - \delta T_{cin}) \qquad (2.21)$$

$$\frac{d\delta T_{c2i}}{dt} = \left(\frac{UA_f}{MC_p}\right)_{ci} (\delta T_{fi} - \delta T_{c1i}) - \frac{2}{\tau} (\delta T_{c2i} - \delta T_{c1i}) \qquad (2.22)$$

where δT_{fi} is the average fuel temperature, δT_{c1i} average coolant temperature in the i'th fuel node, δT_{c2i} outlet coolant temperature in the i'th fuel node, Q_{fi} fraction of total reactor power generated in fuel node i, $(MC_p)_{fi}$ total heat capacity for i'th fuel node, $(MC_p)_{fi}$ total heat capacity of both coolant nodes associated with i'th fuel node, U overall fuel-to-coolant heat transfer coefficient(includes resistance in fuel as well as film resistance), A_f heat transfer area, residence time(both coolant nodes), δT_{cin} deviation in inlet temperature of the first coolant node from its initial steady state value.

As was shown by Kerlin et al[19], a simplified core heat transfer model with 3 heat transfer nodes (1 for fuel and 2 for coolant) has a behavior similar to that of a more detailed one, therefore it was used for the core in the complete system model.

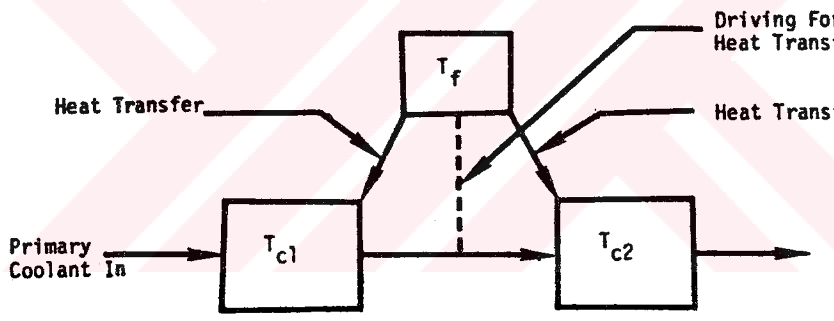

FIGURE 2.2. Schematic of fuel-to-coolant heat transfer model

Evaluation of the coefficients yields

$$\frac{d\delta T_f}{dt} = 0.0756\ \delta P - 0.16466\ \delta T_f + 0.16466\ T_{c1} \qquad (2.23)$$

$$\frac{d\delta T_{c1}}{dt} = 0.05707\ T_f - 2.4403\ T_{c1} + 2.3832\ \delta T_{LP} \qquad (2.24)$$

$$\frac{d\delta T_{c2}}{dt} = 0.05707\ T_f + 2.3262\ T_{c1} - 2.3832\ \delta T_{LP} \qquad (2.25)$$

2.1.2. Pressurizer

Although the pressurizer is a rather simple device consisting of a heated tank containing steam and water, formulation of a dynamic model can be quite complicated if detailed performance analysis is required. Especially for the analysis of small break LOCA's and similar accidents, reliable physical models for all of the components in the loops are necessary so that computer experiments can be run and the best strategies be adopted for handling accidents(23,24). Such models require multiple region models where the pressurizer model is divided into regions according to phase condition and energy, and nonequilibrium conditions are assumed to prevail. However, for normal operation, and especially for our case where only small deviations from an operating point are considered, a pressurizer model based on mass, energy, and volume balances with the assumption that saturation conditions always apply for the steam water mixture in the pressurizer, can describe the physical procesess adequately. And some authors neglect the pressurizer dynamics completely by assuming that the size of the pressurizer is large enough to accommodate the steam generator primary volume surges. Nevertheless, in this study, a model of the pressurizer, however crude, was considered necessary, and therefore incorporated.

The basic equations for the pressurizer model are:

1. Water mass balance

$$\frac{dM_w}{dt} = W_{wi} - W_s \tag{2.26}$$

2. Steam mass balance

$$\frac{dM_s}{dt} = W_s \tag{2.27}$$

3. Water energy balance

$$\frac{dE_w}{dt} = W_{wi} h_{wi} - W_s h_s - P\dot{V}_w + q \tag{2.28}$$

4. Volume balance

$$V_w + V_s = V_t \tag{2.29}$$

5. Compressibility-corrected perfect gas law

$$P_p V_s = M_s R T_s \tag{2.30}$$

where M_w is the mass of water in the pressurizer, M_s mass of steam in the pressurizer, W_{wi} mass flow of water into (or out of) pressurizer, W_s flashing rate(or condensing rate) in the pressurizer, E_w internal energy of water in the pressurizer, h_{wi} enthalpy of water entering the pressurizer, h_s enthalpy of steam in the pressurizer, P_p pressure in the pressurizer, q rate of heat addition to the pressurizer with electric heater, R gas constant, T_s saturation temperature, V_w volume of water in the pressurizer.

The equations are linearized and manipulated to obtain

$$\frac{d\delta P_p}{dt} = B_1 \ \delta P_p + B_2 \ \delta W_w + B_3 \ \delta q \qquad (2.31)$$

The values of B_1, B_2, and B_3 for the H.B. Robinson Nuclear plant calculated by using the data given in Table 2.3 are:

$$B_1 = -1.913 \times 10^{-6} \ (\mathrm{sec}^{-1})$$

$$B_2 = 7.021 \times 10^{-3} \ (\mathrm{psi/lb})$$

$$B_3 = 2.1726 \times 10^{-4} \ [\mathrm{psi/(kW\ sec)}] \ .$$

TABLE 2.3 Pressurizer Design Data.

Water volume, full power, ft^3 (m^3)	780 (22.09)
Steam volume, full power, ft^3 (m^3)	520 (14.72)
Electric heater capacity, kW (total)	1300

The change in mass in the pressurizer is obtained by summing the contribution due to expansion or contraction of the water in each coolant node in the primary loop as follows:

$$\delta M_w = \Sigma \ V_i \ \tau_i \ \delta T_{ci} \qquad (2.32)$$

or

$$\delta M_w = \Sigma \ V_i \ \tau_i \ \frac{d\,\delta T_{ci}}{dt} \qquad (2.32)$$

where V_i is the volume of the i'th coolant node, γ_i is the slope of the coolant density versus temperature curve, T_{ci} is the temperature of the i'th coolant node.

Evaluation of the coefficients for the H.B.Robinson NPP gives

$$\delta W_{w} = 85.33 \frac{d\delta T_{LP}}{dt} + 25.83 \frac{d\delta T_{c1}}{dt} + 25.83 \frac{d\delta T_{c2}}{dt}$$

$$+ 187.5 \frac{d\delta T_{UP}}{dt} + 39.2 \frac{d\delta T_{HL}}{dt} + 39.54 \frac{d\delta T_{IP}}{dt}$$

$$+ 171.55 \frac{d\delta T_{p}}{dt} + 39.54 \frac{d\delta T_{OP}}{dt} + 27.44 \frac{d\delta T_{CL}}{dt} \qquad (2.34)$$

where δT_{LP} is the reactor lower plenum temperature, δT_{c1} coolant temperature in node 1, δT_{c2} coolant temperature in node 2, δT_{UP} reactor upper plenum tempearture, δT_{HL} hot leg temperature, δT_{IP} temperature of primary coolant in the steam generator, δT_{p} temperature of primary coolant node in the steam generator, δT_{OP} temperature of primary coolant in the steam generator outlet plenum, δT_{CL} cold leg temperature.

In order to avoid discrepancies between theory an experiment a pressurizer control system is added. The controller parameters used belong to Sequoyah, a later-generation Westinghouse PWR and this was necessiated by the always recurring problem of lack of sufficent data.

The pressurizer controller uses a heater to compensate for steady state heat losses. It is also used for pressure control against normal pressure variations so that heat input increases for low pressure and decreases for high pressure. When the pressure goes above the control range, spray flow is used to decrease the pressure. The model used in this work includes a heater operating under normal conditions only.

The block diagram for the pressure controller is shown in Fig.2.3. The transfer functions are used to formulate differential equations for inclusion in the state variable model. The resulting equations are:

$$\begin{aligned}\frac{d\delta P_p}{dt} = {} & 0.0207\ \delta T_f - 0.0207\ \delta T_{c1} + 0.0103\ \delta T_{c2} \\ & + 0.240\ \delta T_{UP} - 0.130\ \delta T_{IP} - 0.509\ \delta T_p \\ & + 0.634\ \delta T_m - 0.116\ \delta T_{OP} + 0.121\ \delta T_{LP} \\ & - 0.279\ \delta T_{HL} + 0.0235\ \delta T_{CL} - 0.0106\ \delta P_p \\ & - 0.00213\ \delta X \end{aligned} \tag{2.35}$$

$$\frac{d\delta X}{dt} = 0.00556\ \delta P_p \tag{2.36}$$

where T_m is the steam generator tube metal temperature and X is the integral control action variable. Note that the inclusion of the pressurizer controller affects the matrix by modifying the coefficients in the differential equation for pressurizer pressure and requiring an additional equation to provide for the integral action.

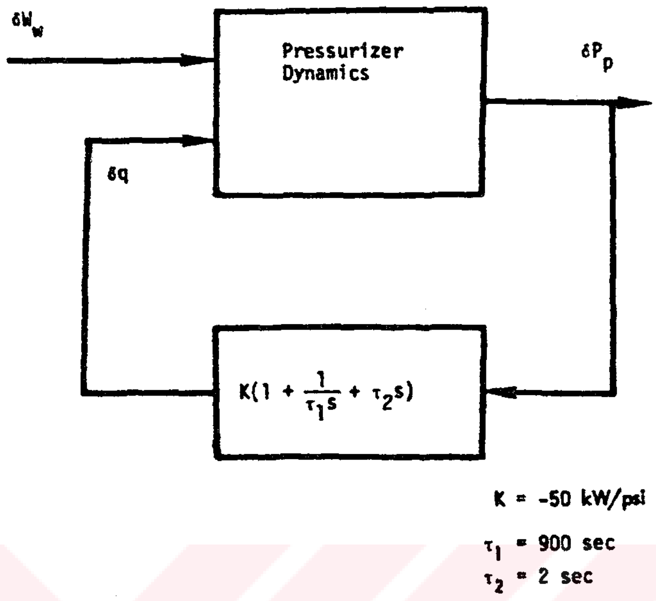

FIGURE 2.3. Pressurizer control system.

2.1.3. Steam Generator

The steam generator provides a dynamic link between the reactor core and the turbine generator in PWR type NPPs and therefore plays an important role in the safe and reliable operation of these plants. Next to the reactor, the steam generators are the most important components with respect to transient phenomena.

The physical processes that determine the thermal performance and operational behavior of the steam generator under steady-state and transient conditions include coupled two-phase flow, natural circulation, and heat transfer phenomena. A good understanding of, and the capability of predicting the normal and off-normal behavior of a steam generator are essential for evaluating the load following mechanism, the operational and accident conditions in PWRs. It is therefore generally necessary to model the steady-state and transient two-phase flow and void distribution in the steam generator to accurately predict the PWR plant response[25,26,27].

The behavior of steam generators is essentially nonlinear because of the nonlinear coupling between energy transport governed by heat transfer coefficients, and mass transport as determined by velocities. Although linearization is out of the question for investigating the large distrubances that are characteristic of safety assesement studies, it is acceptable for studying the control system design under normal operating conditions[27].

The steam generator model used in this work is a simplified one. It uses only three regions to represent the whole steam generator: primary fluid, tube metal, and secondary fluid. The model includes no control action. This is equivalent to assuming that the model applies only for small upsets in which the controller dead bands or long time constants prevent significant changes in the feedwater flow.

The equations are:

1. Primary water energy balance

$$\frac{d\delta T_p}{dt} = \frac{1}{\tau_{SG}} \delta T_{IP} - \frac{(hA)_{pm}}{M_p C_p} (\delta T_p - \delta T_m) - \frac{1}{\tau_{SG}} \delta T_p \quad (2.37)$$

2. Metal energy balance

$$\frac{d\delta T_m}{dt} = \frac{(hA)_{pm}}{M_m C_m} (\delta T_p - \delta T_m)$$

$$- \frac{(hA)_{ms}}{M_m C_m} [\delta T_m - (\frac{\partial T_{sat}}{\partial P_s}) \delta P_s] \tag{2.38}$$

3. Secondary water(liquid phase) mass balance

$$\frac{d\delta M_w}{dt} = \delta W_{FW} - \delta W_s \tag{2.39}$$

4. Secondary water (steam) mass balance

$$\frac{d\delta M_s}{dt} = \delta W_s - \delta W_{s,o} \tag{2.40}$$

5. secondary fluid (steam and liquid phase) energy balance

$$\frac{d\delta E}{dt} = (hA)_{ms} [\delta T_m - (\frac{\partial T_{sat}}{\partial P_s}) \delta P_s]$$

$$+ W_{FW} C_{PFW} [\delta T_{FW} - (\frac{\partial T_{sat}}{\partial P_s}) \delta P_s]$$

$$- h_{st} \delta W_{s,o} \tag{2.41}$$

6. equation of state

$$P_{sv} s = R M_s T_s \tag{2.42}$$

7. volume balance

$$V_s + V_w = V_T \tag{2.43}$$

where $_{ss}$ is the residence time of coolant temperature in the steam generator, h_{pm} is the heat transfer coefficent for primary coolant to metal(includes a portion of the metal resistance as well as the film resistance). h_{st} enthalp of steam, A heat transfer area, C_{FW} is the specific heat of feedwater, h_{ms} is the heat transfer coefficent for metal to secondary coolant (includes a portion of the metal resistance as well as the film resistance), M_m mass of tube metal, C_m specific heat of tube metal, $\partial T_{sat}/\partial P_s$ steam pressure, M_w mass of water in the steam generator, W_{FW} is the feedwater flow rate, M_s is the mass of steam in the steam generator, $W_{s.o}$ is the steam flow rate to the turbine, E internal energy of secondary coolant, V_s steam volume, T_s steam temperature, V_w water volume.

After linearization and appropriate algebraic substitutions the following equations are obtained:

$$\frac{d\delta T_p}{dt} = \frac{1}{_{SG}} \delta T_{IP} - \frac{(hA)_{pm}}{M_p C_p} (\delta T_p - \delta T_m) - \frac{1}{\tau_{SG}} \delta T_p \qquad (2.44)$$

$$\frac{d\delta T_m}{dt} = \frac{(hA)_{pm}}{M_m C_m} (\delta T_p - \delta T_m)$$

$$- \frac{(hA)_{ms}}{M_m C_m} [\delta T_m - (\frac{\partial T_{sat}}{\partial P_s}) \delta P_s] \qquad (2.45)$$

$$\frac{d\delta P_s}{dt} = D_1 \delta P_s + D_2 \delta T_m + D_3 \delta T_{FW} + D_4 \delta W_{FW} + D_5 \delta W_{so} \qquad (2.46)$$

where D_i are the coefficients obtained from algebraic substitutions.

Numerical values for the coefficients are obtained using the H.B. Robinson Nuclear Plant design data given in Table 2.4. The resulting equations for the steam generator are :

$$\frac{d\delta T_p}{dt} = 0.2238\ \delta T_{IP} - 0.76642\ \delta T_p + 0.53819\ \delta T_m \qquad (2.47)$$

$$\frac{d\delta T_m}{dt} = 3.07017\ \delta T_p - 5.3657\ \delta T_m + 0.33272\ \delta P_s \qquad (2.48)$$

$$\frac{d\delta P_s}{dt} = 1.349\ \delta T_m - 0.2034\ \delta P_s + 0.05328\ \delta T_{FW}$$

$$- 0.03843\ \delta W_{FW} - 0.04425\ \delta W_{SD} \qquad (2.49)$$

TABLE 2.4. Steam Generator Data (for each unit)

Steam Generator	
Number of U-tubes	3260
U-tube diameter, In. (cm)	0.875 (2.22)
Average tube wall thickness, in. (cm)	0.050 (0.13)
Mass of U-tube metal, lb (kg)	91 800 (41 677)
Total heat transfer area, ft^2 (m2)	44 430 (4127.7)
Steam Conditions at Full Load	
Steam flow, lb/h (kg/s)	3.169×10^6 (400)
Steam temperature, F (C)	516 (268.89)
Steam pressure, psig (MPa)	770 (5.3)
Primary Side Coolant	
Reactor coolant flow, lb/h (kg/s)	33.93×10^6 (4279)
Reactor coolant water volume. ft^3 (m^3)	928 (26.28)
Secondary Side Fluid	
Feedwater temperature, F (C)	435 (223.89)
Secondary side water volume, full power, ft^3 (m^3)	1526 (43.21)
Secondary side steam volume, full power, ft^3 (m^3)	3203 (90.70)

2.1.4. Piping and plenums

All piping sections and plenums are modelled as well-mixed volumes:

$$\frac{d\delta T}{dt} = \frac{1}{\tau}\delta T_{in} - \frac{1}{\tau}\delta T \qquad (2.50)$$

where T is the temperature of fluid in the section(equal to outlet temperature), T_{in} is the fluid temperature at entrance and τ is the fluid residence time. There are two piping sections; hot-leg and cold-leg and four plenums; reactor upper, reactor lower, steam generator inlet, and steam generator outlet plenums in the model.

Substitution of numerical values for each one of these equations yields:

$$\frac{d\delta T_{UP}}{dt} = 0.33645\ \delta T_{C2} - 0.33645\ \delta T_{UP} \qquad (2.51)$$

$$\frac{d\delta T_{HL}}{dt} = 2.5\ \delta T_{UP} - 2.5\ \delta T_{HL} \qquad (2.52)$$

$$\frac{d\delta T_{IP}}{dt} = 1.45\ \delta T_{HL} - 1.45\ \delta T_{IP} \qquad (2.53)$$

$$\frac{d\delta T_{OP}}{dt} = 1.45\ \delta T_{p} - 1.45\ \delta T_{OP} \qquad (2.54)$$

$$\frac{d\delta T_{CL}}{dt} = 1.48\ \delta T_{OP} - 1.48\ \delta T_{CL} \qquad (2.55)$$

$$\frac{d\delta T_{LP}}{dt} = 0.516\ \delta T_{CL} - 0.516\ \delta T_{LP} \qquad (2.56)$$

2.1.5. Overall System

The nodal structure for the complete system is shown in Fig.2.4.

The linear model obtained by assuming negligible change in feed water temperature and feed flow from the steady-state values is one of a two input one :

1) The reactivity due to control rod movement,

2) The deviation in the steam flow rate from its steady state value W_{so}.

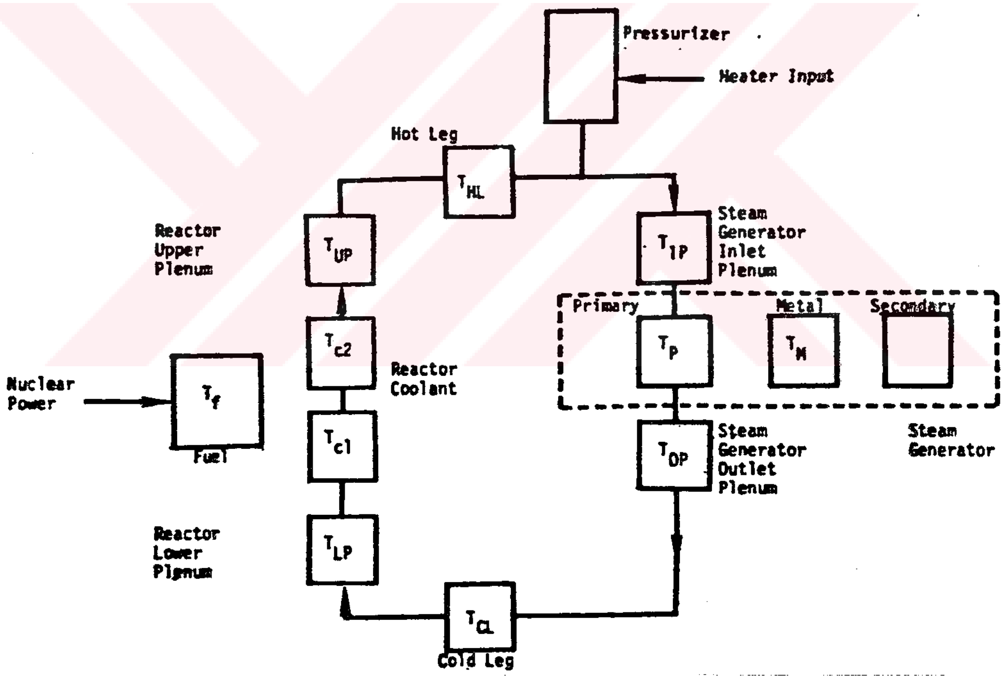

FIGURE 2.4. Nodal structure for complete model.

The model can be represented in the standard state-space form as

$$\dot{\underline{x}} = \underline{\underline{A}}\ \underline{x} + \underline{b}u \tag{2.57}$$

where $\underline{x}$ is the (21 x 1) state vector given by

$$\underline{x}^T = (\delta P,\ \delta C_1,\ \delta C_2,\ \delta C_3,\ \delta C_4,\ \delta C_5,\ \delta C_6,\ \delta T_f,\ \delta T_{c1},\ \delta T_{c2},\ \delta P_p,\ \delta X,\ \delta T_p,\ \delta T_m,\ \delta P_s,\ \delta T_{UP},\ \delta T_{HL},\ \delta T_{IP},\ \delta T_{OP},\ \delta T_{CL},\ \delta T_{LP}) \qquad (2.58)$$

2.1.6 Simplified Model

In order to compare the results of this work with previously designed analytical controllers the following simplified model used by other authors[10] is also considered.

The six groups of delayed neutrons are reduced to one group by evaluating a single decay constant from the weighted harmonic mean of the six group decay constants:

$$\lambda = \left(\frac{1}{\beta} \Sigma \frac{\beta_i}{\lambda_i} \right)^{-1} \qquad (2.59)$$

The other assumption is that the size of the pressurizer is large enough to accommodate the steam generator primary volume surges, thus inclusion of pressurizer dynamics in the system model can be avoided.

This model too, can be represented in the state space form of Eq.(2.57) this time $\underline{x}$ being the (14x1) state vector given by

$$\underline{x}^T = (\delta P,\ \delta C,\ \delta T_f,\ \delta T_{c1},\ \delta T_{c2},\ \delta T_p,\ \delta T_m,\ \delta P_s,\ \delta T_{UP},\ \delta T_{HL},\ \delta T_{IP},\ \delta T_{OP},\ \delta T_{CL},\ \delta T_{LP}) \qquad (2.60)$$

and hence will be called as the 14 variable model. The system distribution matrix $\underline{\underline{A}}$ is given in Fig.2.5.

$$\begin{matrix}
-400 & 0.07686 & -1781 & -13700 & -13700 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
400 & -0.07686 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.0756 & 0 & -0.16466 & 0.16466 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05707 & -2.44030 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.38 \\
0 & 0 & 0.05707 & 2.32620 & -2.38320 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -0.76642 & 0.53819 & 0 & 0 & 0 & 0.22360 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 3.07017 & -5.36570 & 0.33272 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1.349 & -0.20340 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.33645 & 0 & 0 & 0 & -0.33645 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.5 & -2.5 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.45 & -1.45 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1.45 & 0 & 0 & 0 & 0 & 0 & -1.45 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.48 & -1.48 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.516 & -0.51
\end{matrix}$$

FIGURE 2.5 System distribution matrix $\underline{\underline{A}}$ for 14-variable model.

2.2 Numerical Simulation of the NPP

The reactor kinetics equations have the property of stiffness arising from the differences in orders of magnitude between the prompt and delayed neutron genertaion times which puts a restrictive upperbound on the time steps to be used for the numerical solutions[28]. Various methods have been devised to overcome stiffness for the general nonlinear ordinary differential equations with varying degrees of success[29,30,31]. However, the linearized form of the the mathematical model of the NPP given by Eq.(2.57) allows the use of the method suggested by Kerlin et al.[19].

This method uses a matrix-exponential type of solution. The output at time, t+Δt, is given by

$$\underline{x}(t+\Delta t)=\exp(\underline{\underline{A}}\Delta t)\underline{x}(t) + \int_{t}^{t+\ t} \exp[\underline{\underline{A}}(t-t')].[\underline{b}u(t)]\ dt' \quad (2.61)$$

since $\underline{b}u(t)$ can be assumed to be piecewise constant, the integral in Eq.(2.61) can be evaluated and

$$\underline{x}(t+\Delta t)=\exp(\underline{\underline{A}}\Delta t)\underline{x}(t) + [\exp(\underline{\underline{A}}\Delta t) - \underline{\underline{I}}]\underline{\underline{A}}^{-1}\underline{b}u(t) \quad (2.62)$$

can be obtained as the solution. The terms involving matrix exponentials can be evaluated as follows:

$$\exp(\underline{\underline{A}}\Delta t)= \underline{\underline{I}} + \underline{\underline{A}}\Delta t + \frac{1}{2!}(\underline{\underline{A}}\Delta t)^2 + \frac{1}{3!}(\underline{\underline{A}}\Delta t)^3 + \ldots \quad (2.63)$$

$$[\exp(\underline{\underline{A}}\Delta t) - \underline{\underline{I}}]\underline{\underline{A}}^{-1}= t[\underline{\underline{I}} + \frac{1}{2!}\underline{\underline{A}}\Delta t + \frac{1}{3!}(\underline{\underline{A}}\Delta t)^2 + \ldots] \quad (2.64)$$

where $\underline{\underline{I}}$ is the identity matrix.

The features of this method that make it suitable for numerical simulations on microcomputers are as follows:
(a) Expansion in Eq.(2.64) avoids the need for a matrix inversion which is an operation with well-known pitfalls;
(b) The matrices in Eqs.(2.63) and (2.64) need to be computed only once, at the begining of the simulation, and the output at any t can be obtained by simple matrix-vector multiplications; (c) Any number of terms in the expansion can be taken which will allow the user to have a compromise on accuracy and simulation time. However, the time step can be changed only by calculating the matrices in Eqs.(2.64) and (2.65) with the new value of t. Nevertheless, even this is not a

severe restriction because other methods require calculation of the Jacobian matrix at every time step and a matrix inversion as well[29,30,31].

III. RULE-BASED FUZZY LOGIC PROCESS CONTROL

Industrial process control is based on effective analytical methods developed using control theory for designing a controller. The level of technology achieved is an indisputable evidence of the success of these methods. Yet, there are several assumptions that are often difficult to justify but nonetheless made in using the analytical methods. The first assumption is that a precise mathematical model of the process to be controlled can be formulated. Also inherent in control theory is the assumption that a precise model of the corrective process is available. However, most of the industrial processes ,i.e., those that have nonlinear relations between the system state and the control variables do not permit the required precise mathematical modeling. The other assumption is that it is always possible to measure the variations in the conditions involved in the process[2]. Another fundamental but not well founded assumption is that the concept can be implemented as it is designed. However, this is not the case: to obtain say a good PID regulator it is also necessary to consider operator interfaces, operational issues like switching smoothly between manual and automatic operation, transients due to parameter changes, the effects of nonlinear actuators, wind-up of the integral term etc. An operational industrial PID regulator consists of an implementation of the basic control law and heuristic logic that takes care of the above enumerated issues. Similarly, during startup testing and commissioning, it is necessary to tune the regulator parameters of NPP control systems, which is a time consuming and tedious work taking about 12 days to complete while the plant undergoes about 600 forced disturbances having both safety and economical impacts(32).

Complex industrial processes are successfully controlled despite the aforementioned shortcomings of the control theory. They are controlled by human operators who are able to cope with the imprecision involved by developing new skills or heuristics in time. Until recently a theoretical approach toward a consideration of the heuristic factors inherent in the implementation was not made even though their strong influence on the operation of the controller was well appreciated. Instead they were hidden in practical designs. Due to both the difficulty in theoretical analysis and indifference in this respect of many researchers(33). The idea of making the implicit use of heuristics explicit has led to the application of expert system technology to the control area. The key idea behind these developments is that it should be possible to implement fuzzy logic control within the domain in which the process can be controlled successfully by a human operator. However, the value of this technology to the operation of well-charcterized systems has not been clearly demonstrated and this remains the issue to be addressed by the nuclear community(34).

In general the same steps are used for designing an analytical controller and a fuzzy, rule based one namely, the plant process is identified, and a control methodology is developed. However, there is a marked difference between the details of the two approaches[3].

In the design of an analytical controller, the plant design engineers construct a suitable mathematical model which is used together with the performance specifications by the control specialist to do the actual design. However, since there is no such mathematical model in the case of rule-based controllers, situation/action rules are used which necessitates the control designer to be intimately familiar with the plant's operation. Therefore, the plant engineer must also perform this duty by learning the rule-based methodology himself.

Also due to lack of a mathematical model suitable for design, new methods for acquiring the necessary knowledge must be employed. Information on operating rules are obtained from two sources: by observing plant operation and the operators themselves. Acquiring knowledge in this way is a tedious and often frustrating process because often the operator himself is not aware of his actions or can not communicate them effectively for some reason or other, also, distinguishing between the relevant and irrelevant information is a very demanding task. When no expert is available, the primary way of acquiring knowledge becomes simulation of the process where its behavior under normal and transient operating conditions is observed as is the case in this study.

All the controllers are expected to satisfy some performance criteria which influence the design and implementation process. The most important ones among these are that the plant should respond in minimum time, therefore the maximum stress levels will be observed during transients, and that it should be stable etc. In the case of rule-based controllers there is no direct way to use these criteria. The controller must be constructed and then tested to determine its performance. However, this process, by no means exhaustive, is never satisfactory, i.e., it is not possible to establish with certainty that all the possible contingencies have been anticipated and an appropriate corrective action has been implemented.

As already mentioned before, all the controllers must be tuned before they are used. Unlike analytical controllers there is no standard technique for calibration of rule-based controllers and instead they require tedious iterative closed-loop trials, to determine the membership functions and the effectiveness of rules.

Despite these disadvantages there are cases where the rule-based controllers are superior due to their flexibility and robustness. These factors are especially important when the model is inaccurate, signals are noisy, or some parameters assumed to be constant are really a function of time. Of course when no model of the process is available then they are the only automatic controllers available.

The historical development of the fuzzy logic controllers used in industrial processes starting with the pioneering work of Mamdani and Assilian is reviewed in reference 2. Although the concept of fuzzy logic control was devised nearly ten years ago, mainly due to the reluctance of the nuclear industry to use digital control methods, fuzzy logic control applications are very recent and in their early stages[34].

Rule-Based controllers (RBC) are a subset of knowledge-based systems(KBS), rules being a knowledge representation method.

A knowledge-based system is an artificial intelligence(AI) program whose performance depends more on the explicit presence of a large body of knowledge than on the possession of ingenious computational procedures. Expert systems which are a subset of knowledge-based systems seek to model the knowledge and procedures used by a human expert in solving problems within a well-defined domain. However, for many AI applications there are no uniquely qualified human experts[35] as is the case in this study due to the fact that the process to be controlled is too fast for a human being.

In KBSs, computational steps are separate from the control flow as opposed to the conventional computer programs where the information is scattered throughout the code. The domain specific knowledge such as facts and rules or other representations that use those facts as the basis for decision making, resides in the part called the <u>knowledge</u>

base, whereas the general problem-solving knowledge called the inference engine contains an interpreter that decides the order in which the rules should be applied. An additional facility called the user interface is also necessary for modification and explanatory purposes[36]. The organization of a KBS is shown in Fig.3.1.

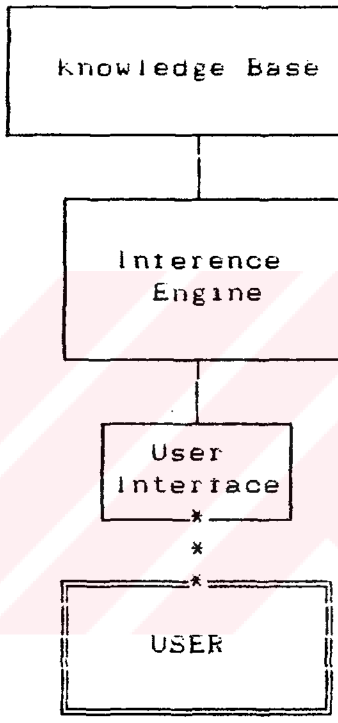

FIGURE 3.1 A block diagram of an expert system.

In RBCs, the model of the process is represented by using rules rather than mathematical equations. However, as an expert system application RBCs are rather simple constructs, due to the fact that their knowledge base is limited and rules can be grouped allowing the use of rather simple inference mechanisms. In the case of fuzzy logic control, forward chaining is sufficient, and in general conflict resolution strategies are inherent in fuzzy logic as will be explained below.

Since rules are used for knowledge representation; only an empirical representation is possible. Because one of the main reasons for using this kind of control is the lack of availability of deep representations or hardness thereof, this is entirely appropriate.

3.1 Knowledge Representation

The success of an AI program depends on effective knowledge representation and integrating different kinds of knowledge into a coherent knowledge base to support the system's activities[37] (see Fig.3.2). A representation in this context can be defined as a set of syntactic and semantic conventions which tell a computer how to interpret symbol structures. The syntax specifies the symbols that may be used and the ways to arrange them whereas, the semantics specifies how meaning is embodied in the symbols and the symbol arrangements allowed by the syntax[38].

Early attempts at building intelligent systems used first-order-predicate calculus as their representation language. The logical approach has the intuitive appeal for knowledge representation because it has a very general expressive power and mathematical deduction can be used to derive new knowledge from old. Although logic is unmatched for the problems it is suited for, there are cases where the following weaknesses must be weighed against other available methods: Since theorem proving programs require search, solution may take too long to be found. Some knowledge can not be represented as axioms, formulation of the problem in logic may require unnecessary effort while solving the problem formulated in another way may be simple. Also, logic does not allow the expression of some obvious heuristic knowledge[38,39].

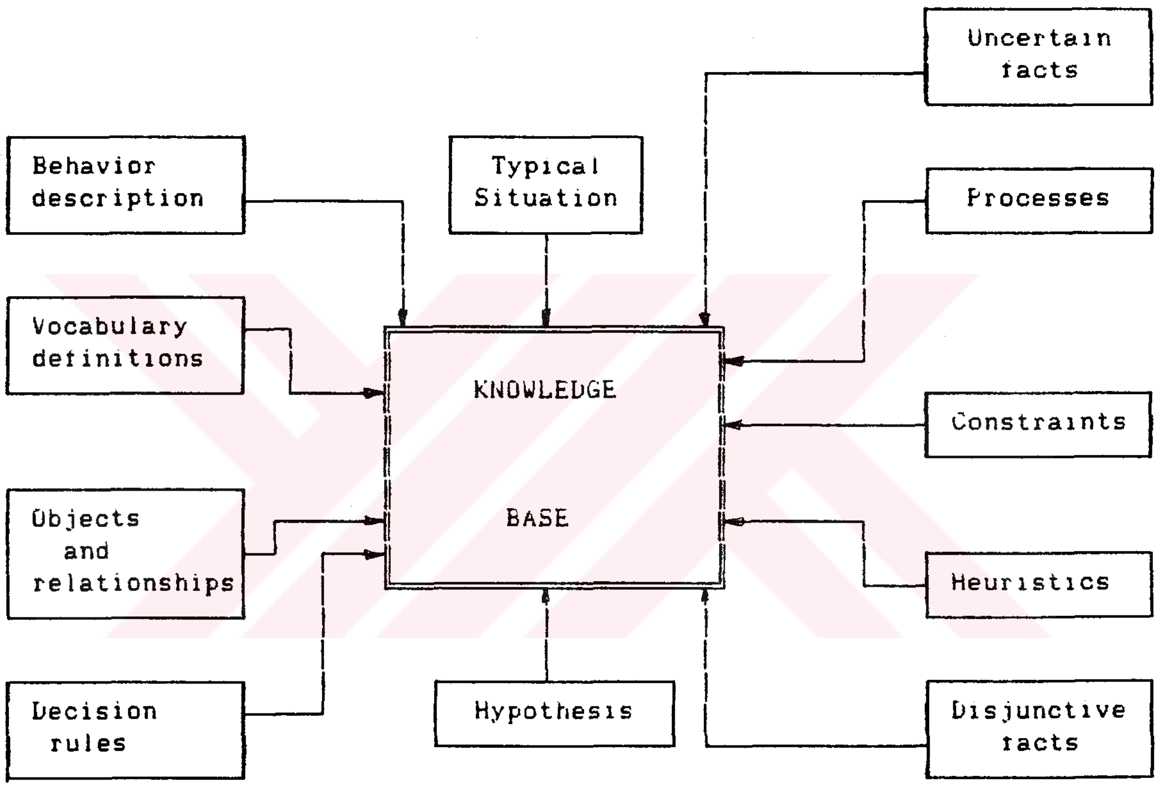

FIGURE 3.2 The kinds of knowledge that can go into a knowledge base.

Psychological research suggests that humans do not exhibit the kinds of reasoning behavior that is associated with theorem proving systems, rather people prefer reasoning from situations to actions. The most popular and effective representational form for declarative descriptions of domain dependent behavioral knowledge in a knowledge-based system therefore has been pattern/action decision rules, called production rules[38,39]. Each rule consists of an if part and a then part, i.e.

If antecedent then consequent.

Given the antecedent as a fact it is concluded by the system that the consequent is true and is added to the knowledge base as a new fact. Production rules are, in this sense, effectively a subset of the predicate calculus with an added prescriptive component indicating how the information in the rules is to be used during reasoning. The difference lies in the fact that the connection between the antecedent and the consequent is rather empirical, i.e., it is often not possible to prove that certain actions are logical consequences of certain situations[40]. Production rules can easily be understood by domain experts and have sufficient expressive power to represent a useful range of domain-dependent inference rules and behavior specifications. However, their expressive power is inadequate for defining terms and for describing domain objects and static relationships among objects. In this respect they allow only a surface representation.

However, a rule-based system can easily explain the why and how of its inference processes which is a very important asset due to the fact that sometimes the user may doubt the conclusions reached and may want to check the line of inference so that he may use the result or reject it. Such a query may help to improve the rule base; the necessity of adding new rules, modifying or even deletion of some existing ones may become apparent.

Semantic nets and frames are also used for knowledge representation(36,37,38,39).

A semantic net consists of nodes which stand for concepts or objects or events and are connected by arcs describing the relations between the nodes. It is a useful way to represent knowledge in domains that use well-established taxonomies to simplify problem solving.

A frame is a structured representation of an object or a class of objects. Like a semantic net it is a network of nodes and relations organized in a hierarchy, where the topmost nodes represent general concepts and the lower nodes more specific instances of these concepts. The difference lies in the fact that in a frame system the concept at each node is defined by a collection of attributes and values for those attributes that are called slots. Each slot can have procedures attached to it which are executed when the information in the slot is changed.

The control strategy in RBCs is represented as production rules which model the operator actions. The antecedent generally consists of the deviation of the observed variables from the setpoint and their rates of change. The consequent part of the rule applies to the manipulated process variables which can be stated in terms of the change to the level of input, or the absolute level of input. The following is an example rule:

> If pressure error is negative small and change in pressure error is negative small, then heat change is positive medium.

A closer look at this rule will reveal the important features concerning the similarities and differences between the conventional and rule-based controllers. Firstly, the term "pressure error" means that this rule is formulated with respect to a pressure setpoint at which the pressure should be held. Also, the antecedent of the rule has not only the deviation from the setpoint but its rate and direction of change. In this respect there is a very definite similarity between the proportional and derivative terms of a conventional PID controller[41,42]. Finally, the rules are expressed using linguistic variables such as "pressure error", "change in pressure error" and "heat change" which can take the fuzzy values "negative small", "positive big" etc. Although these are the terms human beings can comfortably work with it is difficult to implement them on digital computers. When trying to control highly nonlinear and ill-understood processes, this can be as precise a model of how to control the plant as is available.

3.1.1 Representing Inexact Knowledge

One of the difficulties of implementing KBSs is that a complete understanding of the complex domain encountered in a real world situation is generally not available. Much human knowledge is vague and imprecise. Human thinking and reasoning frequently involve inexact information. Nevertheless, the experts have heuristics that are formed in time from experience or some abstract mental models that allow them to perform efficiently in their particular domains.

If an expert system is to exhibit expert behavior then it must have knowledge representation schemes that can encode uncertain knowledge from the possible sources: (a) inherent human fuzzy concepts; (b) unreliable information; (c) matching of similar rather than identical experiences; (d) incomplete information; and (e) differing (expert) opinions.

In classical logic all the propositions are either true(T) or false(F). In order to express uncertain knowledge, a scheme which allows a proposition to have a truth value other T or F is necessary. One approach is to consider a range of truth values extending from definite truth to definite falsity with values allowed in this interval. This new truth value can either be a numerical value between 0 and 1, representing a degree; or a qualitative label, such as "almost true", which is defined as a partition of the truth space[69].

The usual approaches to inexact knowledge in expert systems are: (a) Bayesian approach; (b) certainty factors; (c) Dempster-Shafer theory of evidence; (d) fuzzy logic.

3.1.1.1 Bayesian Approach. Based on probability theory, the Bayesian approach can deal only with uncertainty. Uncertainty in a proposition is represented as a probability between 0 and 1. Bayes' theorem states that, if $E = \{E_1, E_2, \ldots, E_n\}$ is a set of n pieces of evidence and $H = \{H_1, H_2, \ldots, H_m\}$ is a set of m mutually exclusive hypotheses that are under consideration, then

$$P(H_i | E_j) = \frac{P(E_j | H_i)}{P(E_j)} P(H_i)$$

with

$$P(E_j) = P(E_j | H_i) P(H_i)$$

where $P(H_i)$ is the probability of the hypothesis prior to the knowledge of evidence, $P(E_j | H_i)$ is the conditional probability of the evidence E_j given the hypothesis H_i, and $P(H_i | E_j)$ is the posterior probability of the hypothesis after E_j is observed.

This formula can be used as a rule of inference in an expert system. If the knowledge base contains the rule 'if E_j then H_i' and E_j is true then the Bayes theorem updates the belief in H_i from $P(H_i)$ to $P(H_i | E_j)$, provided that $P(E_j | H_i)$ and $P(E_j)$ are known.

Collecting or estimating all the prior conditional and joint probabilities required for this method is difficult for domain experts. However, it has been suggested that employing conditional independence assumptions can reduce the number of probabilities to be estimated. This approach depends also on the availability of a complete set of hypotheses, hence its applicability is restricted[43]. Another major criticism of the use of subjective probability, in this approach, is that it is not possible to represent ignorance. This means that if a piece of evidence partially supports a hypothesis, it would also have to be partially in favor of the negation of that hypothesis. Since people often distinguish between supporting and refuting evidence this is counter-intuitive. Also probabilities can only be assigned to a singleton hypothesis and they must sum up to one[44].

<u>3.1.1.2 Certainty Factors.</u> This approach can deal only with uncertainty. A certainty factor CF(h,e) is a numerical value between zero and one that stands for the degree of confirmation of the hypothesis h based on the evidence e. Certainty factors are used in the Mycin system to handle uncertainty in evidence (facts) and rules. For example:

```
rule:
  IF X is a bird,
  THEN it can fly. (CF=0.9)
fact:
  X is a bird.      (CF=0.8)
conclusion:
  It can fly.       (CF=0.9*0.8=0.72)
```

One advantage of this approach over probability theory is that it does not require prior probabilities and therefore does not require a large volume of statistical data. Moreover, experts are more comfortable assigning certainty factors to the facts and rules. Certainty factors are widely used in expert system shells to handle uncertainty[43,44].

3.1.1.3 Dempster-Shafer Theory of Evidence. The Dempster-Shafer theory calculates belief functions-measurements of the degree of belief. For a set of mutually exclusive hypothesis $H = \{H_1, H_2, \ldots, H_n\}$ the theory allows part of the unity belief to be attributed to any subset of H or any disjunction of H_is. The distribution of the belief over the hypothesis set is called a basic probability assignement m, which has to satisfy the following conditions:

$$\sum_{A_i H} m(A_i) = 1$$

and

$$m(\emptyset) = 0$$

The interpretation of the basic probability for a given set of elements is the amount of belief that is committed exactly to that set, but cannot be subdivided into any subset of itself. Another property of this theory is that, if one attributes part of one's belief to a proposition, the rest of the belief does not have to be assigned to the negation of that proposition. Disbelief and ignorance are distinguished in the representational framework.

In this theory, uncertainty in a proposition is characterized by two values: degrees of belief which is a measure of the evidence for the proposition and plausibility which is defined as 1 - measure of evidence against proposition[43].

This approach, however, involves many numerical computations, and in the case of a long inference chain the structure of the resulting belief function would be very complex[42,43].

3.1.1.4. Fuzzy logic. Fuzzy logic is based on fuzzy set theory. In ordinary set theory an item is either a member of a set or not. However, due to the observation that in the real world membership in a set is not so crisp , i.e., certain sets have imprecise boundaries, fuzzy set theory was developed. An item can be a member of a fuzzy set which is an ill-specified and not a distinct collection of objects with unsharp boundaries to a varying degree ,i.e., transition from membership to nonmembership is gradual rather than abrupt. The degree of membership is determined by its membership function[2,43,44,45].

Fuzzy logic is concerned with the formal principles of approximate reasoning, with precise reasoning viewed as a limiting case. Unlike classical logical systems, it aims at modeling imprecise modes of reasoning that play an essential role in the human ability to make rational decisions in an environment of uncertainty and imprecision. Human beings communicate with each other and reason using seemingly vague concepts without much difficulty and adapt to unencountered situations easily. It rarely occurs to the user that the statement "She is tall" is essentially imprecise in the sense that tallness is not a crisp quality and has different meanings to different people but the important factor that is to be emphasized here is that difference is actually a matter of degree. In other words, an actual measurement of the height of a person may be considered as being tall to a degree by some person and not very tall by another. In this context, height of a person is a linguistic variable which can take values as short, not short, tall, very tall, etc. A more formal definition of linguistic variable is : "a variable whose values are words or sentences in a natural or artificial language."(46)

Linguistic variables take on specific linguistic values which are expressed as fuzzy subsets of the corresponding universes(also called support sets) to which they refer.

A fuzzy subset A of X is represented by a membership function: $\mu_A : X \rightarrow [0,1]$. Here as in our example X can be height of people which is a nonfuzzy support set of a universe of discourse, and A can be the linguistic value such as tall people. Given two such linguistic values A_1 and A_2 on the same support set X the following logical combinations can be defined(41) :

- Complement of A_1 (NOT A_1) is formed by taking $(1-\mu_A)$ as its membership value at each element of the support set.
- A_1 OR A_2 ($A_1 \vee A_2$) is formed by taking $\max(\mu_{A1}, \mu_{A2})$ at each element of the support set.
- A_1 AND A_2 (A_1 A_2) is formed by taking $\min(\mu_{A1}, \mu_{A2})$

at each element of the support set.

One of the major differences of fuzzy logic from other logics is in the nonunique definition of implication. A class of implications have been defined so far, satisfying different kinds of properties where the selection must be subjectively made with regard to the behavior of the inference process[47].

Given a rule of the form "if X is A, then Y is B" the value of implication $\mu_R(x,y)$ is related to $\mu_A(x)$ and $\mu_B(y)$ by the following

$$\mu_R{}^1(x,y) = 1 - \mu_A(x) + \mu_A(x)\mu_B(y)$$

$$\mu_R{}^2(x,y) = \max(1 - \mu_A(x), \min(\mu_A(x), \mu_B(y)))$$

$$\mu_R{}^3(x,y) = \min(\mu_A(x), \mu_B(y))$$

$$\mu_R{}^4(x,y) = \begin{cases} 1 & \text{if } \mu_A(x) \quad \mu_B(y) \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_R{}^5(x,y) = \max(1 - \mu_A(x), \mu_B(y))$$

$$\mu_R{}^6(x,y) = \begin{cases} 1 & \text{if } \mu_A(x) \quad \mu_B(y) \\ \mu_B(y) & \text{otherwise} \end{cases}$$

Using one of these together with the rule defining the relation between the antecedent and consequent one can infer the consequent B' when some value A'(which may be different from A) is given.

Using fuzzy logic it is possible to represent operator action in a form suitable for inferencing to generate control action with a digital computer[3,41,49-55]. In this work the a similar approach is taken. The control rules are formulated in linguistic terms which can be defined on a mathematical scale using fuzzy sets to describe the magnitude of error and its rate of change and the magnitude of the appropriate control action. The factors that affect the number of terms required are, the fineness of control rules and whether the application is process regulation or servocontrol.

The established linguistic terms are Error(E) and Change in Error(CE) which can be complemented in some applications with Change in Change in Error(CCE).

The most common labels used are of the form "positive big", "positive medium", "positive small", "zero", "negative small", "negative medium", and "negative large". These labels are expected to cover the allowable range of the linguistic variable. These fuzzy sets each have membership functions which give the degree of the each measurement in these fuzzy sets. It is important to note here that each measurement can be a member of more than one set to a varying degree. Therefore, in this sense membership grades are not probabilities, since there is no randomness involved. In probabilistic statements the imprecision is about the outcome of an event whereas in a possibilistic statement the imprecision is about the vagueness of the concepts involved[2].

There are no established methods for specifying membership functions, they can be continuous, piecewise continuous or sometimes no functional form is used at all, instead a lookup table is formed. The factors of importance to note are the inferencing mechanism to be used, the capacity in terms of memory and speed of the computer where the implementation will be realized, the number of rules, variables and degree of quantization. Most common membership functions are the continuous S-shaped function shown and the piecewise continuous broken-line function shown.

The S-shaped function is given by the formula

$$\mu(x) = (1+(a(x-c))^{b})^{-1} \qquad (3.1)$$

as shown in Fig.3.3. The desired shape of the fuzzy set can be adjusted by the three parameters: c alters the point of minimum fuzziness (μ=1), a the spread and b the contrast.

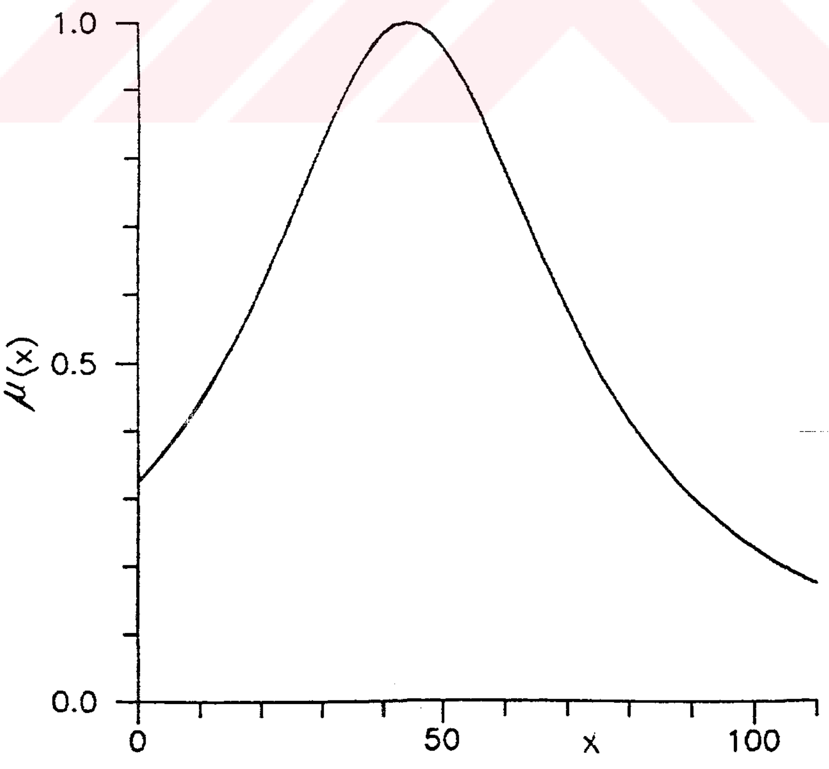

FIGURE 3.3 S-shaped fuzzy set.

The broken-line function is given by the formula

$$\mu(x) = \begin{cases} a_1 x + b_1 & \text{if } c_2 < x < c_1 \\ \cdot & \\ \cdot & \\ \cdot & \\ a_{n-1} x + b_n & \text{if } c_{m-1} < x < c_m \end{cases} \qquad (3.2)$$

as shown in Fig.3.4. This form is in essence similar to the lookup table concepts which requires an interpolation scheme in any case. The shape of the function can be adjusted by changing the constants a_i, b_i and c_i.

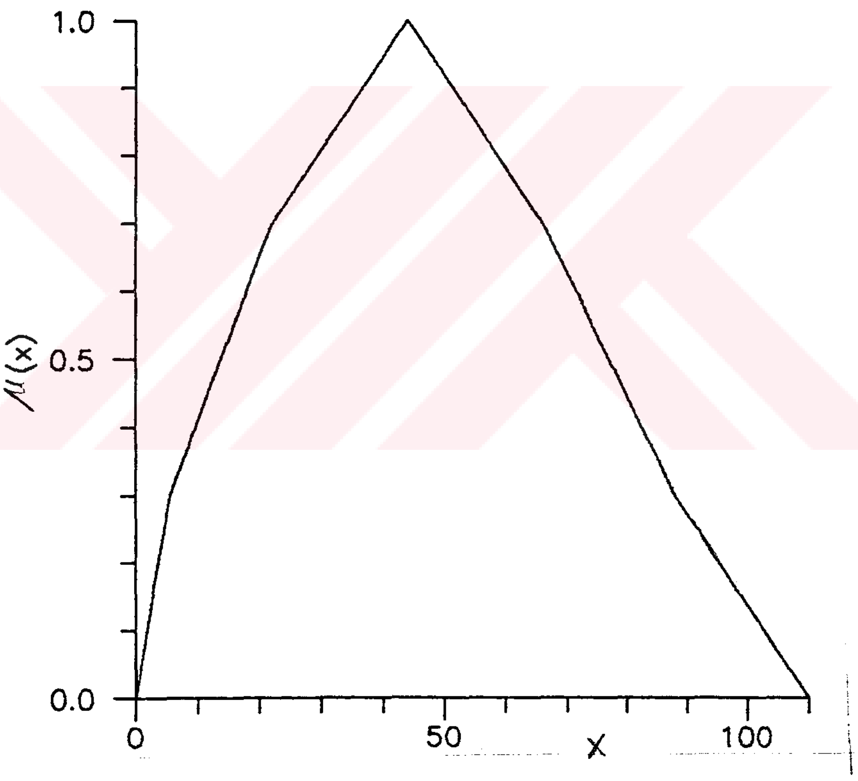

FIGURE 3.4 Broken-line fuzzy set.

3.2 Inference Mechanism

There are two important ways in which rules can be used in a KBS: (a) forward chaining; (b) backward chaining.

The name forward chaining comes from the fact that in this technique movement is from condition specifying if parts to action specifying then parts. When all the conditions in a rule are satisfied by the current situation the rule is said to be triggered. When actions are performed, the rule is said to be fired. Triggering does not always mean firing, because the conditions of several rules may be satisfied simultaneously, triggering them all, making it necessary for a conflict resolution procedure to decide which rule actually fires[38].

If the rule-based system hypothizes a conclusion and uses the antecedent-consequent rules to work backward toward the hypothesis-supporting facts, then such a system is called backward chaining or goal driven.

The purpose of reasoning and the shape of the search space determines the method of chaining. If the goal is to infer one particular fact, then backward chaining must be used.

In this study, of the two inference mechanisms only forward chaining is used since there is no need to generate and test hypotheses. In this respect RBCs are rather simple KBSs. The block diagram of a RBC is shown in Fig.3.5.

Another and more important simplification is due to the fact that all chains of reasoning are one inference long for the conclusion of one rule can never participate in the antecedent of another because the observed and manipulated variables are different.

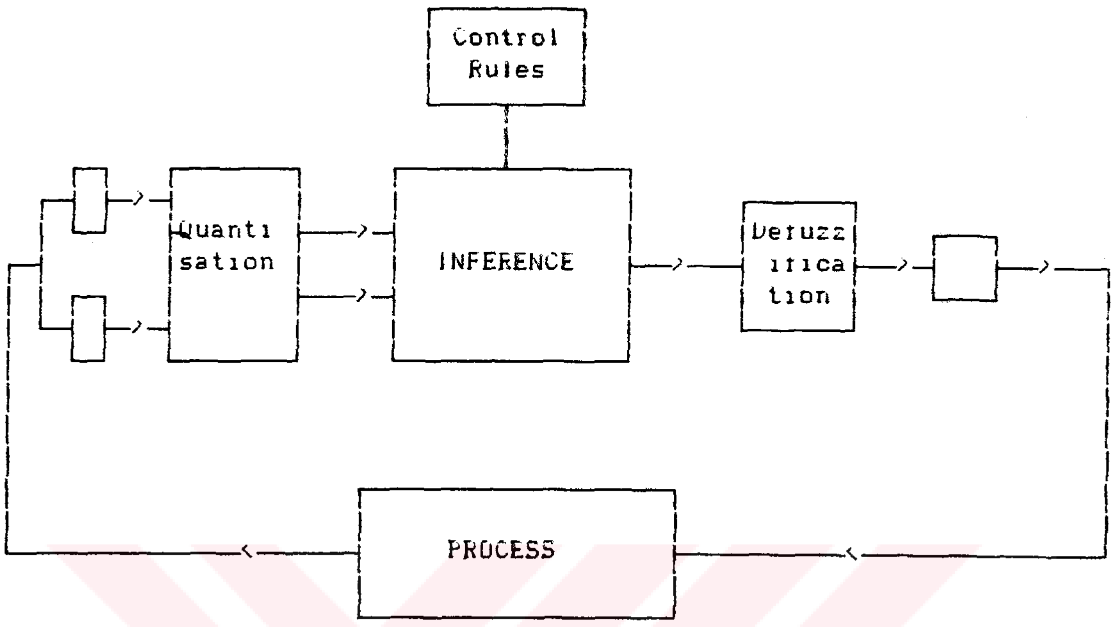

FIGURE 3.5 The block diagram of RBC

Inference process starts with the determination of the degrees of membership of E and CE in the allowable fuzzy sets. Once these quantized inputs are obtained they are compared with each rule in turn. Each rule's degree of fulfillment(DOF) is determined using the equivalents of the AND and OR operations on the fuzzy sets in the antecedent which may be written as

$$DOF_i = \min(\mu_{E_i}(e),\ \mu_{CE_i}(ce)) \quad i=1,..,n \qquad (3.3)$$

where E_i is a term defined on the Error scale, CE_i is a term on the Change in Error scale both of which are the terms in the antecedent of rule i. DOF_i is the calculated degree of fulfillment of rule i. e and ce are the scaled measurements on the Error and Change in Error scales. The total number of rules is denoted by n. The DOF of a rule is a real number in the range [0,1]. If DOF of a rule is greater than zero, then that rule is triggered. Since fuzzy rules are not mutually exclusive all triggered rules contribute to the final control action. The consequents of all the rules are combined using the union operation of sets, thus producing a recommendation reflecting the advice of all the rules.

The outcome of this inference process is a fuzzy set which in this form cannot be used for control. The fuzzy set must be reduced to a single point using a process called defuzzification.

Of the several available methods, center of gravity method is the widely adopted one due to the fact that the generated control actions are smoother.

$$a' = \frac{\Sigma\, a_i\, DOF_i}{\Sigma\, DOF_i} \qquad (3.4)$$

where a' is the recommended, defuzzified control action, ai is a point on the linguistic scale of action A, where DOF_i is equivalent to $\mu_A(a)$. The other methods are the original one used by Mamdani and coworkers and its derivatives called the mean of maxima or average of maxima methods[41,42].

IV. DEVELOPMENT OF THE CONTROLLER

4.1 Knowledge Acquisition

The primary task in building a rule-based fuzzy logic controller is acquisition of the knowledge necessary for forming the rules that make up the knowledge base. This is by no means a simple task due to the fact that the set of rules have to be exhaustive, non-redundant, and error and conflict free. Such a rule base, in general can be formed iteratively and at the expense of a number of compromises.

The usual procedure in knowledge acquisition is based on the cooperation of the knowledge engineer with an identified expert in the field of concern. Generally, the expert is requested to supply the rules by describing in as much detail as possible his/her actions and give reasons for them, and the knowledge engineer tries to clarify vague and conflicting issues by analyzing these statements, or the expert's performance is observed while he/she is working on a case with an already known set of outcomes. The key concepts are thus identified, and the rule base is generated accordingly.

During the development of the controller in this work a different approach necessitated by the lack of availability of an appropriate expert was taken. Namely that the regulation function to be performed by this controller is too fast a process to be expected from a human being and current applications rely on known automatic control methods as described elsewhere in this study. Hence the only suitable method for knowledge acquisition was numerical simulation.

The advantage of numerical simulation arises from the fact that it allows the user complete control on the process that is simulated: it can be stopped at any time, every step can be reversed or changed, and every conceivable control action can be tested including those that are impossible for the process and in the case of a NPP would have catastrophic consequences. Nevertheless, the results obtained can be at most as accurate as the numerical model itself which is the method's inherent weakness that should always be kept in mind.

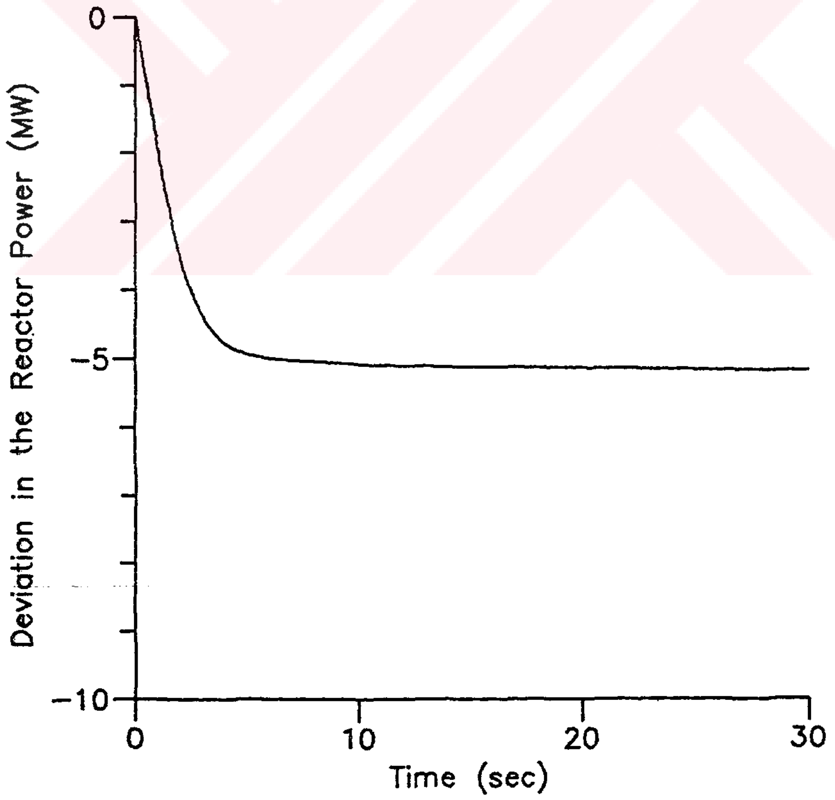

FIGURE 4.1 Response δP to -1 percent step change in steam flow rate

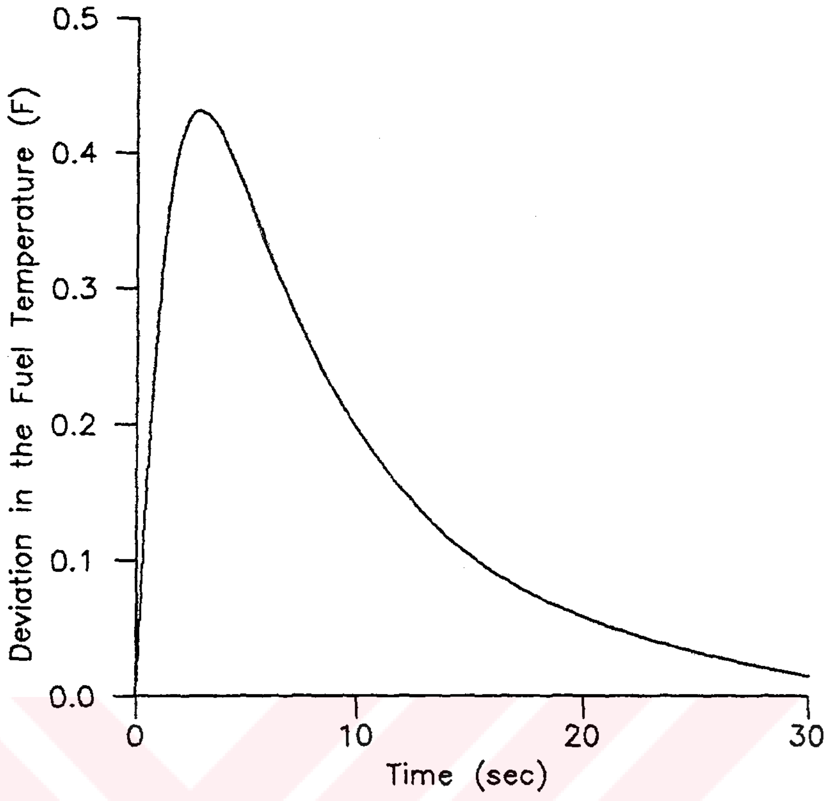

FIGURE 4.2 Response δT_f to -1 percent step change in steam flow rate

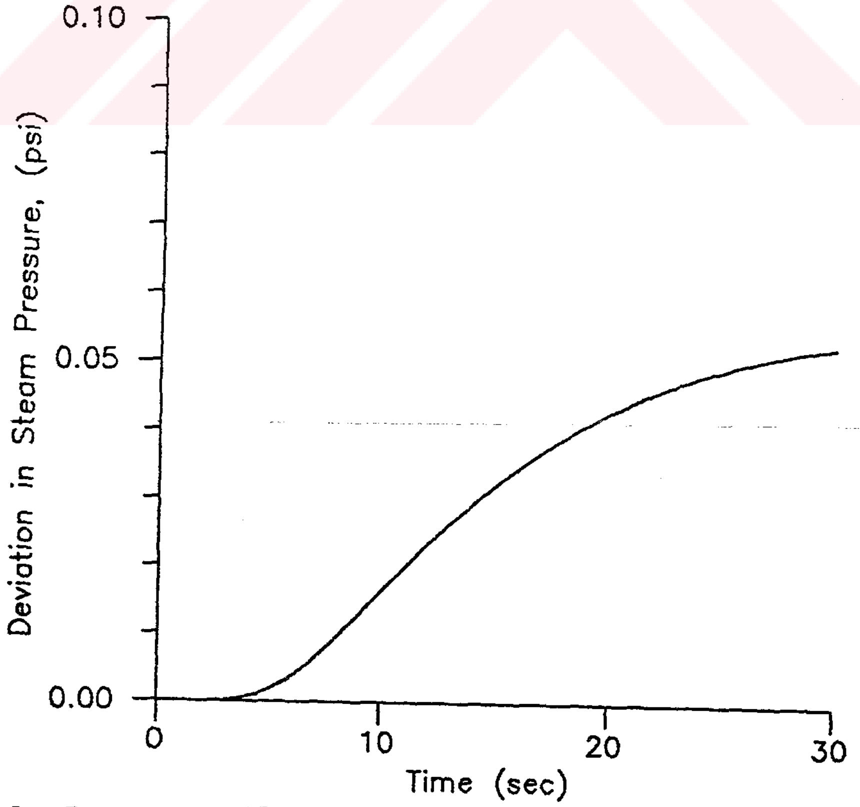

FIGURE 4.3 Response δP_s to -1 percent step change in steam flow rate

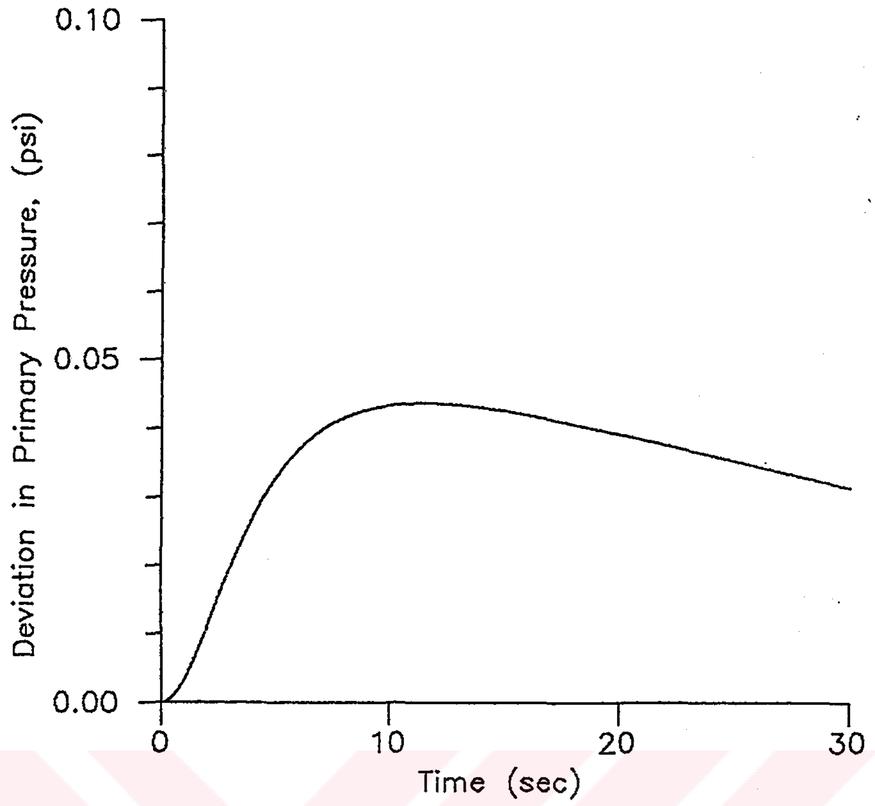

FIGURE 4.4 Response δP_p to -1 percent step change in steam flow rate

The regulator development process started with testing the accuracy of the model and its implementation and comparing the solution method with other solutions[10,12,19]. It was found that within the restrictions allowed for while conceiving the model, the accuracy of the numerical method described in section 2.2 was equivalent to a fourth order Runge-Kutta method[56], Bulirsch and Stoer's method[56], LSTIFF[31] and GENDY[57] and it was considerably faster hence better suited for simulations that are to be run on microcomputers. Later, the response of the NPP to various inputs were investigated some of which are given in Figs.4.1-4.4. These tests were made possible by the interactive software developed specifically for this study that allows the user to stop the simulation and observe each and all of the variables as time domain plots and thus provides the information necessary for forming the rules and making proper selection of the fuzzy variables. The programming language chosen was Turbo Pascal 4.0[58] for ease in programming and ample graphical capabilities which were essential for simulation. The software was implemented on an IBM-compatible Olivetti M 24 PC with a 8 MHz 8086 CPU and 8 MHz 8087 FPU with 640 KB RAM. However, the size of the EXE file is about 80 kB thus simulations can be performed on machines with a smaller RAM capacity.

4.2 Formation of the Knowledge Base

The information about plant behavior thus gathered must be structured for the following reasons: (a) to determine the manipulated and controlled variables; (b) to deduce the linguistic labels that can be used to describe the measured variables and determine the range of each label which will form the basis of fuzzy sets to be used; (c) to determine the rules that will relate the linguistic labels to specific control actions. As was mentioned in chapter 2 there are two variables in the developed mathematical model of H. B. Robinson that can be manipulated to apply the desired control action: rod movement and steam flow rate. However, as the control rods are generally used for power level change, only steam flow was manipulated for power regulation in accordance with previous work[10,12].

The measurable variables that are available to the operator are power(neutron flux), hot and cold leg primary water temperature, primary loop pressure, feedwater flow rate, steam pressure, core outlet temperature, steam generator level, steam flow rate and rod position(19). Of these, power is the variable to be controlled. During the simulation runs it was seen that a sufficiently effective manual control of NPP was possible by only acting on the deviation of power from its steady state value, it was also determined in this process that the rate and direction of the deviation is also considered by the human controller. Based on these observations and simulation runs it was decided to choose power error(PE) corresponding to δP in the model and change in power error(CPE) which is actually the time derivative of PE as the linguistic variables. This choice also follows from the decision to make the rules as simple as possible both for purposes of implementation and development and hence more suitable for practical applications.

The allowable range for power variation that can be the subject of regulation was chosen to be +/- 6 percent since the steady state operation point is full power with only 8 percent overshoot allowed by the safety regulations. The 2 percent margin was arbitrarily determined upon in order not to challenge the safety system. In order to decrease chattering a +/- 0.25 percent band was taken as the deadband where no control action would be applied. The ranges thus selected were spanned with a set of eight fuzzy sets each comprising the range of values of the linguistic variable PE, and another two set of seven fuzzy sets for CPE and change in steam flow rate(CSFR). The fuzzy sets are "negative large", "negative medium", "negative small", "negative zero", "zero", "positive zero", "positive large", "positive medium", and "positive small" (see Fig. 4.5 and Tables 4.1 and 4.2). The shapes of these fuzzy sets and their parameters were adjusted through the process of calibrating the controller.

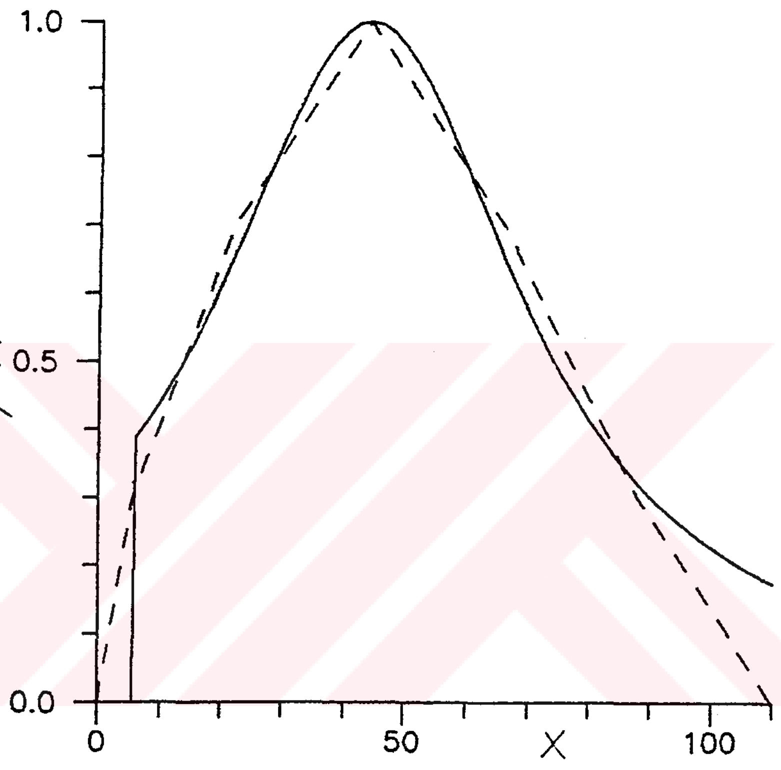

FIGURE 4.5 Fuzzy Sets for "negative small".

The conditional rules that were somehow subconsciously applied by the author during the manual control trials were written down and investigated as to their physical correctness and the rule base was formed after an iterative process in which new rules were added, previous ones deleted or changes were made in the existing ones. This process was by no means a trivial task since the manipulated variable is one of the secondary loop while the controlled variable is that of the primary one. Due to the heat capacity of the steam generators, any action in one of the loops is felt in the other after a considerable time delay thus hindering the credit assignment to the applied rules. It is not easy to determine the effectiveness of the rules that were used consecutively. After long and tedious simulation runs the following twenty two rules were identified. It must be added that making analogies with previous fuzzy controller applications sometimes gives a good set of rules to start with as was the case. All of the following rules are based on the inherent negative temperature feedback in PWR type nuclear reactors: an increase in load decreases the temperature in the primary loop which increases power and each rule is an expert attempt at reverting the adverse developments back to the operating level at an appropriate level.

If power error is negative big or negative medium and
change in power error is negative small
then change in steam flow rate is positive medium.

If power error is negative small and
change in power error is positive small
then change in steam flow rate is positive medium.

If power error is negative zero and
change in power error is positive big or positive medium
then change in steam flow rate is positive medium.

If power error is negative zero and
change in power error is negative big or negative medium
then change in steam flow rate is negative medium.

If power error is negative zero or positive zero, and
change in power error is zero
then change in steam flow rate is zero.

If power error is positive zero, and
change in power error is negative big or negative medium
then change in steam flow rate is positive medium.

If power error is positive zero, and
change in power error is positive big or positive medium
then change in steam flow rate is negative medium.

If power error is positive small, and
change in power error is positive small or zero
then change in steam flow rate is negative medium.

If power error is positive big or positive medium, and
change in power error is negative small
then change in steam flow rate is negative medium.

If power error is negative small, and
change in power error is zero
then change in steam flow rate is positive medium.

If power error is positive medium or positive big, and
change in power error is zero or positive small
then change in steam flow rate is negative big.

If power error is negative big or negative medium, and
change in power error is zero or positive small
then change in steam flow rate is positive big.

If power error is negative big or negative medium, and
change in power error is positive big
then change in steam flow rate is positive big.

If power error is positive big or positive medium, and
change in power error is positive big
then change in steam flow rate is negative big.

If power error is negative small, and
change in power error is positive big or positive medium
then change in steam flow rate is positive big.

If power error is positive small, and
change in power error is positive medium or positive big
then change in steam flow rate is negative big.

If power error is positive small, and
change in power error is positive medium or positive big
then change in steam flow rate is negative big.

If power error is negative small, and
change in power error is negative big or negative medium
then change in steam flow rate is negative small.

If power error is positive small, and
change in power error is negative big or negative medium
then change in steam flow rate is positive small.

If power error is negative small, and
change in power error is negative small
then change in steam flow rate is zero.

If power error is negative zero, and
   change in power error is negative small
then change in steam flow rate is negative small.

If power error is positive zero, and
   change in power error is negative small
then change in steam flow rate is positive small.

If power error is positive small, and
   change in power error is negative small
then change in steam flow rate is zero.

The most important factor that must be kept in mind is the completeness of the rule base. This arises from the limited nature of the allowable and conceivable test cases. Unanticipated transients can be overlooked and it is not possible to guarantee the exhaustiveness of the rules.

4.3 Knowledge Representation

The rules thus selected must be expressed in a form suitable for digital processing. Initially Turbo Prolog[59] was chosen as the programming language for knowledge representation since knowledge representation and inferencing can be done separately. But this approach was later abandoned for the following reasons:(a) the required inferencing mechanism is forward chaining while that of Prolog is essentially backward chaining and this would necessitate special programming in violation of the separability of representation and inferencing; (b) the controller tests require numerical simulation, however Prolog is not suited to numerical calculations and thus interfacing to another language such as C is necessary, which is generally an awkward process with many pitfalls and which also means recoding the numerical simulation software which was already coded in TURBO PASCAL 4.0 since it is not possible to interface Turbo Prolog to Turbo Pascal; (c) execution of a compiled Prolog program is essentially slower while regulation requires faster than one second responses and also the number of test cases that must be run which are necessary for forming the rule base dictates a fast response; (d) the necessary forward chaining inferencing is essentially simple because any or all rules may be fired at the same time without resorting to any additional conflict resolution strategy. This approach is in line with the general trend in the industrial AI applications where there is an increasing use of more conventional languages such as C and Pascal[60]. There are also efforts to prepare methods for converting AI programs to C[34].

```
DOF[8]:=f_AND(power_error_positive_small,
                   f_OR(change_in_power_error_positive_small,
                   change_in_power_error_zero));
ACTION[8]:= change_in_steam_flow_rate_negative_medium(DOF[8]);
```

FIGURE 4.6 A sample rule coded in Pascal.

A sample rule which is coded in Pascal is given in Fig.4.6. Here f_AND(.) and f_OR(.) are the fuzzy AND and fuzzy OR operators implemented as functions while the assignment "DOF[]:=" is the antecedent and "ACTION[]:=" is the consequent parts of each rule respectively and "power_error_positive_small" etc. are the fuzzy sets given in functional forms(see Tables 4.1 and 4.2).

TABLE 4.1 Broken-Line Fuzzy Subsets Used in this Study.

Subset	Range	DOF	Subset	Range	DOF
Power Error [MW] :					
positive zero	0.0 ≤ x < 5.5	1.0	negative zero	-5.5 < x ≤ 0	1.0
	5.5 ≤ x < 44	1.1 - x/55		-44 < x ≤ -5.5	1.1 + x/55
	44 ≤ x < 66	0.9 - 3x/220		-66 < x ≤ -44	0.9 + 3x/220
	otherwise	0.0		otherwise	0.0
Change in Power Error(CPE) [MW/s] :					
zero	0.0 ≤ x < 22	1.0 - 3x/220			
	22 ≤ x < 44	1.1 - x/55			
	44 ≤ x < 66	0.9 - 3x/220			
	-22 < x ≤ 0	1.0 + 3x/220			
	-44 < x ≤ -22	1.1 + x/55			
	-66 < x ≤ -44	0.9 + 3x/220			
	otherwise	0.0			
Power Error [MW] and Change in Power Error(CPE) [MW/s] :					
positive small	0.0 ≤ x* < 5.5	3x/55	negative small	-5.5 < x ≤ 0	3x/55
	5.5 ≤ x < 22	1/6 + 4x/165		-22 < x ≤ -5.5	1/6 - 4x/165
	22 ≤ x < 44	0.4 + 3x/220		-44 < x ≤ -22	0.4 - 3x/220
	44 ≤ x < 66	1.6 - 3x/220		-66 < x ≤ -44	1.6 + 3x/220
	66 ≤ x < 88	1.9 - x/55		-88 < x ≤ -66	1.9 + x/55
	88 ≤ x < 110	1.5 - 3x/220		-110< x ≤ -88	1.5 + 3x/220
	otherwise	0.0		otherwise	0.0
positive medium	22 ≤ x < 44	-0.3 + 3x/220	negative medium	-44 < x ≤ -22	-0.3 - 3x/220
	44 ≤ x < 66	-0.5 + x/55		-66 < x ≤ -44	-0.5 - x/55
	66 ≤ x < 88	-0.2 + 3x/220		-88 < x ≤ -66	-0.2 - 3x/220
	88 ≤ x < 110	2.2 - 3x/220		-110 < x ≤ -88	2.2 + 3x/220
	110 ≤ x < 132	2.7 - x/55		-132 < x ≤ -110	2.7 + x/55
	132 ≤ x < 154	2.1 - 3x/220		-154 < x ≤ -132	2.1 + 3x/220
	otherwise	0.0		otherwise	0.0
positive big	66 ≤ x < 88	-0.9 + 3x/220	negative big	-44 < x ≤ -22	-0.3 - 3x/220
	88 ≤ x < 110	-1.3 + x/55		-66 < x ≤ -44	-0.5 - x/55
	110 ≤ x < 132	-0.8 + x/220		-88 < x ≤ -66	-0.2 - 3x/220
	132 ≤ x	1.0		-110 < x ≤ -88	2.2 + 3x/220
	otherwise	0.0		otherwise	0.0
Change in Steam Flow Rate					
positive small	0 ≤ x < 16	x/16	negative small	-16 < x ≤ 0	-x/16
	otherwise	0		otherwise	0
positive medium	16 ≤ x < 32	-1 + x/16	negative medium	-32 < x ≤ -16	-1 - x/16
	otherwise	0		otherwise	0
positive big	32 ≤ x < 48	-2 + x/16	negative big	-48 < x ≤ -32	-2 - x/16
zero	∀x	0			

* x is either PE or CPE.

TABLE 4.2 S-Shaped Fuzzy Sets Used in this Study.

Subset	Range	DOF	Subset	Range	DOF
Power Error [MW] :					
positive zero	$x < 0$	0	negative zero	$0 < x$	0
	$0 \leq x < 5.5$	1		$-5.5 < x \leq 0$	1
	otherwise	$(1+(2/55(x-5.5))^2)^{-1}$		otherwise	$(1+(-2/55(x+5.5))^2)^{-1}$
Change in Power Error(CPE) [MW/s] :					
zero	$\forall x$	$(1+(x/33)^2)^{-1}$			
Power Error [MW] and Change in Power Error(CPE) [MW/s] :					
positive small	$x^* \leq 5.5$	0	negative small	$-5.5 \leq x$	0
	otherwise	$(1+(-4/121(x-44))^2)^{-1}$		otherwise	$(1+(4/121(x+44))^2)^{-1}$
positive medium	$x \leq 5.5$	0	negative medium	$-5.5 \leq x$	0
	otherwise	$(1+(-1/33(x-88))^2)^{-1}$		otherwise	$(1+(1/33(x+88))^2)^{-1}$
positive big	$x \leq 5.5$	0	negative big	$-5.5 \leq x$	0
	$5.5 < x \leq 132$	$(1+(-1/33(x-132))^2)^{-1}$		$-132 \leq x < -5.5$	$(1+(1/33(x-132))^2)^{-1}$
	$132 < x$	1		$x \leq -132$	1

* x is either PE or CPE.

The initial functional representation for fuzzy sets were obtained by dividing the allowable positive and negative ranges into intervals of 2 percent of power where _small, _medium and _large fuzzy sets have maximum membership values at +/- 2, 4 and 6 percent of power and adjustments were made to the membership values of intermediate values. It must be stressed here that the selection of these functions and ranges are highly arbitrary. The final fuzzy sets decided upon after test runs are of the broken-line type.

4.4 Inferencing

The next step is the generation of the controller action which requires the application of an inferencing mechanism to combine the rules. The selected process is as follows: At every sampling interval for which a suboptimal value is determined as explained below δP (PE) and its rate of change(CPE) is calculated which is actually a simulation of the sensor input. Later the degrees of membership of PE and CPE in every labeled group is determined since a calculated value of PE and CPE can be a member of more than one fuzzy set. The degree of fulfillment (DOF) of each and every rule is calculated as in the code piece given in Fig.4.6. Using DOF the ACTION of each rule is calculated from the corresponding fuzzy set for CSFR given in the consequent part of the rule. The final control action is calculated by weighing ACTION of each rule by its DOF as given in Eq.(3.4). For example, let us assume that we measure PE to be 39.5 MW (% 1.8 FP) and calculate CPE to be -36.72 MW/s we proceed as follows, the degree of fulfillment of the fuzzy sets are calculated. PE is therefore classified as "positive zero" to degree 0.382, "positive small" to degree 0.939, and "positive medium" to degree 0.239. CPE is "zero" to degree 0.432, "negative small" to degree 0.901, "negative medium" to degree 0.201. Degrees of membership of all other sets are zero. Using these values all the antecedents of the rules are calculated. For example, rule 6 states that "If power error is positive zero, and change in power error is negative big or negative medium then change in steam flow rate is positive medium." Applying the OR operation to "negative big" and "negative medium" for CPE gives us the union of these sets which is the maximum of the degrees of membership, i.e, 0.201. Since the clauses for PE and CPE are connected by AND, the intersection of these or the minimum of the degrees of membership must be calculated which is 0.201 for 0.382 is greater. Therefore the DOF of rule #6 is 0.201. The recommended action by this rule is in this case CSFR so as to obtain "positive medium" to degree 0.201. Using the inverse function we calculate the change in steam flow rate that satisfies this condition as 19.21 lb/s (8.72 kg/s). Repeating this process for all the rules we find that only rules 5, 6, 8, 9, 11, 18, 21, and 22 have DOFs greater than 0 which are 0.382, 0.201, 0.432, 0.239, 0.239, 0.201, 0.382, and 0.901 respectively. The recommended changes in steam flow rate by these rules are similarly calculated by using the inverse functions of the fuzzy set descriptions for CSFR to be found as 0, 19.21, -22.92, -19.82, -35.82, 3.21, 6.11, and 0 respectively. The net control action is calculated by weighing the action of each rule by its DOF. Thus, using Eq.(5.4),

$$
\begin{aligned}
\text{CFSR}^{*} = & [(0.382)(0.0)+(0.201)(19.21)+(0.432)(-22.92) \\
& +(0.239)(-19.82)+(0.239)(-35.82)+(0.201)(3.21) \\
& +(0.382)(6.11)+(0.901)(0.0)] \\
& \div[0.382+0.201+0.432+0.239+0.239+0.201+0.382+0.901] \\
= & -5.50
\end{aligned}
$$

is obtained. Therefore there has to be a decrease in steam flow rate by 5.50 lb/s (2.50 kg/s). It is evident that this represents a compromise between differing rules. This process is repeated until a given criterion is satisfied which is generally the time at which a given number of consecutive readings of a selected variable is less than a prescribed value. In this work PE was the selected variable.

4.5 Testing of the Controller

As mentioned before the only presently known method to determine and improve the performance of a RBC is testing and simulation. It was decided to compare the performance of RBC with an optimal controller also developed for the H. B. Robinson[12] plant, however in the past RBC's were compared with PID controllers[17,54]. The selected test case is an initial impulse disturbance of $\delta T_{LP}(0)=2$ F (1.1 C) reported in previous work[12] (see Figs. 4.7-4.10) where the 14-variable model is used.

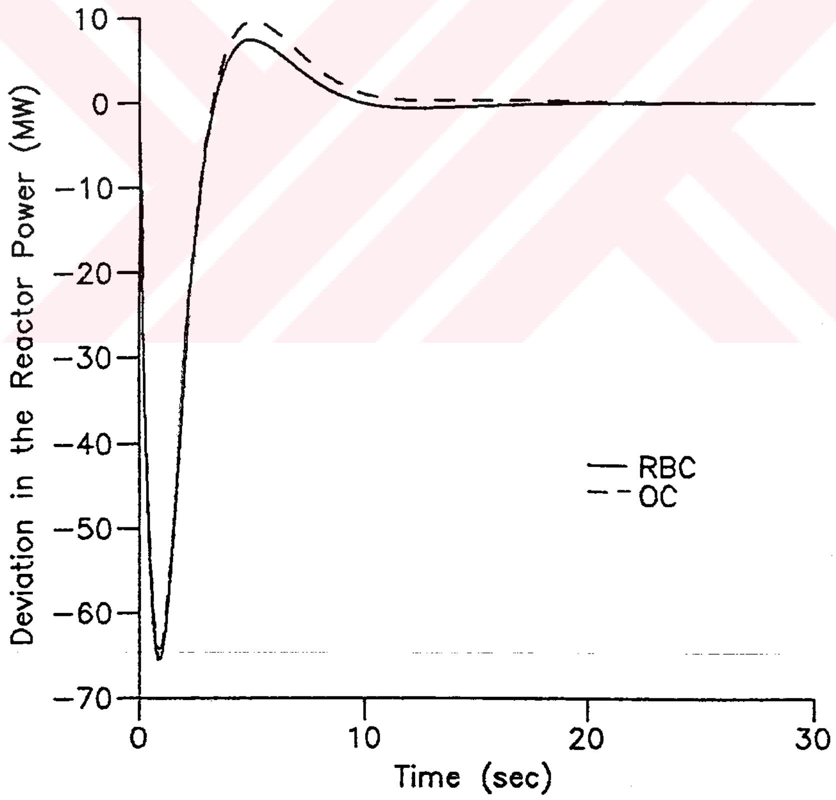

FIGURE 4.7 Comparison of the response δP for a 2 F disturbance in $\delta T_{LP}(0)$ for the optimal controller(OC)[12] and the rule-based controller (RBC).

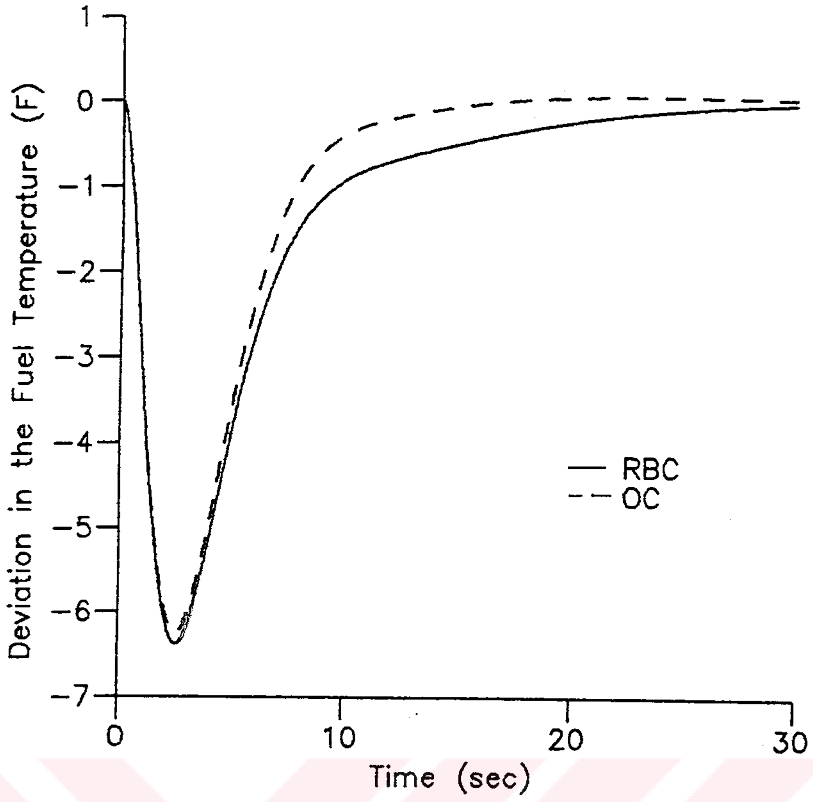

FIGURE 4.8 Comparison of the response δT_f for a 2 F disturbance in $\delta T_{LP}(0)$ for the optimal controller(OC)[12] and the rule-based controller (RBC).

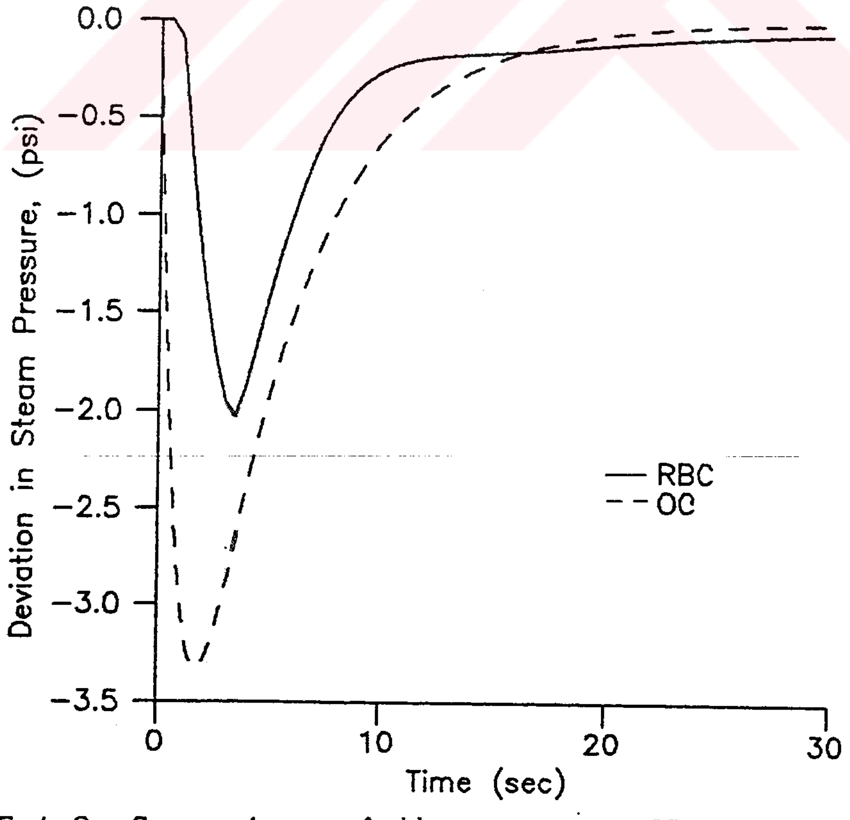

FIGURE 4.9 Comparison of the response δP_s for a 2 F disturbance in $\delta T_{LP}(0)$ for the optimal controller(OC)[12] and the rule-based controller (RBC).

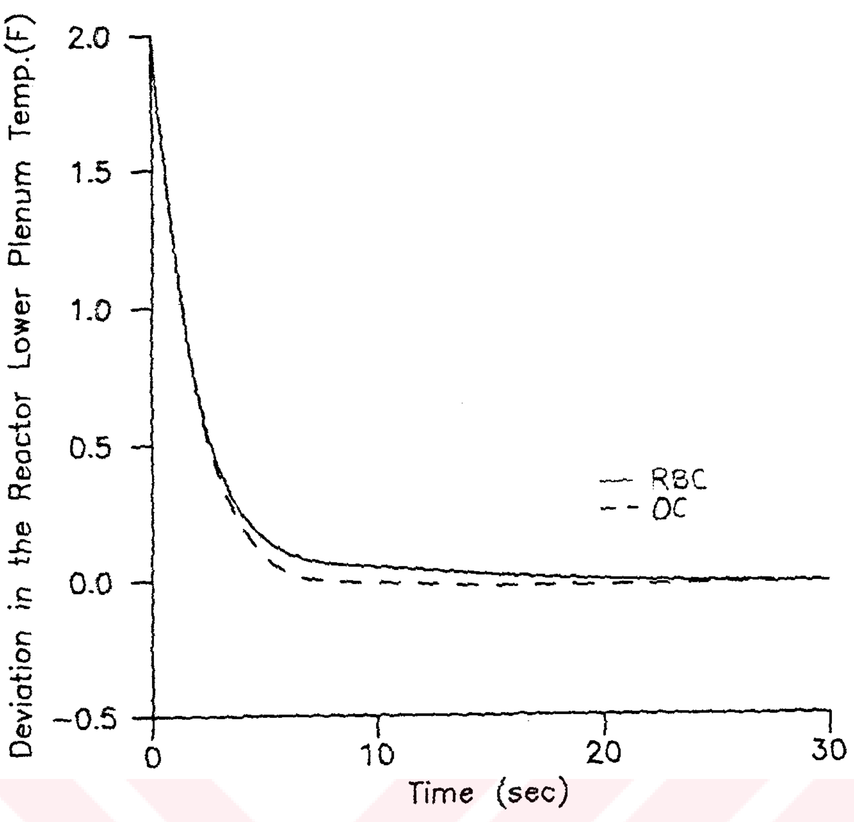

FIGURE 4.10 Comparison of the response δT_{LP} for a 2 F disturbance in $\delta T_{LP}(0)$ for the optimal controller(OC)[12] and the rule-based controller (RBC).

4.5.1. Performance Index

The criterion that is to be used for comparing the performance of both the optimal and RBC and the various implementations of RBC is selected as ITAE(integral of time multiplied by absolute error) since the initial large error is not heavily weighted, whereas the errors that persist are more heavily weighted[18]:

$$PI = \int_0^{\infty} t.|PE|\, dt \qquad (4.1)$$

which is numerically approximated in this study as:

$$PI = \sum_j t\, |PE|\, \Delta t \qquad (4.2)$$

Initially, an ISE(integral of square of error) type criterion was considered

$$PI = \int_{0}^{\infty} (PE)^2 \, dt \qquad (4.3)$$

but it was found to be not sufficiently discriminating and this approach was later abandoned.

4.5.2 Determination of the Control Interval

The control interval in general is a compromise among the following rates at which: the control action is calculated, it is expected to be effective, and the measurements can be conducted. Some of these factors are dependent on the implementation and performance of the actual sensors and actuators which are velocity limited. In this study, the ideal case for both is considered, i.e., the responses are instantaneous since this is also the case for the optimal controller. This idealization does not hamper the worth of the RBC to the same extent as it does in the case of the optimal controller whose actual implementation will be radically different from the ideal case where additional equipment such as observers for nonmeasurable states, etc. are necessary.

The results of the simulation for different values of control intervals are shown in Fig.4.11.

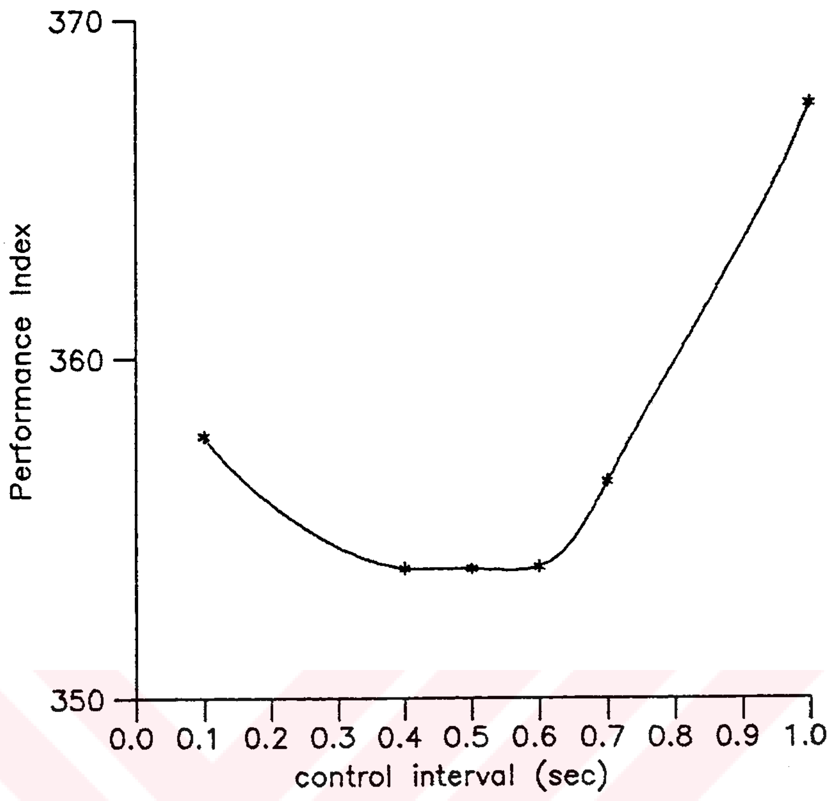

FIGURE 4.11 Variation of PI with control interval.

It is evident from Fig.4.11 that the minimum value of PI lies in the vicinity of t=0.5. This behavior can be explained as follows: for very short control intervals the control action due to time delay has less time to become effective, but as the control interval is increased, again due to time delay, the nature of the transient changes faster than the control action can accommodate. However, it is apparent that there is not a marked degradation in the controller response. Therefore, in implementing a RBC of this type the practical necessities will determine the control interval.

4.5.3 Gain calibration of the controller

The gain calibration is important for the RBC as with all other controllers since it will have a marked effect on the overshoot, oscillation and steady state error characteristics of the controller. The process is as follows. After determining the suboptimal control interval, repeated simulations are performed for different values of control gain and a calibration curve as in Fig.4.12 is obtained, the gain where PI takes its minimum value is selected.

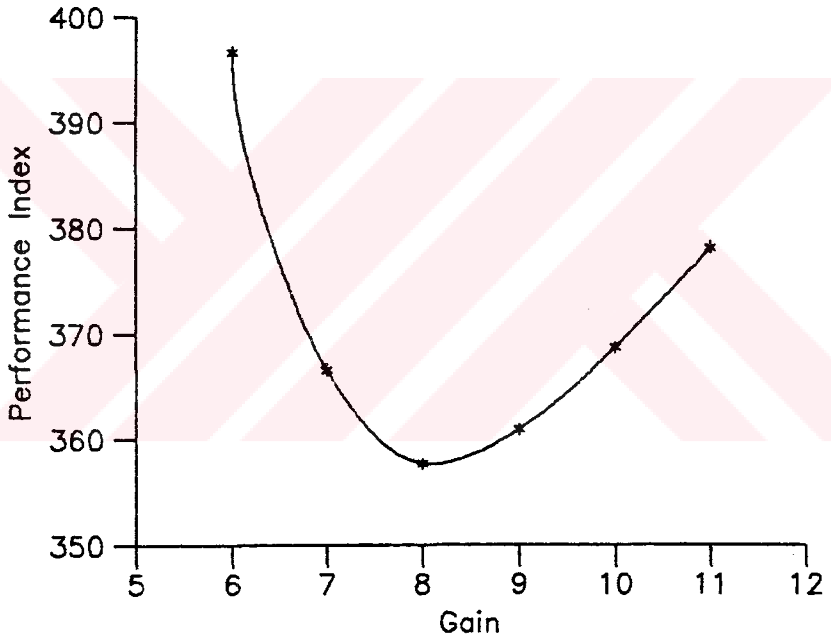

FIGURE 4.12 Controller gain calibration curve.

4.5.4 Determination of the Effect of Measurement Noise

The success of the actual implementation of any controller depends on the dynamical performance of the sensors that supply the necessary feedback information. Unfortunately, most of the time, either the sensors fail or their signal is smeared with noise. The controller is expected to perform even under these degraded conditions or at least degrade gracefully without causing dangerous oscillatory or divergent behavior in the system. In order to investigate the behavior of RBC under noisy conditions the following noise model is used[61]:

$$\delta P_n(t)=\delta P(t)[1 + b.r(t)] \qquad (4.4)$$

where $\delta P_n(t)$ is the variation in power with noise, $\delta P(t)$ the true variation in power, b half-width of noise band, r(t) random number between -1 and 1. The result of the simulations are shown in Fig.4.13 . Apparently, RBC is extremely robust under such conditions, the degradation in its performance is negligible.

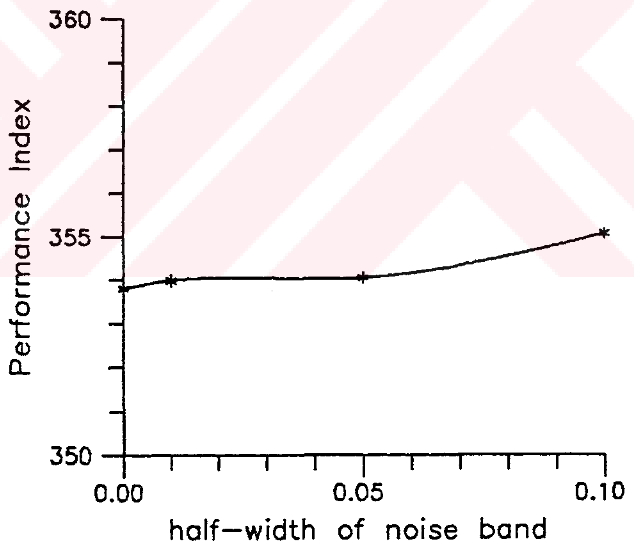

FIGURE 4.13 Effect of noise on controller performance.

4.5.5 Effect of the Variation in Reactor Parameters

Generally it is assumed that the process to be controlled is time invariant. Although this assumption is valid during the interval the controller is effective, this is not the case for the life time of the plant where drifts or changes will occur in the process parameters thus decreasing the validity of the model. This is generally compensated for by tuning the controllers, as this effect becomes noticeable.

In the case of an NPP with a PWR core, with burnup, the moderator coefficient of reactivity becomes more negative primarily as a result of boric acid dilution but also to a significant extent from the effects of the buildup of plutonium and fission products[62]. In this work a moderator temperature coefficient of reactivity parameter variation of 2% from its nominal value is considered[10]. The results of the simulation runs are presented in Figs. 4.14 - 4.17 . The variation in performance is almost unnoticeable as is reflected in the small variation in PI which is 360.52 for the case of off-nominal operation while that of nominal is 357.7.

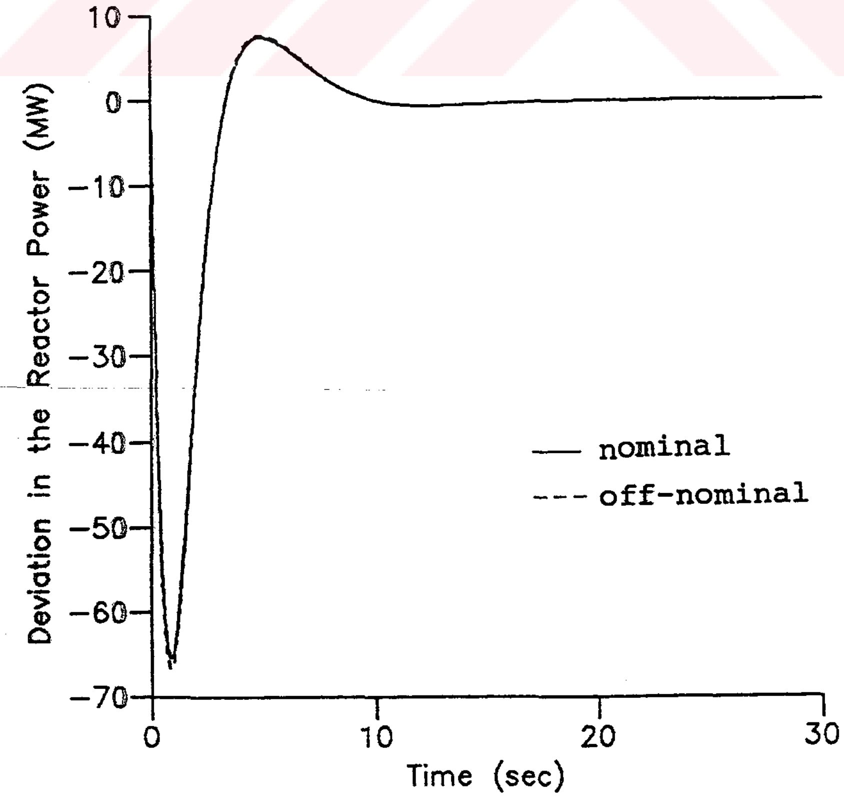

FIGURE 4.14 Response δP for a 2 F disturbance in $\delta T_{LP}(0)$ for the case of 2 percent variation in α_c.

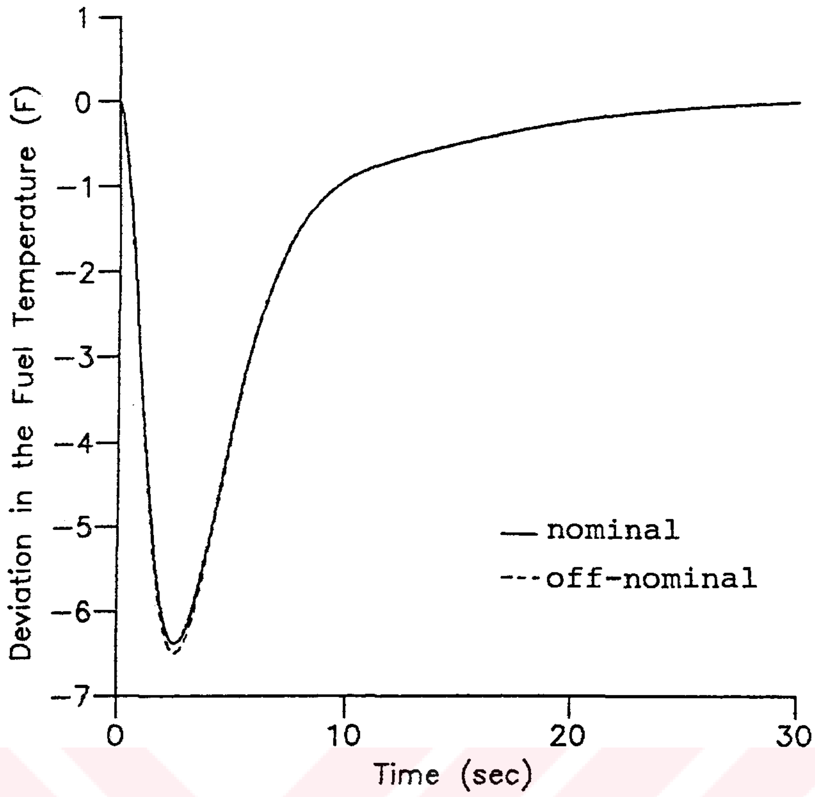

FIGURE 4.15 Response δT_f for a 2 F disturbance in $\delta T_{LP}(0)$ for the case of 2 percent variation in α_c.

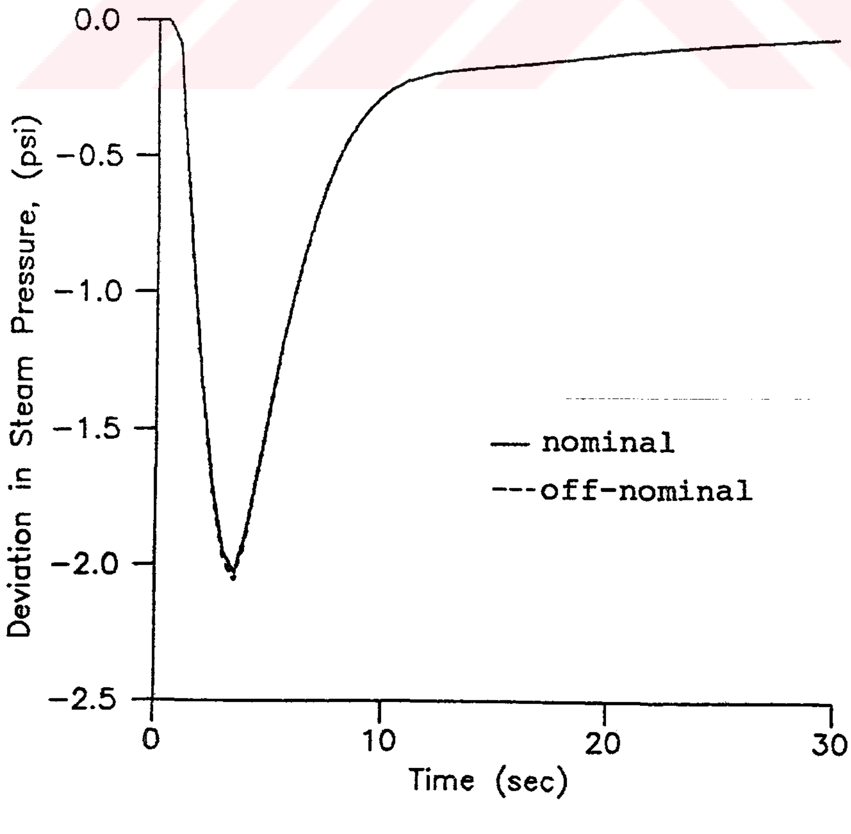

FIGURE 4.16 Response δP_s for a 2 F disturbance in $\delta T_{LP}(0)$ for the case of 2 percent variation in α_c.

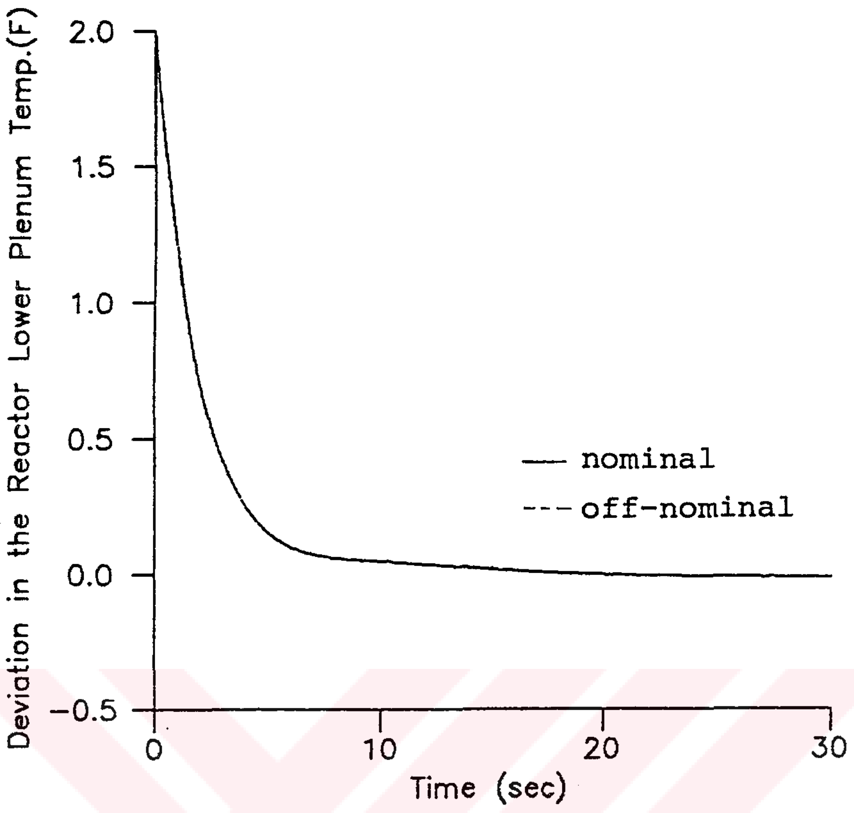

FIGURE 4.17 Response δT_{LP} for a 2 F disturbance in $\delta T_{LP}(0)$ for the case of 2 percent variation in α_c.

V. DISCUSSION AND CONCLUSIONS

Although the analogue controllers in existing NPPs are satisfactory for base load operations, the power plants where digital control is applied, especially CANDU type plants have outstanding operational records. The reliability and strength of digital control techniques become clearer when it is recalled that complicated continuous fuel management schemes are also handled by these techniques in the case of CANDU type reactors. Recently, the nuclear industry has begun to replace analog equipment to benefit from the flexibility and fault tolerance of digital systems in dealing with the problems of equipment obsolescence, low reliability as well as for reductions in scram frequency[63]. However, in most of the cases the initial steps are rather conservative, beginning with an emulation of the original analog control strategies. This reluctance is mainly due to a desire to avoid extensive retraining of operators, and major layout changes.

In any case, the control strategies are designed using the analytical approaches of classical and modern control theory. However, the implementations of the designed controllers have no such mathematically sound basis and is in general governed by heuristics. In the case of complex nonlinear processes where no adequate models are available, the analytical approach fails whereas experienced operators are able to control such processes efficiently. Inspired from this success, knowledge-based controllers for such processes were conceived. This was made possible by the developments in AI such as the knowledge representation techniques and in particular by the ability to represent vagueness in human thinking using fuzzy sets which is essential to this success.

In this work a rule-based fuzzy logic controller was developed for a validated model of PWR type H.B. Robinson nuclear power plant and its performance is compared with that of an analytical controller. The main design criteria were simplicity, ease of implementation and robustness.

Although, the usual approach in developing such a knowledge-based controller starts with knowledge acquisition through interaction with an identified expert in the field, in this work knowledge acquisition was accomplished through numerical simulation, analogy with similar systems and an examination of the mathematical model of the power plant. This somewhat inferior technique is necessitated by the lack of an appropriate human expert because the time scales involved in the implementation of the controller are far beyond the limits of human response based on observation. Such a knowledge acquisition process, though long and tedious with inherent pitfalls, proved to be successful and can be resorted to in similar cases. The knowledge thus acquired is represented as production rules that are expressed in terms of two linguistic variables which are the deviation of power from its steady state value and its time rate of change whose values can be represented by using the fuzzy sets such as "negative small", "positive big", etc.

The structure of the controller is determined by its rule base whereas its performance can be tuned by adjusting the fuzzy sets. There are different functional forms for the fuzzy sets that can be used for this purpose. In this work two of them, the broken-line and the s-shaped fuzzy sets were considered. After tests, the broken-line fuzzy set was decided upon, which can easily be modified and is computationally inexpensive. The persistent residual behavior of the s-shaped fuzzy sets in the neighborhood of 0 degree of membership is one of the fundamental factors that influence the tuning process and is intuitively hard to account for.

Instead of an AI programming language such as PROLOG which was initially considered, the rule-base and the inferencing scheme was coded in PASCAL mainly due to problems in interfacing to simulation software. Although, in this case changes in rules requires some major changes in coding, the final implementation is considerably faster, this is also facilitated by the fact that inferencing is data driven or forward chaining only. This will help in actual implementations, for the control interval in this work is less than a second, and the decision process must take considerably less time.

A performance index was used in order to compare different controllers and the different implementations of rule-based controllers alike. Of the two indices considered, the ITAE(integral of time multiplied with absolute error) type was found to be more discriminating than the

ISE(integral of square of error) type. This is mainly due to the considerable time delay between the manipulated and the controlled variable which reside in the secondary and primary loops respectively. The ISE type punishes the initial errors severely, where the possible latter fluctuations are shadowed by this effect.

The control interval was determined through calibration process at the end of which the value giving the minimum performance index was taken. Time delay was observed to be the main factor influencing the control interval.

It was also possible to tune the gain of the controller by varying the gain, constructing a calibration curve and taking the value corresponding to the minimum value of the performance index. It was noted that this process requires almost no iteration, because interaction of the gain and control interval was weak for the range of values considered.

The performance of the developed controller was compared to that of optimal controller[12] for the case of a $\delta T_{LP}(0)=2$ F. The performance index of the optimal controller is 474.24 while that of rule-based controller turns out to be 353.80. An examination of Fig.4.7 reveals that the overshoot of the rule-based controller is about 2 MW less than that of optimal controller which is a considerable amount. Also from Fig.4.9, the decrease in steam pressure is less. In these respects, it can be concluded that the performance of rule-based controller is better than that of optimal controller. The reason for this is as stated by Kiszka et al[52] : " the linguistic algorithm of control comprises all 'metaphysical' skills of the operator, such as intuition, experience, intelligence and so on, which cannot be dealt with by modern mathematics."

Also investigated were the behavior of the rule-based controller under noisy operation conditions and drift in process variables. Using the noise model of MacDonald et al[61] it was observed that up to 10 percent noise the effect is negligible. For the case of 2 percent variation in moderator temperature reactivity coefficient[10] the degradation is also negligible. Thus it can be concluded that rule-based controller is robust which is an extremely important asset for nuclear power plant operation.

In summary, a robust, simple and easily understandable knowledge-based regulator for a PWR type nuclear power plant has been developed thus showing that (a) simulation can be used for knowledge acquisition; (b) rule-based controllers can succesfully be applied to well characterized processes to compete with analytical controllers.

For further work, besides generalization of the present linear system model to a nonlinear one with the intention of full range control, the development of a more realistic controller whose representation includes the sensor and actuator dynamics can be considered. In order to ease the knowledge acquisition process and to build a more comprehensive rule-base the conception of a controller with a learning capability is a necessity. The next step can be a multivariable controller which will encompass all of the control loops and that will lead the way to fully automated nuclear power plants. Each and all of these possible developments do not seem to be daunting tasks, however the power plant data necessary for implementation is either presently or widely not available in a self consistent form.

APPENDIX

PROGRAM LISTING

```
{$B+}     {Boolean complete evaluation on}
{$S-}     {Stack checking on}
{$I+}     {I/O checking on}
{$N+}     {numeric coprocessor}
{$M 65500,16384,655360}
{$R-}

program KnowledgeBasedPWRController;
{Uses steam flow rate for control, broken line fuzzy sets}
Uses
  Crt,
  Dos,
  Graph,Grafik{Draws the curves};
const
  n=14;
  cont=2;
  ia=10;
  io=23;
  ipmax=500;

type
  real=extended;
  strng8=string[8];
  strng50=string[50];
  arrk=array[1..n] of real;
  arrkk=array[1..n,1..n] of real;

var
  yil,month,gun,hafgun,saat,dakika,saniye,salise:word;
  comment:string[30];
  ch:Char;

okcomment,dtchanged,break,oldu,time,initconds,initplots,simul
:boolean;

lsav,Iprt,Icont,ContStep,step,ip,iy,npi,i,j,npr,npp:integer;

SFRmult,deltaT,ISE,ITAE,period,dt,dtsav,dtprev,t,tmax,prevp:r
eal;
  nam:array[1..n] of strng8;
  name:array[1..ia] of strng8;
  namu:array[1..cont] of strng8;
  title:string[80];
  np:array[1..n] of integer;
  yz:array[1..io] of real;
  pdraw:array[1..ia,1..ipmax] of real;
  fmin,fmax:array[1..ia] of real;
  f,y,y1,yin,integ:arrk;
  expa,intexpa:arrkk;
  u:array[1..cont] of real;
  chosen:array[1..n] of boolean;
  outfil:text;

procedure Reversetext;
begin
  textcolor(0);
  textbackground(15);
```

```
 end;

 procedure NormalText;
 begin
   textcolor(15);
   textbackground(0);
 end;

 procedure reverse(x,y:integer;mess:strng50);
 begin
   gotoxy(x,y);
   reversetext;writeln(mess:30);normaltext;
 end;

 procedure normal(x.y:integer;mess:strng50);
 begin
   gotoxy(x,y);
   normaltext;writeln(mess:30);
 end;

 procedure beeps;
 begin
   sound(500);
   delay(400);
   nosound;
 end;

 procedure InitNames;
 begin
 {***** system variables *****}
   nam[1]:='   δP(t)  ';
   nam[2]:='   δC  (t) ';
   nam[3]:='   δTf(t) ';
   nam[4]:=' δTc1(t) ';
   nam[5]:=' δTc2(t)';
   nam[6]:='   δTp(t) ';
   nam[7]:='   δTm(t)';
   nam[8]:='   δPs(t)';
   nam[9]:=' δTup(t)';
   nam[10]:=' δThl(t) ';
   nam[11]:=' δTip(t) ';
   nam[12]:=' δTop(t)';
   nam[13]:=' δTcl(t)';
   nam[14]:=' δTlp(t) ';
 {**** control variables *****}
   namu[1]:=' Vrod(t)';
   namu[2]:=' δWso(t)';
 end;

 Function GetKey(var FunctionKey: Boolean): char;
 var ch: char;
 BEGIN
   ch:=readkey;
   If (Ch = #0) Then  { it must be a function key }
   BEGIN
     ch:=readkey;
     FunctionKey := true;
   END
   else FunctionKey := false;
   GetKey := Ch;
END;

procedure InitFile;
begin
end;

procedure Initcond;
const
    maxlin=cont;
var
 ch:char;
 i,line,prevline:integer;

procedure reverse8(x,y:integer;mess:strng8;value:real);
begin
  gotoxy(x,y);
  reversetext;writeln(mess:8,'= ',value);normaltext;
end;

procedure normal8(x,y:integer;mess:strng8;value:real);
begin
  gotoxy(x,y);
  normaltext;writeln(mess:8,' = ',value);
end;


begin
  line:=1;prevline:=maxlin;
  reverse(1,1,'     CONTROL VARIABLES INITIALIZATION MENU  ');
  reverse8(1,3,namu[1],u[1]);
  for i:=2 to maxlin do
    normal8(1,2*i+1,namu[i],u[i]);
  repeat
    repeat
      ch:=readkey;
      if (ch <> #27) and (ch <> #13) and (ch =#0) then
         ch:=readkey;
    until ch in [#13,#27,#72,#80];
    prevline:=line;
    case ch of
      #13:begin
            gotoxy(1,15);
            writeln('Please     enter     new     value     for
',namu[line]);
            readln(u[line]);
            reverse8(1,2*line+1,namu[line],u[line]);
            gotoxy(1,15);clreol;
            gotoxy(1,16);clreol;
          end;
      #72: begin
             line:=line-1;
             if line < 1 then
               line:=maxlin;
```

```
normal8(1,2*prevline+1,namu[prevline],u[prevline]);
                reverse8(1,2*line+1,namu[line],u[line]);
             end;
       #80: begin
                line:=line+1;
                if line > maxlin then
                  line:=1;

normal8(1,2*prevline+1,namu[prevline],u[prevline]);
                reverse8(1,2*line+1,namu[line],u[line]);
             end;
     end;
  until (ch=#27);
  ClrScr;
  initconds:=true;
end;

procedure Control;
var
  i:integer;

procedure FuzzyController;
const
  numrules=22;
var
  i:integer;
  DOF,ACTION:array[1..numrules] of real;
  FE,CFE,sumACDOF,sumDOF,
  power_error_positive_big,power_error_positive_medium,
  power_error_negative_zero,power_error_positive_zero,
  power_error_negative_small,power_error_negative_medium,
  power_error_negative_big,power_error_positive_small,

change_in_power_error_negative_small,change_in_power_error_po
sitive_small,
  change_in_power_error_positive_big,

change_in_power_error_negative_big,change_in_power_error_posi
tive_medium,

change_in_power_error_negative_medium,change_in_power_error_z
ero:real;

function f_AND(r1,r2:real):real;
begin
  if r1 < r2 then
    f_AND:=r1
  else
    f_AND:=r2
end;

function f_OR(r1,r2:real):real;
begin
  if r1 > r2 then
    f_OR:=r1
  else
    f_OR:=r2
```

```
end;

procedure CalculateFuzzySets;

function pz(PE:real):real:
begin
  if (PE < 0.0) or (PE >= 66.0) then
    pz:=0.0
  else
    if (PE >= 0.0) and (PE < 5.5) then
      pz:=1.0
    else
      if (PE >= 5.5) and (PE < 44.0) then
         pz:=1.1-PE/55.0
      else
         if (PE >= 44.0) and (PE < 66.0) then
            pz:=0.9-3.0*PE/220.0;
end;


function ps(PE:real):real;
begin
  if (PE < 0.0) or (PE >= 110.0) then
    ps:=0.0
  else
    if (PE >= 0.0) and (PE < 5.5) then
      ps:=3*PE/55.0
    else
      if (PE >= 5.5) and (PE < 22.0) then
         ps:=1.0/6.0+4.0*PE/165.0
      else
         if (PE >= 22.0) and (PE < 44.0) then
           ps:=0.4+3.0*PE/220.0
         else
           if (PE >= 44.0) and (PE < 66.0) then
             ps:=1.6-3.0*PE/220.0
           else
             if (PE >= 66.0) and (PE < 88.0) then
               ps:=1.9-PE/55.0
             else
               if (PE >= 88.0) and (PE <110.0) then
                 ps:=1.5-3.0*PE/220.0;
end;

function pm(PE:real):real;
begin
  if (PE < 22.0) or (PE >= 154.0) then
    pm:=0.0
  else
    if (PE >= 22.0) and (PE < 44.0) then
      pm:=-0.3+3.0*PE/220.0
    else
      if (PE >= 44.0) and (PE < 66.0) then
         pm:=-0.5+PE/55.0
      else
         if (PE >= 66.0) and (PE < 88.0) then
           pm:=-0.2+3.0*PE/220.0
```

```
            else
              if (PE >= 88.0) and (PE < 110.0) then
                pm:=2.2-3.0*PE/220.0
              else
                if (PE >= 110.0) and (PE < 132.0) then
                  pm:=2.7-PE/55.0
                else
                  if (PE >= 132.0) and (PE <154.0) then
                    pm:=2.1-3.0*PE/220.0;
end;


function pb(PE:real):real;
begin
  if (PE < 66.0) then
    pb:=0.0
  else
    if (PE >= 66.0) and (PE < 88.0) then
      pb:=-0.9+3.0*PE/220.0
    else
      if (PE >= 88.0) and (PE < 110.0) then
         pb:=-1.3+PE/55.0
      else
         if (PE >= 110.0) and (PE < 132.0) then
           pb:=-0.8+3.0*PE/220.0
         else
           if (PE >= 132.0) then
             pb:=1.0;
end;

function nz(PE:real):real;
begin
  if (PE > 0.0) or (PE <= -66.0) then
    nz:=0.0
  else
    if (PE <= 0.0) and (PE > -5.5) then
      nz:=1.0
    else
      if (PE <= -5.5) and (PE > -44.0) then
         nz:=1.1+PE/55.0
      else
         if (PE <= -44.0) and (PE > -66.0) then
            nz:=0.9+3.0*PE/220.0;
end;

function ns(PE:real):real;
begin
  if (PE > 0.0) or (PE <= -110.0) then
    ns:=0.0
  else
    if (PE <= 0.0) and (PE > -5.5) then
      ns:=-3*FE/55.0
    else
      if (PE <= -5.5) and (PE > -22.0) then
         ns:=1.0/6.0-4.0*PE/165.0
      else
         if (PE <= -22.0) and (PE > -44.0) then
```

```
                ns:=0.4-3.0*PE/220.0
              else
                if (PE <= -44.0) and (PE > -66.0) then
                  ns:=1.6+3.0*PE/220.0
                else
                  if (PE <= -66.0) and (PE > -88.0) then
                    ns:=1.9+PE/55.0
                  else
                    if (PE <= -88.0) and (PE > -110.0) then
                      ns:=1.5+3.0*PE/220.0;
end;

function nm(PE:real):real;
begin
  if (PE > -22.0) or (PE <= -154.0) then
    nm:=0.0
  else
    if (PE <= -22.0) and (PE > -44.0) then
      nm:=-0.3-3.0*PE/220.0
    else
      if (PE <= -44.0) and (PE > -66.0) then
         nm:=-0.5-PE/55.0
      else
         if (PE <= -66.0) and (PE > -88.0) then
           nm:=-0.2-3.0*PE/220.0
         else
           if (PE <= -88.0) and (PE > -110.0) then
             nm:=2.2+3.0*PE/220.0
           else
             if (PE <= -110.0) and (PE > -132.0) then
               nm:=2.7+PE/55.0
             else
               if (PE <= -132.0) and (PE > -154.0) then
                 nm:=2.1+3.0*PE/220.0;
end;


function nb(PE:real):real;
begin
  if (PE > -66.0) then
    nb:=0.0
  else
    if (PE <= -66.0) and (PE > -88.0) then
      nb:=-0.9-3.0*PE/220.0
    else
      if (PE <= -88.0) and (PE > -110.0) then
         nb:=-1.3-PE/55.0
      else
         if (PE <= -110.0) and (PE > -132.0) then
           nb:=-0.8-3.0*PE/220.0
         else
           if (PE <= -132.0) then
             nb:=1.0;
end;

function cpez:real;
begin
```

```
   if (CPE <= -66.0) or (CPE >= 66.0) then
     cpez:=0.0
   else
     if (CPE >= 0.0) and (CPE < 22.0) then
       cpez:=1.0-3.0*CPE/220.0
     else
       if (CPE >= 22.0) and (CPE < 44.0) then
          cpez:=1.1-CPE/55.0
       else
          if (CPE >= 44.0) and (CPE < 66.0) then
             cpez:=0.9-3.0*PE/220.0
          else
             if (CPE <= 0.0) and (CPE > -22.0) then
                cpez:=1.0+3.0*CPE/220.0
             else
               if (CPE <= -22.0) and (CPE > -44.0) then
                  cpez:=1.1+CPE/55.0
               else
                  if (CPE <= -44.0) and (CPE > -66.0) then
                     cpez:=0.9+3.0*PE/220.0;
end;

begin
  PE:=y[1];
  CPE:=(y[1]-prevp)/dt;
  power_error_positive_big:=pb(PE);
  power_error_positive_medium:=pm(PE);
  power_error_negative_zero:=nz(PE);
  power_error_positive_zero:=pz(PE);
  power_error_negative_small:=ns(PE);
  power_error_negative_medium:=nm(PE);
  power_error_negative_big:=nb(PE);
  power_error_positive_small:=ps(PE);
  change_in_power_error_negative_small:=ns(CPE);
  change_in_power_error_positive_small:=ps(CPE);
  change_in_power_error_positive_big:=pb(CPE);
  change_in_power_error_negative_big:=nb(CPE);
  change_in_power_error_positive_medium:=pm(CPE);
  change_in_power_error_negative_medium:=nm(CPE);
  change_in_power_error_zero:=cpez;

end;

function  steam_flow_rate_negative_big(DOF:real):real;
begin
  steam_flow_rate_negative_big:=SFRmult*(-4-2*DOF)
end;

function  steam_flow_rate_negative_medium(DOF:real):real;
begin
  steam_flow_rate_negative_medium:=SFRmult*(-2-2*DOF)
end;

function  steam_flow_rate_negative_small(DOF:real):real;
begin
  steam_flow_rate_negative_small:=SFRmult*(-2*DOF)
end;
```

```
function  steam_flow_rate_zero:real;
begin
  steam_flow_rate_zero:=0.0;
end;

function  steam_flow_rate_positive_small(DOF:real):real;
begin
  steam_flow_rate_positive_small:=SFRmult*(2*DOF)
end;

function steam_flow_rate_positive_medium(DOF:real):real;
begin
  steam_flow_rate_positive_medium:=SFRmult*(2+2*DOF)
end;

function  steam_flow_rate_positive_big(DOF:real):real;
begin
  steam_flow_rate_positive_big:=SFRmult*(4+2*DOF)
end;

begin
  gotoxy(1,1);write('C');
  CalCulateFuzzySets;
DOF[1]:=f_AND(f_OR(power_error_negative_big,power_error_negat
ive_medium),
             change_in_power_error_negative_small);
ACTION[1]:= steam_flow_rate_positive_medium(DOF[1]);

DOF[2]:=f_AND(power_error_negative_small,change_in_power_erro
r_positive_small);
ACTION[2]:= steam_flow_rate_positive_medium(DOF[2]);

DOF[3]:=f_AND(power_error_negative_zero,
              f_OR(change_in_power_error_positive_big,
                   change_in_power_error_positive_medium));
ACTION[3]:= steam_flow_rate_positive_medium(DOF[3]);

DOF[4]:=f_AND(power_error_negative_zero,
              f_OR(change_in_power_error_negative_big,
                   change_in_power_error_negative_medium));
ACTION[4]:= steam_flow_rate_negative_medium(DOF[4]);

DOF[5]:=f_AND(f_OR(power_error_negative_zero,power_error_posi
tive_zero),
              change_in_power_error_zero);
ACTION[5]:= steam_flow_rate_zero;

DOF[6]:=f_AND(power_error_positive_zero,
              f_OR(change_in_power_error_negative_big,
                   change_in_power_error_negative_medium));
ACTION[6]:= steam_flow_rate_positive_medium(DOF[6]);

DOF[7]:=f_AND(power_error_positive_zero,
              f_OR(change_in_power_error_positive_big,
                   change_in_power_error_positive_medium));
ACTION[7]:= steam_flow_rate_negative_medium(DOF[7]);
```

```
DOF[8]:=f_AND(power_error_positive_small,
              f_OR(change_in_power_error_positive_small,
                   change_in_power_error_zero));
ACTION[8]:= steam_flow_rate_negative_medium(DOF[8]);

DOF[9]:=f_AND(f_OR(power_error_positive_big,power_error_posit
ive_medium),
              change_in_power_error_negative_small);
ACTION[9]:= steam_flow_rate_negative_medium(DOF[9]);

DOF[10]:=f_AND(power_error_negative_small,change_in_power_err
or_zero);
ACTION[10]:= steam_flow_rate_positive_medium(DOF[10]);

DOF[11]:=f_AND(f_OR(power_error_positive_medium,power_error_p
ositive_big),
               f_OR(change_in_power_error_zero,
                    change_in_power_error_positive_small));
ACTION[11]:= steam_flow_rate_negative_big(DOF[11]);

DOF[12]:=f_AND(f_OR(power_error_negative_big,power_error_nega
tive_medium),
               f_OR(change_in_power_error_zero,
                    change_in_power_error_positive_small));
ACTION[12]:= steam_flow_rate_positive_big(DOF[12]);

DOF[13]:=f_AND(f_OR(power_error_negative_big,power_error_nega
tive_medium),
               change_in_power_error_positive_big);
ACTION[13]:= steam_flow_rate_positive_big(DOF[13]);

DOF[14]:=f_AND(f_OR(power_error_positive_big,power_error_posi
tive_medium),
               change_in_power_error_positive_big);
ACTION[14]:= steam_flow_rate_negative_big(DOF[14]);

DOF[15]:=f_AND(power_error_negative_small,
               f_OR(change_in_power_error_positive_big,
                    change_in_power_error_positive_medium));
ACTION[15]:= steam_flow_rate_positive_big(DOF[15]);

DOF[16]:=f_AND(power_error_positive_small,
               f_OR(change_in_power_error_positive_medium,
                    change_in_power_error_positive_big));
ACTION[16]:= steam_flow_rate_negative_big(DOF[16]);

DOF[17]:=f_AND(power_error_negative_small,
             f_OR(change_in_power_error_negative_big,
                  change_in_power_error_negative_medium));
ACTION[17]:= steam_flow_rate_negative_small(DOF[17]);

DOF[18]:=f_AND(power_error_positive_small,
               f_OR(change_in_power_error_negative_big,
                    change_in_power_error_negative_medium));
ACTION[18]:= steam_flow_rate_positive_small(DOF[18]);
```

```
DOF[19]:=f_AND(power_error_negative_small,change_in_power_err
or_negative_small);
ACTION[19]:= steam_flow_rate_zero;

DOF[20]:=f_AND(power_error_negative_zero,change_in_power_erro
r_negative_small);
ACTION[20]:= steam_flow_rate_negative_small(DOF[20]);

DOF[21]:=f_AND(power_error_positive_zero,change_in_power_erro
r_negative_small);
ACTION[21]:= steam_flow_rate_positive_small(DOF[21]);

DOF[22]:=f_AND(power_error_positive_small,change_in_power_err
or_negative_small);
ACTION[22]:= steam_flow_rate_zero;

sumDOF:=0.0;
sumACDOF:=0.0;
for i:=1 to numrules do
begin
  sumDOF:=sumDOF+DOF[i];
  sumACDOF:=sumACDOF+ACTION[i]*DOF[i];
end;
u[2]:=sumACDOF/sumDOF;
  gotoxy(1,1);write('  ');
end;

begin
  Icont:=Icont+1;
  if Icont = ContStep then
  begin
    FuzzyController;
    Icont:=0;
  end;
  f[8]:=-0.04425*u[2];

  for i:=1 to n do
  begin
    integ[i]:=0.0;
    for j:=1 to n do
    begin
      integ[i]:=integ[i]+intexpa[i,j]*f[j];
    end;
  end;
end;

procedure MatrixExponential;
label ExitLoop;
var
   i,j,k:integer;
   pmax,fac,fac2,top:real;
   a,a2,a3:arrkk;
   turn:boolean;

procedure Coefficients;
var
  i,j:integer;
```

```
begin
  for I:=1 to N do
  for J:=1 to N do
  begin
    a[I,J]:=0.0;
    expa[i,j]:=0.0;
    intexpa[i,j]:=0.0;
  end;
  for i:=1 to n do
  begin
    expa[i,i]:=1.0;
    intexpa[i,i]:=dt;
  end;
  a[1,1]:=-400.0;
  a[1,2]:=0.076884;
  a[1,3]:=-1781.0;
  a[1,4]:=-15070.0;
  a[1,5]:=-15070.0;

  a[2,1]:=400.0;
  a[2,2]:=-0.076884;

  a[3,1]:=0.0756;
  a[3,3]:=-0.16466;
  a[3,4]:=0.16466;

  a[4,3]:=0.05707;
  a[4,4]:=-2.4403;
  a[4,14]:=2.3832;

  a[5,3]:=0.05707;
  a[5,4]:=2.3262;
  a[5,5]:=-2.3832;

  a[6,6]:=-0.76642;
  a[6,7]:=0.53819;
  a[6,11]:=0.2238;

  a[7,6]:=3.07017;
  a[7,7]:=-5.3657;
  a[7,8]:=0.33272;

  a[8,7]:=1.349;
  a[8,8]:=-0.2034;

  a[9,5]:=0.33645;
  a[9,9]:=-0.33645;

  a[10,9]:=2.5;
  a[10,10]:=-2.5;

  a[11,10]:=1.45;
  a[11,11]:=-1.45;

  a[12,6]:=1.45;
  a[12,12]:=-1.45;
```

```
    a[13,12]:=1.48;
    a[13,13]:=-1.48;

    a[14,13]:=0.516;
    a[14,14]:=-0.516;
  end; {Coefficients}


  procedure matmult(var a,b,c:arrkk);
  var
    i,j,k:integer;
    top:real;
  begin
    for i:=1 to n do
    for j:=1 to n do
    begin
      top:=0.0;
      for k:=1 to n do
        top:=top+a[i,k]*b[k,j];
      c[i,j]:=top;
    end;
  end;

  procedure addit(var ek:arrkk);
  var
    i,j,ii,jj:integer;
    dum:real;
  begin
      pmax:=1.0e-10;
      for i:=1 to n do
      for j:=1 to n do
      begin
        dum:=ek[i,j]*fac;
        expa[i,j]:=expa[i,j]+dum;
        if abs(dum) > pmax then
        begin
    ii:=i;
    jj:=j;
    pmax:=abs(dum);
        end;
        intexpa[i,j]:=intexpa[i,j]+ek[i,j]*fac2;
      end;
  end;

  begin {MatrixExponential}
    dtchanged:=false;
    Coefficients;
    fac:=dt;
    fac2:=dt*dt*0.5;
    addit(a);
    turn:=true;
    matmult(a,a,a2);
    fac:=fac*dt*0.5;
    fac2:=fac2*dt/3.0;
    addit(a2);
    k:=2;
    while (pmax > 1.0e-9) do
```

```
     begin
       k:=k+1;
       fac:=fac*dt/k;
       fac2:=fac2*dt/(k+1);
       if turn then
       begin
         matmult(a2,a,a3);
         addit(a3);
       end
       else
       begin
         matmult(a3,a,a2);
         addit(a2);
       end;
       turn:=not(turn);
     end;

  ExitLoop:
      Control;

  end;

  procedure printout;
  var
     i,j,npi:integer;
  begin
     ip:=ip+1;
     pdraw[1,ip]:=t;
     pdraw[2,ip]:=u[2];
     pdraw[3,ip]:=0.5*(y[10]+y[13]);
     for i:=1 to npr do
     begin
       npi:=np[i];
       pdraw[i+3,ip]:=y[npi];
     end;
     iprt:=0;
  end;

  procedure Extrema;
  var
     i,j,npi:integer;
     r:array[1..ia] of real;
  begin
     r[1]:=t;
     r[2]:=u[2];
     r[3]:=0.5*(y[10]+y[13]);
     for i:=1 to npr do
     begin
       npi:=np[i];
       r[i+3]:=y[npi];
     end;
     for J:=1 to npp do
     begin
       if(r[j] > fmax[j])  then
          fmax[j]:=r[j];
       if(r[j] < fmin[j])  then
           fmin[j]:=r[j];
```

```
   end;
 end;


 procedure Display;
 begin
   gotoxy(10,1);writeln('ISE=  ',ISE:8:4);
   gotoxy(10,40);writeln('ITAE=  ',ITAE:10:4);
   gotoxy(45,1);writeln(' t= ',t:8:4);
   gotoxy(5,3);writeln(nam[1]:8,'  =  ',y[1]:10:3);
   gotoxy(7,5);writeln(nam[13]:8,'  =  ',y[13]:10:3);
   gotoxy(40,5);writeln(nam[10]:8,'  =  ',y[10]:10:3);
   gotoxy(40,11);writeln(nam[8]:8,'  =  ',y[8]:10:3);
   gotoxy(40,3);writeln('Period= :8,  =  ',period:10:3);
 end;

 procedure Simulate;
 var
    i,j:integer;
 begin
   control;
   t:=t+dt;
   for i:=1 to n do
   begin
     y[i]:=0.0;
     for j:=1 to n do
       y[i]:=y[i]+expa[i,j]*y1[j];
     y[i]:=y[i]+integ[i];
   end;
   ISE:=ISE+sqr(y[1])*dt;
   ITAE:=ITAE+t*abs(y[1])*dt;
   if y[1] = y1[1] then
      period:=1.0e+38
   else
      period:=(2178.0+y[1])*dt/(y[1]-y1[1]);
   yz[1]:=t;
   iprt:=iprt+1;
   if (iprt = isav) and (ip < ipmax-1) then printout;
   Display;
   Extrema;
   prevp:=y1[1];
   for i:=1 to n do
     y1[i]:=y[i];
 end;

 procedure SimulatePWR;
 var
   inkey:Char;
 begin
   if (time)  and (initconds) and (initplots)  and (okcomment)
 then
   begin
     gettime(saat,dakika,saniye,salise);
     getDate(yil,month,gun,hafgun);
     gotoxy(14,22);writeln('  SIMULATION  IN  PROGRESS  PLEASE
 WAIT !...');
     Icont:=0;
    ISE:=0.0;
    ITAE:=0.0;
    fmin[1]:=0.0;
    fmax[1]:=0.0;
    for i:=2 to npp do
    begin
      fmin[i]:=1.0e20;
      fmax[i]:=-1.0e20;
    end;
    for i:=1 to n do
    begin
      y1[i]:=yin[i];
      y[i]:=yin[i];
      f[i]:=0.0;
      integ[i]:=0.0;
    end;
    prevp:=y1[1];
    for i:=1 to cont do
      u[i]:=0.0;
    if dtchanged then
       MatrixExponential;
    ip:=0;
    t:=0.0;
    Iprt:=0;
    Extrema;
    printout;
    break:=false;
    repeat
      if KeyPressed then
      begin
        ch:=upcase(readKey);
        case ch of
          #27 : break:=true;
          'C' : begin
                  CloseGraph;
                  InitCond;
                end;
        end;
      end;
      Simulate;
    until (t >=tmax) or (Break);
    readln;
    closegraph;
    clrscr;
    gotoxy(10,14);
    simul:=true;
    if break then
      writeln(' SIMULATION INTERRUPTED')
    else
    begin
      writeln(' SIMULATION COMPLETED !...');
      printout;
      break:=false;
      okcomment:=false;
    end;
    beeps;delay(1000);
end
```

```
    else
    begin
      gotoxy(1,15);textcolor(16);textbackground(31);
      writeln('Please do the initialization first!.. ');
      textcolor(15);textbackground(0);
      beeps;delay(1000);
    end;
  end;


  procedure Initime;
  const
     maxlin=4;
  type
     messarr=array[1..maxlin] of strng50;
  var
   ch:char;
   i,line,prevline:integer;
   value:array[1..maxlin] of real;
   arr:messarr;
   prevdt:real;

  procedure reverse30(x,y:integer;mess:strng50;value:real);
  begin
    gotoxy(x,y);
    reversetext;writeln(mess:30,' = ',value);normaltext;
  end;

  procedure normal30(x,y:integer;mess:strng50;value:real);
  begin
    gotoxy(x,y);
    normaltext;writeln(mess:30,' = ',value);
  end;

  begin
    prevdt:=dt;
    dtsav:=tmax/ipmax;
    value[1]:=dt;
    value[2]:=tmax;
    value[3]:=dtsav;
    value[4]:=deltaT;
    line:=1;prevline:=maxlin;
    arr[1]:=' Step length dt                 ';
    arr[2]:=' Stopping time tmax             ';
    arr[3]:=' Step length for data storage';
    arr[4]:=' Control interval delta T       ';
    clrscr;
    reverse(1,1,' TIME INITIALIZATION MENU');
    reverse30(1,3,arr[1],value[1]);
    for i:=2 to maxlin do
       normal30(1,2*i+1,arr[i],value[i]);
    repeat
      repeat
        ch:=readkey;               {read the keystroke}
        if (ch <> #13) and (ch =#0) then
           ch:=readkey;
      until ch in [#13,#27,#72,#80];
     prevline:=line;
     case ch of
       #13:begin
             gotoxy(1,15);
             writeln('Please enter new value for  ',arr[line]);
             readln(value[line]);
             reverse30(1,2*line+1,arr[line],value[line]);
             gotoxy(1,15);clreol;
             gotoxy(1,16);clreol;
           end;
       #72: begin
              line:=line-1;
              if line < 1 then
                line:=maxlin;

normal30(1,2*prevline+1,arr[prevline],value[prevline]);
              reverse30(1,2*line+1,arr[line],value[line]);
            end;
       #80: begin
              line:=line+1;
              if line > maxlin then
                line:=1;

normal30(1,2*prevline+1,arr[prevline],value[prevline]);
              reverse30(1,2*line+1,arr[line],value[line]);
            end;
     end;
  until (ch=#27);
  dt:=value[1];
  tmax:=value[2];
  dtsav:=value[3];
  deltaT:=value[4];
  if prevdt <> dt then
     dtchanged:=true;
  ContStep:=trunc(deltaT/dt+0.5);
  ISav:=trunc(dtsav/dt+0.5);
  time:=true;
end; {InitTime}



procedure InitSys;
const
   maxlin=7;
var
 ch:char;
 i,col,prevcol,line,prevline:integer;

procedure reverse8(x,y:integer;mess:strng8;value:real);
begin
  gotoxy(x,y);
  reversetext;writeln(mess:8,'= ',value);normaltext;
end;

procedure normal8(x,y:integer;mess:strng8;value:real);
begin
  gotoxy(x,y);
  normaltext;writeln(mess:8,'=  ',value);
end;

procedure checkline;
begin
  if line < 2 then
  begin
     line:=maxlin+1;
     col:=1;
  end
  else
    if line > maxlin+1 then
       line:=2;
{  if line=maxlin then
     col:=1;}
end;

procedure checkcol;
begin
  if line=maxlin then
    col:=1
  else
    if col < 1 then
      col:=1
    else
      if col > 2 then
         col:=2;
end;

function pos(col,line:integer):integer;
begin
  pos:=2*line+col-4;
end;


begin
  line:=2;prevline:=2;col:=1;prevcol:=1;
  clrscr;
  reverse(1,1,'   SYSTEM VARIABLES INITIALIZATION  MENU ');
  reverse8(1,2,nam[1],yin[1]);normal8(40,3,nam[2],yin[2]);
  for i:=2 to maxlin+1 do
    begin
      normal8(1,i,nam[pos(1,i)],yin[pos(1,i)]);
      normal8(40,i,nam[pos(2,i)],yin[pos(2,i)]);
    end;
    {normal8(1,maxlin,nam[n],yin[n]);}
  repeat
    repeat
      ch:=readkey;              {read the keystroke}
      if (ch <> #27) and (ch <> #13) and (ch =#0) then
         ch:=readkey;
    until ch in [#13,#27,#72,#75,#77,#80];
    prevline:=line;
    prevcol:=col;
    case ch of
      #13:begin
            gotoxy(1,20);
```

```
             writeln('Please      enter     new      value     for
',nam[pos(col,line)]);
             readln(yin[pos(col,line)]);

reverse8(39*col-38,line,nam[pos(col,line)],yin[pos(col,line)]
);
             gotoxy(1,20);clreol;
             gotoxy(1,21);clreol;
           end;
      #72: begin      {up arrow}
              line:=line-1;
              checkline;

normal8(39*col-38,prevline,nam[pos(prevcol,prevline)],yin[pos
(prevcol,prevline)]);

reverse8(39*col-38,line,nam[pos(col,line)],yin[pos(col,line)]
);
            end;
      #80: begin      {down arrow}
              line:=line+1;
              checkline;

normal8(39*col-38,prevline,nam[pos(prevcol,prevline)],yin[pos
(prevcol,prevline)]);

reverse8(39*col-38,line,nam[pos(col,line)],yin[pos(col,line)]
);
            end;
      #75,#77: begin     {left arrow}
              col:=3-col;

normal8(39*prevcol-38,prevline,nam[pos(prevcol,prevline)],yin
[pos(prevcol,prevline)]);

reverse8(39*col-38,line,nam[pos(col,line)],yin[pos(col,line)]
);
            end;
     end;
  until (ch=#27);
  ClrScr;
  initconds:=true;
end;

procedure Initplot;
const
   maxlin=7;
var
 ch:char;
 i,col,prevcol,line,prevline:integer;

procedure reverse8(x,y:integer;mess:strng8;mark:boolean);
begin
  gotoxy(x,y);
  reversetext;
  if mark then
     writeln('√',mess:10)
```

```
    else
       writeln(mess:11);
    normaltext;
  end;

  procedure normal8(x,y:integer;mess:strng8;mark:boolean);
  begin
    gotoxy(x,y);
    normaltext;
    if mark then
       writeln('√',mess:10)
    else
       writeln(mess:11);
  end;

  procedure checkline;
  begin
    if line < 2 then
    begin
       line:=maxlin+1;
       col:=1;
    end
    else
      if line > maxlin+1 then
         line:=2;
  {  if line=maxlin then
       col:=1;}
  end;

  procedure checkcol;
  begin
   { if line=maxlin then
       col:=1
    else }
      if col < 1 then
         col:=1
      else
        if col > 2 then
           col:=2;
  end;

  function pos(col,line:integer):integer;
  begin
    pos:=2*line+col-4;
  end;

  procedure select;
  begin
    if not(chosen[pos(col,line)]) then
    begin
      if npr < ia then
      begin
        npr:=npr+1;
        chosen[pos(col,line)]:=true;
      end
    end
    else
   begin
     npr:=npr-1;
     chosen[pos(col,line)]:=false;
   end;
 end; {select}

 begin
   line:=2;prevline:=2;col:=1;prevcol:=1;
   clrscr;
   reverse(1,1,'  PLOT VARIABLES SELECTION MENU');

reverse8(1,2,nam[1],chosen[1]);normal8(40,3,nam[2],chosen[2])
;
   for i:=2 to maxlin+1 do
     begin
       normal8(1,i,nam[pos(1,i)],chosen[pos(1,i)]);
       normal8(40,i,nam[pos(2,i)],chosen[pos(2,i)]);
     end;
     {normal8(1,maxlin,nam[n],chosen[n]);}
   repeat
     repeat
       ch:=readkey;                     {read the keystroke}
       if (ch <> #27) and (ch <> #13) and (ch =#0) then
          ch:=readkey;
     until ch in [#13,#27,#72,#75,#77,#80];
     prevline:=line;
     prevcol:=col;
     case ch of
       #13:begin
             select;

reverse8(39*col-38,line,nam[pos(col,line)],chosen[pos(col,lin
e)]);
            end;
       #72: begin      {up arrow}
              line:=line-1;
              checkline;

normal8(39*col-38,prevline,nam[pos(prevcol,prevline)],chosen[
pos(prevcol,prevline)]);

reverse8(39*col-38,line,nam[pos(col,line)],chosen[pos(col,lin
e)]);
            end;
       #80: begin      {down arrow}
              line:=line+1;
              checkline;

normal8(39*col-38,prevline,nam[pos(prevcol,prevline)],chosen[
pos(prevcol,prevline)]);

reverse8(39*col-38,line,nam[pos(col,line)],chosen[pos(col,lin
e)]);
            end;
       #75,#77: begin     {left arrow}
              col:=3-col;
```

```
normal8(39*prevcol-38,prevline,nam[pos(prevcol,prevline)],cho
sen[pos(prevcol,prevline)]);

reverse8(39*col-38,line,nam[pos(col,line)],chosen[pos(col,lin
e)]);
           end;
    end;
  until (ch=#27);
  ClrSCr;
  npr:=0;
  for i:=1 to n do
    if chosen[i] then
    begin
      npr:=npr+1;
      np[npr]:=i;
    end;
  npp:=npr+3;
  for i:=4 to npp do
    name[i]:=nam[np[i-3]];
    name[1]:='    t     ';
    name[2]:=namu[2];
    name[3]:=' Tav      ';
  initplots:=true;
end;

procedure GetComment;
begin
  Clrscr;
  writeln('Comment:   ',comment);
  readln(comment);
  ClrScr;
  okcomment:=true;
end;

procedure Initialize;
const
   maxlin=6;
type
   messarr=array[1..maxlin] of strng50;
var
 ch:char;
 i,line,prevline:integer;
 arr:messarr;

begin
  line:=1;prevline:=maxlin;
  arr[1]:='     1 : Initialize Time                ';
  arr[2]:='     2 : Initialize Control Variables   ';
  arr[3]:='     3 : Initialize System Variables    ';
  arr[4]:='     4 : Select plot variables          ';
  arr[5]:='     5 : Specify Comment                ';
  arr[6]:='     6 : Exit initialization step       ';
  clrscr;
  reverse(1,1,'          INITIALIZATION MENU            ');
  reverse(1,3,arr[1]);
  for i:=2 to maxlin do
     normal(1,2*i+1,arr[i]);
    repeat
      repeat
        ch:=readkey;                   {read the keystroke}
        if (ch <> #13) and (ch =#0) then
           ch:=readkey;
      until ch in [#13,#72,#80];
      prevline:=line;
      case ch of
        #13:begin
              case line of
                1 : InitTime;
                2 : begin Clrscr;InitCond; end;
                3 : InitSys;
                4 : Initplot;
                5 : GetComment;
              end;
              ClrSCr;
              reverse(1,1,'         INITIALIZATION MENU
');
              for i:=1 to maxlin do
                normal(1,2*i+1,arr[i]);
              reverse(1,2*line+1,arr[line]);
             end;
        #72: begin
               line:=line-1;
               if line < 1 then
                 line:=maxlin;
               normal(1,2*prevline+1,arr[prevline]);
               reverse(1,2*line+1,arr[line]);
             end;
        #80: begin
               line:=line+1;
               if line > maxlin then
                 line:=1;
               normal(1,2*prevline+1,arr[prevline]);
               reverse(1,2*line+1,arr[line]);
             end;
      end;
    until (line=maxlin) and (ch=#13);
    ClrScr;
  end; {Initialize}

  procedure Restart;
  var
    i:integer;
  begin
    if (break) then
    begin
      gotoxy(14,22);writeln(' SIMULATION RESTARTED  PLEASE WAIT
!...');
      for i:=1 to n do
      begin
        y1[i]:=y[i];
      end;
      break:=false;
      repeat
        if KeyPressed then
        begin
          ch:=upcase(readkey);
          case ch of
            #27  : break:=true;
            'C'  : begin
                      CloseGrapn;
                      InitCond;
                    end;
          end;
        end;
        Simulate;
      until (abs(t-tmax) < 1.0e-8) or (Break);
      printout;
      closegraph;
      clrscr;simul:=true;
      gotoxy(10,14);
      if break then
        writeln(' SIMULATION INTERRUPTED')
      else
      begin
        writeln(' SIMULATION COMPLETED !...');
        t:=0.0;
        break:=false;
      end;
      beeps;delay(1000);
    end;
  end;


  procedure Report;
  var
     fileadi:string[14];
     i,j:integer;
  begin
    if simul then
    begin
      Clrscr;
      writeln('Please enter report file name  ');
      readln(fileadi);
      assign(outfil,fileadi);
      {$I-} rewrite(outfil) {$I+};
      writeln(outfil,'1','**** PROGRAM SIMULATE OUTPUT***** ');
      writeln(outfil,'   H.B   ROBINSON  PWR   (REDUCED  MODEL)
  SIMULATION ');
      writeln(outfil,comment);
      writeln(outfil,'Date:  ',gun:2,'/',month:2,'/',yil:4,'
  Time:  ',saat:2,':',dakika:2);
      writeln(outfil);
      writeln(outfil,'   NO OF VARIABLES    ',n:6);
      writeln(outfil,'   DELTA-T       ',dt:12);
      writeln(outfil,'   TOTAL TIME    ',tmax:12);
      writeln(outfil);
      writeln(outfil,'  NO VARIABLE   INIT VALUE  ');
      for     i:=1     to     n     do     writeln(outfil,i:4,'
  ',NAM[i],'=',yin[i]:13);
      writeln(outfil);
      writeln(outfil,'  INITIAL VALUES OF CONTROL VARIABLES');
     for     i:=1     to      cont     do      writeln(outfil,i:4,
 ,namu[i], ' = ',u[i]:13);
     writeln(outfil);
     writeln(outfil,                        , 'MINIMUM',
', 'MAXIMUM');
     for    i:=1    to    npp    do    writeln(outfil,name[i]:6,
 ,fmin[i]:15,'    ',fmax[i]:15);
     writeln(outfil);
     writeln(outfil,'ISE=  ',ISE,'  ITAE=  ',ITAE);
     writeln(outfil);
     for i:=1 to npp do write(outfil,name[i]:10,'    ');
     writeln(outfil);
     for i:=1 to ip do
     begin
       writeln(outfil);
       for j:=1 to  npp do write(outfil,poraw[j,i]:14:7);
     end;
     writeln(outfil);
     close(outfil);
   end
   else
   begin
     gotoxy(1,15);textcolor(16);textbackground(31);
     writeln('Please do the simulation first!..');
     textcolor(15);textbackground(0);
     beeps;delay(1000);
   end;
 end;

 procedure main_menu;
 type
    messarr=array[1..6] of strng50;
 var
  ch:char;
  i,line,prevline:integer;
  arr:messarr;

 begin
   line:=1;prevline:=6;
     arr[1]:='    1 : Initialize          ';
     arr[2]:='    2 : Simulate            ';
     arr[3]:='    3 : plot variables      ';
     arr[4]:='    4 : Restart Simulation';
     arr[5]:='    5 : Report              ';
     arr[6]:='    6 : Exit program        ';
     clrscr;
     reverse(1,1,'         MAIN MENU         ');
     reverse(1,3,arr[1]);
     for i:=2 to 6 do
       normal(1,2*i+1,arr[i]);
 repeat
   repeat
     ch:=readkey;              {read the keystroke}
     if (ch <> #13) and (ch =#0) then
        ch:=readkey;
   until ch in [#13,#72,#80];
   prevline:=line;
  case ch of
      #13:begin
            case line of
              1 : Initialize;
              2 : SimulatePWR;
              3 : Grafik;
              4 : Restart;
              5 : Report;
             end;
             ClrScr;
             reverse(1,1,'          MAIN MENU          ');
             for i:=1 to 6 do
               normal(1,2*i+1,arr[i]);
             reverse(1,2*line+1,arr[line]);
           end;
      #72: begin
             line:=line-1;
             if line < 1 then
               line:=6;
             normal(1,2*prevline+1,arr[prevline]);
             reverse(1,2*line+1,arr[line]);
           end;
      #80: begin
             line:=line+1;
             if line > 6 then
               line:=1;
             normal(1,2*prevline+1,arr[prevline]);
             reverse(1,2*line+1,arr[line]);
           end;
  end;
  until (line=6) and (ch=#13);
end;

begin { main program}
  time:=false;
  okcomment:=false;
  initconds:=false;
  initplots:=false;
  simul:=false;
  dtchanged:=true;
  dtprev:=0.0;
  dt:=0.01;
  tmax:=30.0;
  dtsav:=0.02;
  deltaT:=0.5;
  period:=1.0e+38;
  step:=0;
  ContStep:=trunc(deltaT/dt+0.5);
  Isav:=trunc(dtsav/dt+0.5);
  InitNames;
  InitFile;
  SFRmult:=8.0;
  comment:='';
  for i:=1 to n do
    begin
      y[i]:=0.0;
      yin[i]:=0.0;
      end;
    for i:=1 to cont
      do u[i]:=0.0;
    npr:=0;
    for i:=1 to n do
      chosen[i]:=false;
      Main_menu;
end.
```

REFERENCES

1. Michaelsen, R. H, Michie, D., Boulanger, A., "The Technology of Expert Systems," Byte, pp.303-312, April 1985.

2. Maiers, J., Sherif, Y .S., "Applications of Fuzzy Set Theory," IEEE Transactions on Systems, Man, and Cybernetics, vol.SMC-15, no.1, January/February 1985.

3. Bernard, J. A., "Use of a Rule-Based System for Process Control," IEEE Control Systems Magazine, pp.3-13, October 1988.

4. Frogner, B., Rao, H. S., "Control of Nuclear Power Plants," IEEE Transactions on Automatic Control, Vol. AC-23, No.3, pp.405-417, June 1978.

5. McMorran, P. D., "Multivariable Control in Nuclear Power Stations: Survey of Design Methods," Atomic Energy of Canada Limited, AECL-6583, December 1979.

6. Tsuji, M., Ogawa, Y., "Load Follow-up Control of a Pressurized Water Reactor Power Plant by Using an Approximate Noninteractive Control," IEEE Transactions on Nuclear Science, vol. NS-33, no.4, pp.1090-1099, August 1986.

7. Cho, N.Z., Grossman, L. M., "Optimal Control for Xenon Spatial Oscillations in Load Follow of a Nuclear Reactor," Nuclear Science and Engineering, vol.83, pp.136-148, 1983.

8. Yoon, M. H., No, H. C., "Direct Numerical Technique of Mathematical Programming for Optimal Control of Xenon Oscillation in Load Following Operation," Nuclear Science and Engineering, vol.90, pp.203-220, 1985.

9. Park, G. T., Miley, G. H., "Application of Adaptive Control to a Nuclear Power Plant," Nuclear Science and Engineering, vol.94, pp.145-156, 1986.

10. Nair, P. P, Gopal, M., "Sensitivity-Reduced Design for A Nuclear Pressurized Water Reactor," IEEE Transactions on Nuclear Science, vol.NS-34, no.6, pp.1834-1842, December 1987.

11. Tsuji, M., Ogawa, Y., "Suboptimal Control of Pressurized Water Reactor Power Plant Using Approximate Model-Following Method," Journal of Nuclear Science and Technology, vol.24, no.12, pp.1022-1034, December 1987.

12. Saif, M., "A Novel Approach for Optimal Control of a Pressurized Water Reactor," IEEE Transactions on Nuclear Science, vol.36, no.1, pp.1317-1325, February 1989.

13. Godet, J., Gorez, R., "Optimal Control of Primary Coolant Temperature in a Nuclear Plant," Automatica, vol.18, no.4, pp.373-383, 1982.

14. Parlos, A. G. et al, "Nonlinear Multivariable Control of Nuclear Power Plants Based on the Unknown-but-Bounded Disturbance Model," IEEE Transactions on Automatic Control, vol.33, no.2, pp.130-137, February 1988.

15. Bernard, J. A., Henry, A. F., Lanning, D. D., "Application of the 'Reactivity Constraint Approach' to Automatic Reactor Control," Nuclear Science and Engineering, vol.98, pp.87-95, 1988.

16. Chung, "A Knowledge-Based System for Control of Xenon-Induced Spatial Power Oscillations During Load Following Operations," Ph.D. Dissertation, Iowa State University, 1988.

17. Arakawa, A., Sekimizu, K., Sumida, S., "Fuzzy Logic Control Application for BWR Recirculation Flow Control System," Journal of Nuclear Science and Technology, vol.25, no.3, pp.263-273, March 1988.

18. Doebelin, E., Control System Principles and Design, John Wiley, 1985.

19. Kerlin, T. W. et al, "Theoretical and Experimental Dynamic Analysis of the H.B. Robinson Nuclear Plant," Nuclear Technology, vol.30, pp.299-316, September 1976.

20. Akcasu, Z., Lellouche, G. S., Shotkin, L. M., Mathematical Methods in Nuclear Reactor Dynamics New York: Academic Press, 1971.

21. Kerlin, T. W., "Dynamic Analysis and Control of Pressurized Water Reactors," in C.T. Leondes (Ed.) Control and Dynamic Systems, Advances in Theory and Applications, New York: Academic Press, 1978.

22. El Wakil, M. M., Nuclear Heat Transport, Scranton:International Textbook, 1971.

23. Kim, S. N., Griffith, P., "PWR Pressurizer Modeling," Nuclear Engineering and Design, vol.102, pp.199-209, 1987.

24. Baek, S. M., No, H. C., Park, I. Y., "A Nonequilibrium Three-Region Model for Transient Analysis of Pressurized Water Reactor Pressurizer," Nuclear Technology, vol.74, pp.260-266, September 1986.

25. Arwood, D. C., Kerlin, T. W., "A Mathematical Model for an Integral Economizer U-Tube Steam Generator," Nuclear Technology, vol.35, pp.12-32, Mid-August 1977.

26. Hoeld, A., "A Theoretical Model for the Calculation of Large Transients in Nuclear Natural-Circulation U-Tube Steam Generators (Digital Code UTSG)," Nuclear Engineering and Design, vol.47, pp.1-23, 1978.

27. Kalra, S. P., "Modeling Transients in PWR Steam Generator Units," Nuclear Safety, vol.25, no.1, pp.33-52, January-February 1984.

28. Chao, Y. A., Attard, A., "A Resolution of the Stiffness Problem of Reactor Kinetics," Nuclear Science and Engineering, vol.90, pp.40-46, 1985.

29. Gupta, G. K., Sacks-Davis, R., Tischer, P. E., "A Review of Recent Developments in Solving ODEs," ACM Computing Surveys, vol.17, no.1, pp.5-47, March 1985.

30. Shampine, L. F., Gear, C. W., "A User's View of Solving Stiff Ordinary Differential Equations," SIAM Review, vol.21, no.1, pp.1-17, January 1979.

31. Bui, T. D., "Solving Stiff Differential Equations for Simulation," Mathematics and Computers in Simulation, vol.XXIII, pp.149-156, 1981.

32. Suzuki, S., Fukunishi, K., "An Auto-Tuning Method for Control System Parameters in Nuclear Power Plants," Nuclear Technology, vol.58, pp.379-387, September 1982.

33. Astrom, K. J., Anton, J. J., Arzen, K., E., "Expert Control," Automatica, vol.22, no.3, pp.277-286, 1986.

34. Bernard, J. A., "Applications of AI to Reactor and Plant Control," Nuclear Engineering and Design, vol.113, pp.219-227, 1989.

35. Duda, R. O., Shortliffe, E. H., "Expert Systems Research," Science, vol.220, no.4594, pp.261-268, April 1983.

36. Waterman, D. A., A Guide to Expert Systems, Reading: Addison Wesley, 1986.

37. Fikes, R., Kehler, T., "The Role of Frame-Based Representation in Reasoning," Communications of the ACM, vol. 28, no.9, pp.904-920, September 1985.

38. Winston, P. H., Artificial Intelligence, Second edition, Reading: Addison-Wesley, 1984.

39. Rich, E., Artificial Intelligence, New York: McGraw Hill Co, 1983.

40. Jackson, P., "Review of Knowledge-Representation Tools and Techniques," IEE Proceedings, vol.134, Pt.D, no.4, pp.224-230, July 1987.

41. Mamdani, E. H., "Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis," IEEE Transactions on Computers, vol. C-26, no.12, pp.1182-1191, December 1977.

42. Efstathiou, J., "Rule-based Process Control Using Fuzzy Logic," in E. Sanchez, L. A. Zadeh (Eds) Approximate Reasoning in Intelligent Systems, Decision and Control pp.145-158, Pergamon Press, 1987

43. Leung, K. S., Lam, W., "Fuzzy Concepts in Expert Systems," Computer, pp.43-56, September 1988.

44. Pang, D., Bigham, J., Mamdani, E. H., "Reasoning with Uncertain Information," IEE Proceedings, vol.134, Pt.D, no.4, pp.231-237, July 1987.

45. Zadeh, L. A., "Fuzzy Logic," Computer, pp.83-93, April 1988.

46. Zadeh, L. A., "The Concept of a Linguistic Variable and its Application to Approximate Reasoning-I," Information Sciences, vol.8, pp.199-249, 1975.

47. Bouchon, B., Despres, S., "Propagation of Uncertainties and Inaccuracies in Knowledge-Based System," in B. Bouchon and Yager (Eds.) Uncertainty in Knowledge Based Systems, Springer-Verlag, 1986.

48. Mamdani, E. H., Sembi., B. S., "Process Control Using Fuzzy Logic," in P. P. Wang, S. K. Chang (Eds.) Fuzzy Sets: Theory and Applications to Policy Analysis and Information Systems, New York:plenum Press, 1980.

49. King, P. J., Mamdani, E. H., "The Application of Fuzzy Control Systems to Industrial Processes," Automatica, vol.13, pp.235-242, 1977.

50. Procyk, T. J., Mamdani, E. H., "A Linguistic Self-Organizing Process Controller," Automatica, vol.15, pp.15-30, 1979

51. Braae, M., Rutherford, D. A., "Theoretical and Linguistic Aspects of the Fuzzy Logic Controller," Automatica, vol.15, pp.553-577, 1979.

52. Kiszka, J. B., Gupta, M. M., Nikiforuk, P. N., "Some Properties of Expert Control Systems," in M.M.. Gupta, A. Kandel, W. Bandler, J.B. Kiszka (Eds.) Approximate Reasoning in Expert Systems, Elsevier, 1985.

53. Sripada, N. R., Fisher, D. G., Morris, A. J., "AI Application for Process Regulation and Servo Control," IEE Proceedings, vol.134, Pt.D, no.4, July 1987.

54. Kickert, W. J. M., Van Nauta Lemke, H. R., "Application of a Fuzzy Controller in a Warm Water Plant," Automatica, vol.12, pp.301-308, 1976.

55. Tong, R. M., "A Control Engineering Review of Fuzzy Systems," Automatica, pp.559-569, 1977.

56. Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., Numerical Recipes, The Art of Scientific Computing, New York: Cambridge University Press, 1986.

57. Fliebig, R., Krueger, A., "GENDY - A General Reactor Dynamics Programme", GKKS 73/E/6, 1973.

58. Anon., TURBO PASCAL Owner's Handbook Version 4.0, Scotts Valley: Borland Int., 1987.

59. Anon., TURBO PROLOG, Owner's Handbook, Scotts Valley: Borland Int., 1986.

60. Engelmore, R. S., Majumdar, D., "A Quick Overview of Artificial Intelligence and Expert Systems," Nuclear Engineering and Design, vol.113, pp.173-180, 1989.

61. MacDonald, J .L., Koen, B. V., "Application of Artificial Intelligence Techniques to Digital Computer Control of Nuclear Reactors," Nuclear Science and Engineering, vol.56, pp.142-151, 1975.

62. Anon., "Preliminary Safety Analysis Report," Wisconsin Utilities Project, Wisconsin Electric Power Company, 1975.

63. Fournier, R. D., Hammer, M. F., Sun, B. K.-H, "Digital Control and Protection Retrofits in Nuclear Power Plants," Nuclear News, pp.50-57, November 1988.