TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADC	Analog to Digital Converter
AC	Alternating Current
AGA	Adaptive Genetic Algorithm
AI	Artificial Intelligence
ANN	Artificial Neural Network
AQM	Active Queue Management
BSFC	Binary String Fitness Characterisation
cGA	compact Genetic Algorithm
CPS	Comparative Partner Selection
D	Derivative
DAC	Digital to Analog Converter
DAQ	Data Acquisition
DC	Direct Current
DD	Data-Driven
EA	Evolutionary Algorithm
ECM	Evolutionary Computation Methods
FIR	Finite Impulse Response
FLBPI	Fuzzy Logic
FLBPI	Fuzzy Logic Based self tuning PI
FLBPID	Fuzzy Logic Based self tuning PID
FLC	Fuzzy Logic Controller
FLGA	Fuzzy Logic guided Genetic Algorithm
FLIC	Fuzzy Logic parallel Integral Controller
FPI	Fuzzy Proportional Integral
FRBS	Fuzzy Rule Base System
FS	Fuzzy System
FSPID	Fuzzy-Scheduled PID
GA	Genetic Algorithm
GP	Genetic Programming
HGA	Hierarchical Genetic Algorithm
HGU	Hydroelectric Generating Unit
I	Integral
IGBT	Insulated Gate Bipolar Transistor
KVL	Kirchoff Voltage Law
MA	Moving Average
MIE	Minimum Inference Engine
MRDE	Matrix Riccati Differential Equation
P	Proportional
PC	Personal Computer
PFP	Power Factor Precompensators
PGA	Parallel Genetic Algorithm
PI	Proportional-Integral

PID	Proportional-Integral-Derivative
PIE	Product Inference Engine
PLC	Programmable Logic Controller
PWM	Pulse Width Modulation
RS	Regulatory Sequence
RWM	Roulette Wheel Mechanism
SG	Structural Gene
sGA	standard GA
SPOGA	Semi-Parallel Operation Genetic Algorithm
SPOGA-FLBPI	SPOGA optimized Fuzzy Logic Based self tuning PI
SPOGA-FLBPID	SPOGA optimized Fuzzy Logic Based self tuning PID
SPOGA-FLIC	SPOGA optimized Fuzzy Logic parallel Integral Controller
SSPR	Stochastic Sampling with Partial Replacement
SSR	Stochastic Sampling with Replacement
SUMP	Shift and Uniform based Multi-Point
SUS	Stochastic Universal Sampling
TS	Takagi-Sugeno
TS-PID	Takagi-Sugeno-Proportional Integral Derivative
USB	Universal Serial Bus

NOMENCLATURES

A	Ampere
$\%O_s$	Percent of overshoot
$\%O_{s2}$	Percent of second overshoot
$\%S_p$	Percent of steady state error for position
$\%U_s$	percent of undershoot
A_P	Input element for position
A_{SSP}	Actual speed set point
A_v	Input element for speed
b	Damping ratio
b_l	Damping ratio of load
bl	Bit length
D	Change of error input of FLC
D_t	Delay time in process model
e	Controller input (error)
E	Error input of FLC
fit_p	Total fitness function for position control
fit_v	Total fitness function for speed control
$fit_{v,g}$	Total fitness function of SPOGA-optimized hybrid controller for speed control in the experiment x
$fit_{v,h}$	Total fitness function of non-SPOGA-optimized hybrid controller for speed control in the experiment x
fit_x	Total fitness function for speed and position control in experiment x
f_{os}	Fitness funxtion based on % overshoot
f_p	Fitness function for position based on ITAEp
f_{sp}	Fitness function based on %Sp
f_{ts}	Fitness function based on settling time
f_{vp}	Fitness function for the first 8-second starting speed
h	Output of fuzzy controller in FLBPI/FLBPID
$H_{(w,x,y)}$	Fuzzy rule string
H_P	Feedback element for position
H_v	Feedback element for speed
i_FIT	Fitness value for identification process
I_a	Armature current
IAE	Integral of Absolute value of Error
i_f	Field current
I_{pvx}	Improvement value for speed control in the experiment x
$ITAE$	Integral of Time Absolute value of Error
J	Moment of inertia
J_l	Moment of inertia of load
Ka	Feedback constant of saturation feedback
K_{ce}	Input scale for fuzzy change of error input

K_D	Derivative constant
K_{Dm}	Maximum value of KD
K_e	Input scale for fuzzy error input
K_I	Integral constant
K_{Im}	Maximum value of KI
K_P	Proportional constant
K_{Pt}	Output element for position
$K_{P\text{-}u}$	Ultimate value of KP
K_{Pm}	Maximum value of KP
K_{Pp}	Proportional constant of position controller
K_{Pv}	Variable position constant
K_u	Output scale for fuzzy output
K_v	Output element for speed
L	Electric inductance
m	Controller output
M	Output of controller to be fed to the signal conditioner
m_o	Initial condition of controller
M_p	Output of variable proportional controller
N	Number of points
n	Number of poles
NB	Negative big
N_c	Population size
N_g	Number of generation
n_i	Iteration number
NM	Negative medium
NS	Negative small
P	Process model
PB	Positive big
p_c	Crossover rate
P_E	Position error
p_m	Mutation probability/rate
PM	Positive medium
p_{m0}	Initial mutation probability/rate
P_{MV}	Output of position controller
P_{PV}	Position process value
P_R	Position reference
PS	Positive small
P_{SP}	Position set point
PV	Process value
R	Electric resistance
rpm	Revolutions per minute
S_E	Input to the speed controller
S_{EC}	Output of speed comparator
SP	Set point
S_{PV}	Speed process value
T	Rotor torque
T_l	Load torque
t_p	Peak time
t_r	Rise time

T_s	Sampling period
t_s	Settling time
T_u	Ultimate period of oscillation
U_c	Possible complex-valued poles
U	Output of FLC
V	Source voltage
W	Watt
w_l	The height of lth fuzzy membership functions
X	Input
Y	Output
Y_m	Real (measured) output of identification process
Z	Zero in process model
zc	Control genes
ZE	Zero
zp	Parameter genes
θ	Position
Φ	Flux magnetic
$\hat{y}$	Estimated output of identification process
$\bar{y}$	Mean value of real output in identification process

CHAPTER 1

INTRODUCTION

1.1 Background

Servomotors are used in a variety of applications in industrial electronics and robotics that includes precision positioning as well as speed control. Sometimes, the robot searches an optimum or approximate optimum non-collision path from start state to goal state according to a performance objective [1]. They use feedback controller to control the speed or the position or both. A robot used as manipulator has an end effector mounted on the last link. This end effector can be anything from a welding device to a mechanical hand used to manipulate the environment.

Consider a robot-manipulator system with its controller. The objective of the controller would be to move the robot arm via an effective control of the drive with DC servomotor [2]. The control system to implement can be classified into the following three stages [2] :

i. develop the circuit and the corresponding software for control of servomotors,
ii. develop a control system of the pneumodrive, and
iii. arrange a control system of the robot - manipulator.

Control theory is an interdisciplinary branch of engineering and mathematics, that deals with the behavior of dynamical systems. The desired output of a system is called the reference. When one or more output variables of a system need to follow a certain reference over time, a controller manipulates the inputs to the system to obtain the desired effect on the output of the system.

The basic continuous feedback control is PID controller. PID controllers have good performance but are not adaptive enough. This is appealing when the load is

changed, where the original controller generally cannot maintain the desired performance and thus should be re-designed for the new system conditions [3].

The pioneering work dealing with expert knowledge that can be well applied to the control of systems with uncertained, nonlinear dynamics is credited to Zadeh [4] who proposed fuzzy control theory to overcome the weakness of conventional controllers, and investigated by which owns good robustness [5]. Experimentally, the response of a fuzzy logic controller is slower than a PID controller. It has been reported in a number of papers that hybrid of PID or PI, with fuzzy logic in control system can overcome the set-back of fuzzy logic controller, see [6-9].

Fuzzy systems are capable of handling complex, non-linear and sometimes mathematically intangible dynamic systems using simple solutions. Fuzzy logic uses human-like but systematic properties of converting linguistic control rules based on expert knowledge into automatic control strategies. It requires time, experience and skills of the designer for the tedious fuzzy tuning exercise [10] because it lacks a learning mechanism [11]. It is expected that any algorithm can overcome some of the problems.

The most significant advantage of using evolutionary search lies in the gain of flexibility and adaptability to the task at hand and the global search characteristics. Among various evolutionary computation methods (ECM) is genetic algorithms (GA) which employ a random, yet directed search for locating global optimal solution [12].

GA is effective in acquiring the optimal or near-optimal for solving optimization problems [13]. The typical task of a GA in a control engineering application is finding the best values for a predefined set of free parameters which defining either a process model or a control law [14].

1.2 Issues on Optimization Using GA

Besides the advantages as explained in Sec 1.1, in contrast, GAs also have some problems as follows:

1. There are possibilities that the premature convergence and local maxima occur.
2. During the evolution of solutions using genetic programming (GP) there is generally an increase in average tree size or population size without a corresponding increase in fitness. This phenomenon is commonly referred to as bloat [15].
3. It is well known that the searching speed of the conventional genetic algorithms is not desirable [16].

1.3 Motivations

Based on the recent work on GA as reported in [12-16], the need to the issues of local maxima, premature convergence and bloat has been recognized.

There are some solutions to the premature convergence and local maxima problems. Adjusting to the proper mutation rate is basically able to solve the problem. In a recent work, Lau et. al. [13] consider the fuzzy logic guided genetic algorithm (FLGA) with shift and uniform based multi-point (SUMP) crossover and swap mutation in GA to fight against premature convergence, based on the knowledge that the probability of crossover and mutation are adaptable. Adaptive probability of crossover and mutation also prevent a local optima in solution [12],[17]. Selecting parents using binary string fitness characterisation (BSFC) and comparative partner selection (CPS) can also prevent the premature convergence problem [18].

There are some solutions to the bloat problem, one of them is by using spatial population structure in combination with local elitist replacement. However, this method is quite complex and is effective only for parallel genetic algorithms [15].

Motivated by this noticeable inadequacy, this research attempts to find answers to the problem of local maxima, premature convergence and bloat that make the searching speed slow. The recent work on GA shows some advances that have been made, but these remain the requirement to improve the searching speed. The task of solving such a searching speed problem using the method based on the GA application and its implementation in optimizing the performance of a controller for a DC servomotor is challenging and significant.

1.4 Objective and Contribution of Research

The objective of the research is improving the controller's performance (minimize the overshoot, settling time, IAE/ITAE and making the steady state error zero) for a DC servomotor application. A hybrid-fuzzy is selected as a controller which is optimized using GA in off-line mode. The GA is improved to reduce the searching speed problem.

In this research, the GA is applied to optimize the parameters of the hybrid-fuzzy for DC servomotor control using Simulink/MATLAB platform. Control application using this platform requires the simplicity in the programming. Optimizing the parameters of hybrid-fuzzy controller requires that some parameters to be optimized in parallel. The approach to solve the searching speed problem that would lead to the contribution of the research are as follows:

- Modify the conventional population initialization which use random process to the population initialization using patterned structure.
- Modify the conventional genetic operations to the new genetic operations which are appropriate for the problem to be solved
- Modify the conventional solution searching to the method based on elitism process [19]

The main contributions of this research are that, this thesis presents a new genetic algorithm in which improve the performance of the hybrid controllers with the reduction of the number of test runs (iterative number) and the duration time of the optimization process, and with the more consistent in the genetic process. The hybrid controller is also developed to improve the controller's performance.

1.5 Thesis Outline

The thesis consists of six chapters: introduction, literature review, simulation and real-time implementation on a test rig, discussion on the simulation results, discussion on the experiment results, and conclusions and the suggested future work.

Chapter 1 presents the background of the research, highlights the problem to be solved in the research, the motivations for the research, the research objective and approach, and the basis for the contributions of this research.

Chapter 2 presents the previous work related to the research topic, overview of the proposed methods based on the previous work, and discusses the basic theory used in the simulation and experiment work.

Chapter 3 discusses the simulation and experiment procedures from *s*-modelling of a DC servomotor, the hardware design, the GA design, etc. to the performance comparisons of the proposed method to the conventional methods.

Chapter 4 discusses the results of simulation for seven predefined conditions for evaluations of the controller's performances from the conventional to SPOGA-optimized hybrid-fuzzy controllers.

Chapter 5 discusses the results of hardware experiment for the seven predefined conditions for controllers from the conventional to SPOGA-optimized hybrid-fuzzy controllers.

Finally, Chapter 6 gives the conclusion for both the simulation and hardware experiment and suggests future research directions relevan to the controller for a servomotor.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter provides an overview of the work related to the research topic, the basis of the proposed method for use in the controller's performance optimization, and the foundation for the controller's performance criterion used in the simulation/experiment.

The basic theory starts with DC servomotor and power amplifier. The modelling of a DC servomotor based on the specification and the method of getting the *s*-transfer function using MATLAB command are presented. The basic control theory discussed is the feedforward and feedback control, PID controller, and fuzzy logic controller. The basic theory on genetic algorithm, parallel genetic algorithm, and hierarchical genetic algorithm are also presented. Finally, the controller performance criterion used for comparisons and evaluations are presented.

2.2 Servomotor

Servomotors are used in a variety of applications in industrial electronics and robotics that includes precision positioning as well as speed control. There are some types of motor : induction AC, synchronous AC, stepper DC, brushless DC, and brushed DC.

The speed of electrical motor can be sensorless but an estimation of parameters is required, as in their work in 2007 Karanayil et. al. [20] presented a new method of online estimation for the stator and rotor resistances of the induction motor for speed sensorless indirect vector controlled drives, using artificial neural networks.

2.3 System Identification

System identification is about building models from data. A data set is characterized by several pieces of information: the input and output signals, the sampling interval, the variable names and units, etc. Similarly, the estimated models contain information of different kinds, estimated parameters [21], their covariance matrices, and model structure and so on. The modelling process can be done using MATLAB command [22]. A gray-box modeling is useful to preserve the physical meaning of the model parameters and to naturally impose physical constraints to the model [23].

In 2005, Pereira et al. [24] proposed a system identification and PID tuning using GA sequentially. The identification is for first order plant cascaded by PID system. The result is better than PID control system optimized using Ziegler-Nichols.

Practically, the dynamic model of an embedded mechanical structure can change over time due to many factors such as structural deformation and sensor/actuator degradation [25].

2.4 Speed and Position Controller

There are two types of controller based on the control signal: analog controller, and digital controller. In digital controllers, efforts to reduce computational time is desirable and applicable when controllers have large number of states, when computational time becomes an issue [26].

The basic controller that is commonly used is PID controller. Yamamoto et al. in 2009, [27] proposed a new design scheme of PID controllers based on data-driven (DD) for nonlinear systems. In this method, a suitable set of PID parameters is automatically generated based on input/output data pairs of the controlled object stored in the database. The performance result is better as compared to fixed PID controller.

Position is a time integral of speed. Therefore, a certain position can be reached with a certain speed for a certain time duration. In 2007, Lacevic et al. [28] reported

an experiment of speed and position control in cascade mode where speed control loop is in the position control loop.

2.5 Fuzzy Logic Controllers

The basic idea of fuzzy logic control was suggested in notes published in 1968. After the basic idea of fuzzy logic control became well-understood, the use of fuzzy logic control is being pursued in many application areas especially in Japan. In most of current applications, the medium for fuzzy implementation in control is software. That is why fuzzy logic chips especially computers are more cheap and effective in implementation [4].

In his responses to the many misconceptions about fuzzy logic, Zadeh [29] gives the following explanation. Fuzzy logic is not fuzzy, but basically it is about a precise logic of imprecision and approximate reasoning. Hence, fuzzy logic is much more than a logical system and has many facets. The principal facets are: logical, fuzzy-set-theoretic, epistemic and relational. Most of the practical applications of fuzzy logic are associated with its relational facet.

The first type of fuzzy rule base system (FRBS) that deals with real inputs and outputs was proposed by Mamdani in 1974 [30], who was able to augment Zadeh's initial formulation in a way that allows it to apply a fuzzy system (FS) to a control problem [31].

2.6 Hybrid-Fuzzy Controllers

A combination of PID controller and fuzzy logic controller is called a hybrid-fuzzy controller. The purpose of hybrid-fuzzy controller is to overcome the problem of standalone PID controller and fuzzy logic controller.

In 1993, Zhen-Yu Zhao et al. [5] developed a fuzzy gain scheduling scheme of PID controllers for process control. Fuzzy rules and reasoning are utilized online to determine the controller parameters based on the error signal and its first difference.

Simulation results demonstrate that better control performance can be achieved in comparison with Ziegler-Nichols controllers and Kitamori's PID controllers.

Wang et al. in 2008, [32] proposed an optimal fuzzy control in HGU system which the membership functions and rules are optimized by GA and the real plant is identified using dynamic neural networks. Simulation results show that the advanced knowledge acquisition technique makes control parameters and rules of fuzzy controller arrive to optimization and its control performance is superior to conventional controller. The output of fuzzy controller is proportional based and paralleled with integrator to make the steady state error zero. The structure of optimal fuzzy controller is shown in Fig. 2.1.

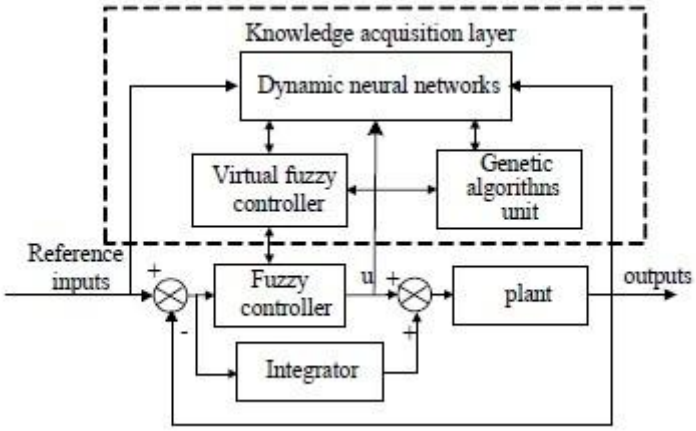


Fig. 2.1 Structure of optimal fuzzy control system [32]

In more recent work in 2010, Solihin et al. [33] proposed a PID controller which the parameters are tuned using fuzzy logic in the application of automatic gantry crane. First, the PID gains were obtained from root locus as a guideline for fuzzy output of the proposed fuzzy-tuned PID controllers. The fuzzy has error and error rate as inputs and the tuned gain as the output with Mamdani fuzzy inference system. There are seven singleton fuzzy membership functions for output and five Gaussian fuzzy membership functions for input. The scheme of fuzzy-tuned PID controller is shown in Fig. 2.2 where the range of α is [1.00,2.00] and obtained experimentally. The proposed controller has satisfactory performance

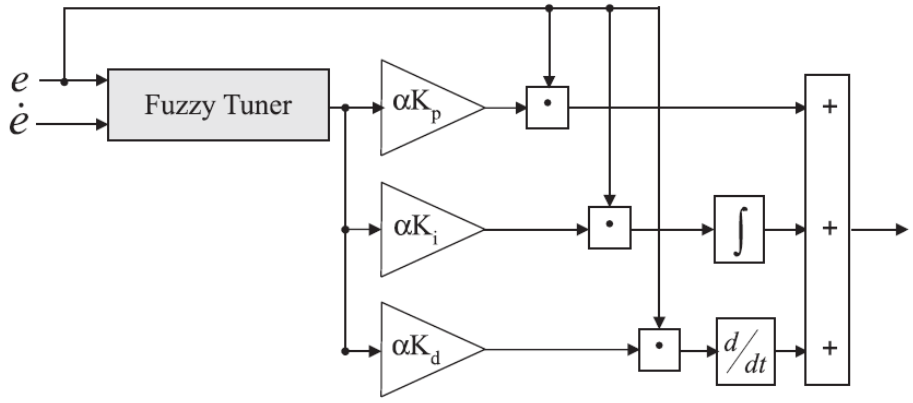


Fig. 2.2 Fuzzy-tuned PID controller scheme [33]

2.7 Overview of Genetic Algorithms (GA)

An AI field of major importance is comprised by the so-called soft-computing techniques that include genetic algorithms (GA), fuzzy logic (FL), artificial neural networks (ANN) and combination among them. The combination of soft-computing techniques for resolving scientific problems has produced results that could not have been extracted with traditional methods [34]. GAs are more robust than other local search algorithms because the population provides the advantage of maintaining diversity [35].

In their paper in 1996, Man et al. [36] introduces GA as a complete entity, in which knowledge of this emerging technology can be integrated together to form the framework of a design tool for industrial engineers.

In quite a recent paper in 2009, Balasubramaniam et al. [37] proposed a novel approach to find the solution of the matrix riccati differential equation (MRDE) for nonlinear singular systems using genetic programming (GP). The technique of GP is based on the evolution of the large number of candidate solutions through genetic operations such as reproduction, crossover and mutation. The GP approach to the problem is qualitatively better in terms of accuracy as compared to traditional Runge Kutta (RK) method.

One of the problem in GP is premature convergence that has been highlighted by Day et al. [18] who presented a binary string fitness characterization (BSFC) which gives both population measures and a pairwise mating strategy, and comparative partner selection (CPS), with aim of evolving a population that promotes effective solutions by reducing population-wide weakness. Actually, the CPS operation makes the process requires extra time (typically around 15%).

Fitness prediction is a technique used to replace fitness evaluations in evolutionary algorithms with a lightweight approximation that adapts with the solution population [38]. It is possible to identify the fundamental difficulties faced in many fitness prediction applications as :

i. Model training effort: Often significant computational effort is required to train the desired fitness model
ii. Level of approximation: It is often unclear what level of approximation is accurate enough to achieve desired results. High-quality approximations provide greater accuracy, but require more computation. Low-quality approximations are less accurate, but require less computation.
iii. Loss of accuracy: Similarly, even high-quality approximations are bound to have some loss accuracy due to either the model structure itself or the data available to tune it. In the worst case, this effect can hide or even change the global optimum in which case, exact fitness calculations are still needed to find the optimal solution.

In 2004, Xiu et al. [39] did an optimization design of Takagi-Sugeno-proportional integral derivative (TS-PID) fuzzy controllers based on GA. TS-PID controllers are a class of Takagi-Sugeno (TS) fuzzy controllers whose rule consequences employ PID expressions. The essential character of a typical TS-PID fuzzy controller is investigated to be a nonlinear PID controller. Based on GA, an optimal design method of TS-PID fuzzy controllers is discussed in detail by means of the analytical model. The simulations results of a marine control system show that a TS-PID fuzzy controller designed via GA's has a good performance.

The selection of a fitness function will influence the performance of GA. From any former studies, it can achieve a good effect to use

$J(ITAE) = \int_0^\infty t|e(t)|dt = min$ as object function in optimizing a control system. Since a GA is to search for the maximum of solutions, the $ITAE$ is changed by a fitness function as follows [39]:

$$f(i) = \frac{ITAE_{max} - ITAE(i)}{\sum_i (ITAE_{max} - ITAE(i))} \quad (2-1)$$

In 2008, Yang et al. [17] proposed a new improvement of GA, i.e. adaptive genetic algorithm (AGA). The probability of crossover and mutation is adapted according to the generation. When the generation is increased, the probability of mutation is decreased. The probability of crossover compensates the probability of mutation. When the crossover operation effect is feeble, the probability of mutation is increased. The superiority of AGA is: speed up the convergence, and restrain the premature.

In their work in 2007, Zhang et al. [12] proposed adaptive probability of crossover and mutation in GA using fuzzy logic. This is not only improved the convergence rate of the GA, but also prevents a local optima in solution. The result is better than using fixed probability of crossover and mutation.

In 2009, Lau et al. [13] proposed a shift and uniform based multi-point (SUMP) crossover and swap mutation in GA which are adjusted using fuzzy logic, called Fuzzy logic guided genetic algorithm (FLGA). In this method, the probability of crossover and mutation are adjusted by fuzzy logic after ten consecutive generations. FLGA with SUMP can fight against premature convergence but requires longer running time than standard GA since that the operations involved in fuzzy logic involve additional computation effort.

In a quite recent paper in 2009, Duzinkiewics et al. [19] presented a genetic hybrid applied to a predictive controller for optimized dissolved-oxygen tracking at lower control level. The used initialization scheme is hybrid. A fixed number of initial solutions are obtained using a priori knowledge about the aeration system. The remaining solutions are initialized randomly. Elitism process is done before the next

generation to maintain the diversity of the population while the best solutions found so far are preserved and used in the search process.

There is a method of parallel genetic algorithms in which a parallel client server single population is configured in order to reach a global solution in the least possible iteration. The fitness value assessment is done on different parallel processors; as genetic operators affect all the population in this method the model is called a global genetic algorithm. This method was applied to optimize the performance of the first three Dejong function [40].

In 2003, Fatta et. al. [41] did an experiment of parallel GA using two basic ones: the simple global model and the coarse grained model, to design a fuzzy proportional integral (FPI) controller for active queue management (AQM) on Internet routers. The parallel GA is valid enough as a tool for optimal tuning of the fuzzy controller parameters.

In his paper in 1996, Lis [42] proposed a parallel genetic algorithm with dynamic mutation probability. If p_{m0} is initial mutation probability and n_i is iteration number, then the mutation probability (p_m) is [42]

$$p_m \approx p_{m0}\left(\frac{1}{n_i}\right) \qquad (2-2)$$

There is no need for choosing any initial mutation probability and decreasing mutation probability convergence rate.

An application of hierarchical genetic algorithms (HGA) to optimize the membership functions of fuzzy logic controller (FLC) is shown in Fig. 2.3.

There were seven membership functions of FLC as an initial. The control genes (z_c) which the values are {0, 1} controlled the existence of membership functions. If $z_c = 0$ then the corresponding membership function would be deleted. Otherwise, it would remain exist. The parameter genes (z_p) which had real values controlled the boundaries of membership function [43]. This method will results in that the crossover points of a membership function have not the value equal to 0.5.

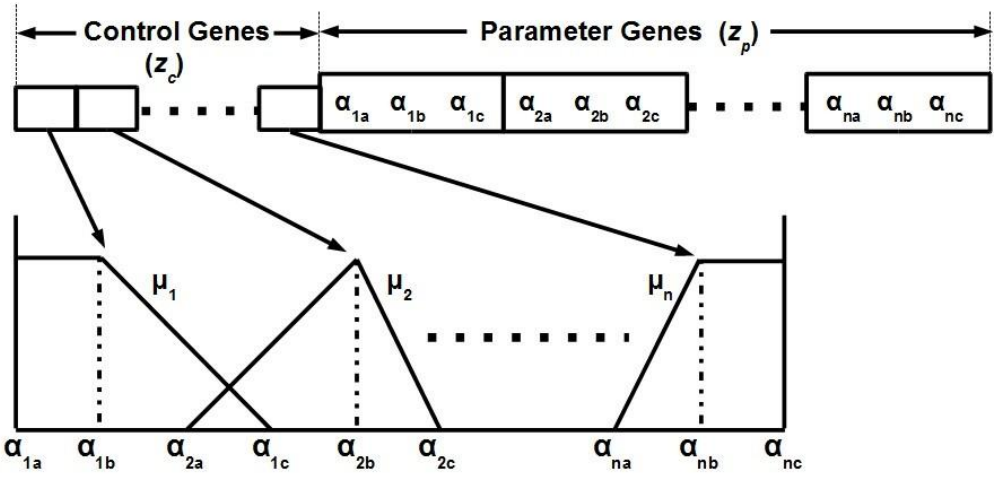


Fig. 2.3 A hierarchical membership chromosome [43]

2.8 Some Related Work on GA Applications

Genetic algorithms can be applied on the optimization of hybrid-fuzzy controllers in either simulation experiments or hardware experiments.

In 2006, Bousserhane et al. [44] did an optimal fuzzy gain scheduling of PI controller to the speed control of induction motor. The parameters of PI controller is scheduled by FLC where the parameters of FLC is first optimized by genetic algorithm. The performance result is better as compared to FLC which is optimized by human operator.

Applying GA on a memory-constrained hardware (e.g. microcontroller, PLC, etc) needs a compact GA (cGA) which proposed by Mininno et al. [14] in 2008. A cGA iteratively processes the PV with updating mechanisms that mimic the typical selection and recombination operations performed in a standard GA (sGA) until a stopping criterion is met. A cGA is almost equivalent to a sGA with binary tournament selection and uniform crossover on a number of test problems, and also suggested some mechanisms to alter the selection pressure in the cGA.

2.9 The Proposed Methods

Based on the previous work on the electrical motor, the system identification, the control technique, fuzzy logic controller, hybrid-fuzzy controller, and the controller parameters optimization using genetic algorithm, the following steps (i to viii) are identified in this work as guidelines for the controller development:

i. DC motor is used as a plant because it is easy to control the speed and the starting torque is good.
ii. The DC servomotor is identified to get the *s*-transfer function for off-line mode control using MATLAB command as in [22].
iii. Cascaded speed and position control is used where the speed control loop is in the position control loop as in [32]
iv. The conventional, fuzzy, and hybrid-fuzzy controllers are compared and the best one is selected to be optimized using GA.
v. There are three hybrid-fuzzy controllers: (1) Fuzzy-logic based self-tuning PI controller (FLBPI) as in [45], (2) Fuzzy-logic based self-tuning PID controller (FLBPID) as in [45] with additional D component, and (3) Fuzzy logic parallel integral controller (FLIC) as in [32].
vi. The new GA-based optimization algorithm, namely Semi-parallel operation GA (SPOGA) is proposed. The method is based on HGA as in [43] but the string structure is for parallel optimization of parameters.
vii. The structure of twisted ring counter is used as a population initialization as a replacement of random generation.
viii. The solution process is done by searching the chromosome with the best fitness value among the all chromosomes in all generations, this method is based on elitism process as in [19].

2.10 DC Servomotor and Power Amplifier

An electric motor which has control system components is called a servomotor. Basically, any motor can be used in a servo system [46]. There are some types of motor : induction AC, synchronous AC, stepper DC, permanent magnet DC, serial field DC, shunt field DC, compound field DC, brushless DC, and brushed DC.

Stepper motor are widely used in applications which need high precision speed and position, but if there are mechanical changes such as load torque disturbances and inertia variations, can lead to a loss of synchronism for high stepping rates [47].

According to the field type of DC motor, there is also a permanent magnet stepper motors which can spontaneously reverse their direction of rotation when controlled in full step, open-loop mode [48]. In the closed loop mode, they cannot spontaneously reverse their direction.

DC motors have better starting torque than AC motors although they are more expensive than AC motors [46]. There are several method in controlling the DC servomotor, one of them is armature-controlled DC servomotors which are widely used in the motion control area in the process control industry [49].

2.10.1 Power Amplifier

Controller signals usually have the range of [0.00,10.00] volts with the current output is limited in the order of milliampere. Consequently, a controller needs a power amplifier to drive a DC motor. One of the power amplifier which can drive a motor is chopper type where the process needs DC-AC and AC-DC converters.

Power factor precompensators (PFP) are an important class of switched ac-dc converters. The main drawback is that they require very high switching frequency (typically in few hundred kHz) leading to high converter losses. Pulse width modulation (PWM) control techniques are more interesting which can be implemented using lower switching frequency (typically 10 kHz) [50].

PWM are widely used in power electronic system and control systems. The reasons for the wide applicability of PWM are as follows:

i. the control variable has only two or three values such that the realization of PWM control is simple and
ii. the PWM can process large signals with high efficiency and low sensitivity of noise [51].

2.10.2 Transfer Function Model of a DC Motor

A DC motor has two main components: electrical component and mechanical component. The electrical circuit of the armature and the free body diagram of the rotor are shown in Fig. 2.4.

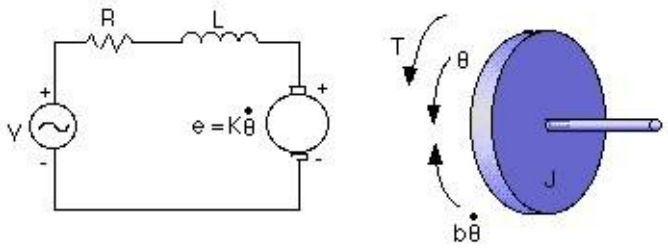


Fig. 2.4: Electrical circuit and free body diagram of the rotor of DC motor [52]

The flux magnetic $\emptyset$ in the field coil is a constant k_1 by the assumption that the field current i_f is constant [53]. If T is the rotor torque, k_2 is torque constant, and i_a is armature current then

$$T(t) = k_2 i_a(t) \qquad (2-3)$$

If L is electric inductance, R is electric resistance, V is source voltage, K is electromotive force constant, and θ is rotor position, then using KVL principle for the armature circuit,

$$V(t) = L\frac{\partial i_a}{\partial t} + R.i_a(t) + K\frac{\partial \theta}{\partial t} \qquad (2-4)$$

If J is moment of inertia of the rotor, J_l is moment inertia of load, b is damping ratio of the mechanical system, b_l is damping ratio of load and T_l is load torque, then using Newton II principle for mechanical system,

$$T(t) = T_l(t) + J\frac{\partial \theta(t)}{\partial t} + b\theta(t) \qquad (2-5)$$

$$T_l(t) = J_l\frac{\partial \theta(t)}{\partial t} + b_l\theta(t) \qquad (2-6)$$

Substituting Eq (2 - 6) to Eq (2 - 5) the result is:

$$T(t) = (J + J_l)\frac{\partial\theta(t)}{\partial t} + (b + b_l)\theta(t) \qquad (2 - 7)$$

Converting Eq. (2 - 3), (2 - 4), and (2 - 7) to the *s*-domain results :

$$T(s) = k_2 I_a(s) \qquad (2 - 8)$$

$$V(s) = sLI_a(s) + RI_a(s) + sK\theta(s) \qquad (2 - 9)$$

$$T(s) = s(J + J_l)\theta(s) + (b + b_l)\theta(s) \qquad (2 - 10)$$

$I_a(s)$ can be found using Eq (2 - 9)

$$I_a(s) = \frac{V(s) - sK\theta(s)}{Ls + R} \qquad (2 - 11)$$

Practically, $k_2 = K$ [53]. If $s\theta(s)$: speed, $J_e = J + J_l$, $b_e = b + b_l$, then substituting Eq (2 - 8), (2 - 10), and (2 - 11) gives the result :

$$\frac{s\theta(s)}{V(s)} = \frac{K}{J_e Ls^2 + (J_e R + b_e L)s + b_e R + K^2} \qquad (2 - 12)$$

When $J_l = J$ and $b_l = b$, Eq (2 - 12) becomes

$$\frac{s\theta(s)}{V(s)} = \frac{K/2}{JLs^2 + (JR + bL)s + bR + 2K^2} \qquad (2 - 13)$$

When no load, Eq (2 - 12) becomes

$$\frac{s\theta(s)}{V(s)} = \frac{K}{JLs^2 + (JR + bL)s + bR + K^2} \qquad (2 - 14)$$

Dividing Eq (2 - 12) to Eq (2 - 14) by *s* to produce the transfer function of the position,

$$\frac{\theta(s)}{V(s)} = \frac{K}{J_e L s^3 + (J_e R + b_e L)s^2 + (b_e R + K^2)s} \quad (2 - 15)$$

$$\frac{\theta(s)}{V(s)} = \frac{K/2}{JLs^3 + (JR + bL)s^2 + (bR + 2K^2)s} \quad (2 - 16)$$

$$\frac{\theta(s)}{V(s)} = \frac{K}{JLs^3 + (JR + bL)s^2 + (bR + K^2)s} \quad (2 - 17)$$

An example of DC motor parameters are shown as discussed in [52] as in Table 2.1.

Table 2.1 Example of DC motor parameters

Parameter	Symbol	Unit	Magnitude
Moment of inertia	*J*	kg.m^2	0.01
Damping constant	*b*	kg.m^2/sec	0.1
Constant	*K*	-	0.01
Resistance	*R*	ohm	1.00
Inductance	*L*	henry	0.50
Source voltage (input)	*V*	volt	Variable
Rotor position (output)	*θ*	rad	Variable

In Table 2.1, the rotor and shaft are assumed to be rigid. Thus, the transfer functions of the speed and position are given by:

$$\frac{s\theta(s)}{V(s)} = \frac{20}{s^2 + 102s + 200.2} \quad \text{(speed in rad/s)} \quad (2 - 18)$$

$$\frac{\theta(s)}{V(s)} = \frac{20}{s^3 + 102s^2 + 200.2s} \quad \text{(position in rad/s)} \quad (2 - 19)$$

When the plant is loaded with the same mechanical system, then

$$\frac{s\theta(s)}{V(s)} = \frac{10}{s^2 + 102s + 200.1} \quad \text{(speed in rad/s)} \quad (2 - 20)$$

$$\frac{\theta(s)}{V(s)} = \frac{10}{s^3 + 102s^2 + 200.1s} \quad \text{(position in rad/s)} \quad (2 - 21)$$

In the simulation experiment, the loaded DC motor can be approximated by multiplying the plant with 0.5.

2.10.3 Input-Output Modelling of a DC Servomotor

Servomotor controllers need optimization to give a good performance as desired. Sometimes, it requires more time and has big risk in optimization process. To overcome the problem, a real plant (DC motor) is identified to get a transfer function and build a virtual controller. The virtual controller is optimized, and then the optimized parameters are applied to the real controller in the real control system.

The system identification problem is to estimate a model of a system based on the observed input-output data [22]. The typical identification process consists of stages where the model structure is iteratively selected, compute the best model in the structure, and evaluate this model's properties [22]. This cycle can be itemized, as follows [22]:

i. Design an experiment and collect input-output data from the process to be identified
ii. Examine the data. Refine the data by removing trends and outliers, and select useful portions of the original data. Apply filters to the data to enhance important frequency ranges
iii. Select and define a model structure (a set of candidate system descriptions), within which a model is to be found
iv. Compute the best model in the model structure according to the input-output data and a given criterion for goodness of fit.
v. Examine the properties of the model obtained. If the model is good enough, then stop; otherwise go back to step iii to try another model structure. Attemp other estimation methods (step iv), or work further on the input-output data (steps i and ii).

If T_s is sampling period, then the data for identification is obtained from the MATLAB command below:

data=iddata(workspace_output,workspace_input,T_s);

Using MATLAB command, the general command to get the process model is as follows:

model=pem(data, 'PnD$_t$ZU$_c$');

where,

P : (required) for process model
n : 0,1,2 or 3 (required) for the number of poles
D_t : (optional) to include a time-delay term
Z : (optional) to include a process zero (numerator term)
U_c : (optional) to indicate possible complex-valued (underdamped) poles

To select the best process model, the fitness function has to be obtained. If i_FIT is the fitness function for identification process, Y_m is the real (measured) output, $\hat{Y}$ is the estimated output, and $\bar{Y}$ is the mean value of real output, then the fitness function can be obtained from the formula as shown in Eq (2 - 22), [22]

$$i\_FIT = \left(\frac{1 - NORM\left(Y_m - \hat{Y}\right)}{NORM(Y_m - \bar{Y})}\right) * 100 \qquad (2 - 22)$$

2.11 Overview on Control Theory

Control systems are an integral part of modern society. It consists of subsystems and processes (or plants) assembled for the purpose of controlling the outputs of the processes. In its simplest form, a control system provides an output or response for a given input or stimulus, as shown in Fig. 2.5 [54].

Fig. 2.5 Simplified description of a control system [54]

2.11.1 Feed Forward and Feedback Control

A generic open-loop system or feed forward control system is shown in Fig. 2.6. It starts with a subsystem called an *input transducer*; which converts the form of the input to that used by the controller. The controller drives a process or plant. The input is sometimes called the *reference* or *set-point*, while the output can be called the *controlled variable* or *process value*. Other signals, such as disturbances, are shown added to the controller and process outputs via summing junctions, which yield the algebraic sum of their input signals using associated signs. These systems are simply commanded by the input but do not correct for disturbances [54].

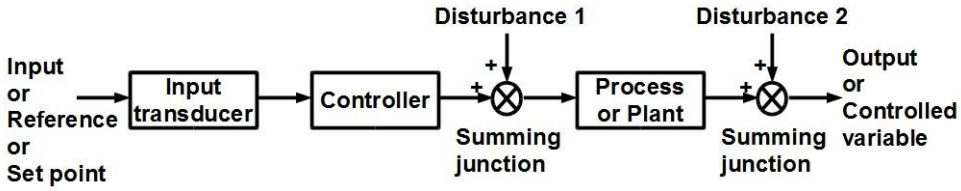


Fig. 2.6 Block diagram of feed forward control system [54]

The disadvantages of feed forward control systems, namely sensitivity to disturbances and inability to correct for these disturbances, may be overcome in closed-loop systems or feedback control systems. The generic architecture of a feedback control system is shown in Fig. 2.7 [54].

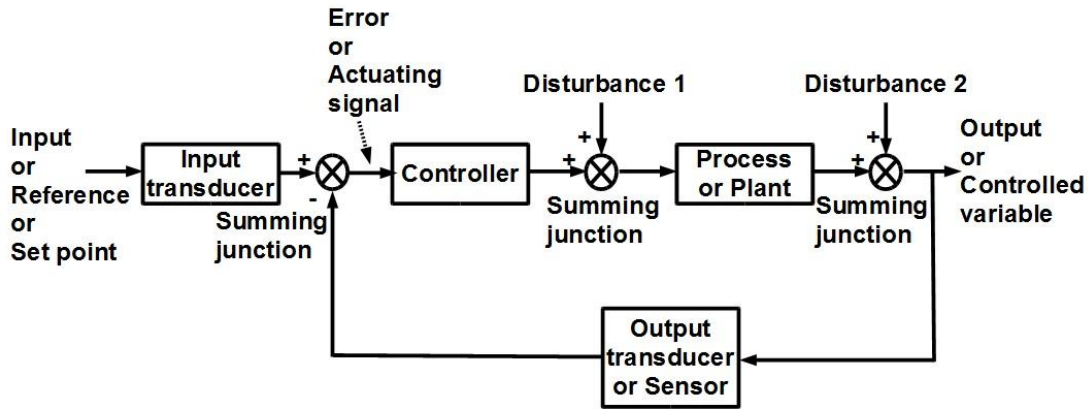


Fig. 2.7 Block diagram of feedback control system [54]

The input transducer converts the form of the input to the form used by controller. An output transducer or sensor measures the output response and converts it into the form used by the controller [54].

The first summing junction algebraically adds the signal from the input to signal from the output, which arrives via the feedback path, the return path from the output to the summing junction. In Fig. 2.7, the output signal is subtracted from the input signal. The result is generally called the *actuating signal*. However, in systems where both the input and output transducers have unity gain, the actuating signal's value is equal to the actual difference between the input and output. Under this condition, the actuating signal is called the *error* [54].

The feedback system compensates for disturbances by measuring the output response, feeding that measurement back through a feedback path, and comparing that response to the input at the summing junction. If there is any difference between the two responses, the system drives the plant, via the actuating signal, to make a correction. If there is no difference, the system does not drive the plant, since the plant's response is already the desired response [54].

Feedback systems, then, have the obvious advantage of greater accuracy than feed forward systems. They are less sensitive to noise, disturbances, and changes in the environment. Transient response and steady-state error can be controlled more conveniently and with greater flexibility in closed-loop systems, often by a simple adjustment of gain (amplification) in the loop and sometimes by redesigning the controller. On the other hand, feedback systems are more complex and expensive than feed forward systems [54].

In many modern systems, the controller (or compensator) is a digital computer. The advantage of using computer is that many loops can be controlled or compensated by the same computer through time sharing. Furthermore, any adjustments of the compensator parameters required to yield a desired response can be made by changes in software rather than hardware. The computer can also perform supervisory functions, such as scheduling many required applications [54].

2.11.2 PID Controller

PID controller is a combination of three basic characteristics of system, that is, proportional (P), integral (I), and derivative (D). There are two types of system combination: parallel and cascaded. If m is output, e is input, m_0 is initial condition or offset, K_P is proportional constant, K_I is integral constant and K_D is derivative constant, then in parallel combination, the general form of PID controller is formulated as [55]

$$m(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt} + m_0(t) \qquad (2 - 23)$$

in t-domain, and

$$M(s) = K_P E(s) + \frac{K_I}{s} E(s) + K_D s E(s) + M_0(s) \qquad (2 - 24)$$

in s-domain

In more general form, Eq. (2 - 23) can be written as,

$$m(t) = m_P(t) + m_I(t) + m_D(t) + m_0(t) \qquad (2 - 25)$$

If T_s is sampling period, then Eq. (2 - 25) can be written in discrete form as follows [56] :

$$m(kT_s) = m_P(kT_s) + m_I(kT_s) + m_D(kT_s) + m_0(kT_s) \qquad (2 - 26)$$

The proportional part, $m_P(kT_s)$, of Eq. (2 - 26) is [56]

$$m_P(kT_s) = K_P e(kT_s) \qquad (2 - 27)$$

The integral part, $m_I(kT_s)$, of Eq. (2 - 26) is approximated as

$$m_I(kT_s) = K_I T_s \sum_k e(kT_s) + m_I(0) \qquad (2 - 28)$$

The derivative part, $m_D(kT_s)$, of Eq. (2 - 26) is approximated as

$$m_D(kT_s) = \frac{K_D}{T_s}[e(kT_s) - e(kT_s - T_s)] \quad (2 - 29)$$

The parameters of PID controller (K_P, K_I and K_D) can be set using ultimate cycle method (Ziegler-Nichols). If K_{P-u} is ultimate value of K_P, i.e. the value of K_P where the output response is constant oscillation and T_u is the ultimate period of oscillation, then there are three combination of PID controller [57]:

a. P controller

$$K_P = 0.5K_{P-u} \quad (2 - 30)$$

b. PI controller

$$K_P = 0.45K_{P-u}$$
$$K_I = \frac{1.2}{T_u} \quad (2 - 31)$$

c. PID controller

$$K_P = 0.6K_{P-u}$$
$$K_I = \frac{2}{T_u} \quad (2 - 32)$$
$$K_D = \frac{T_u}{8}$$

2.11.3 Integral Windup

When the error or actuating signal has the same sign for a long time, the integral part output of PID controller will increase larger and larger. This symptom is called *integral windup*. The integral windup can inflict overshoot [56].

One of the methods to overcome the problem caused by integral windup is saturation feedback [58]. The output of the integral part in Eq. (2 - 23), Eq. (2 - 24) and Eq. (2 - 28) will increase as the time increase but the manipulated variable is limited by hardware, e.g. data acquisition (DAQ). The difference between manipulated variable and controller output is added to the error or actuating signal through the feedback constant, *Ka*. The block diagram of saturation feedback is shown in Fig. 2.8.

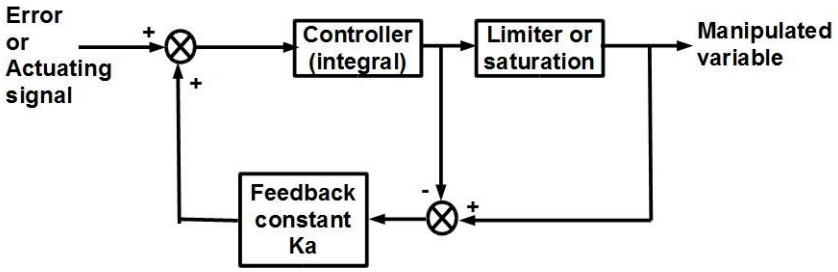


Fig. 2.8 Saturation feedback as an anti integral windup [58]

2.11.4 Fuzzy Logic Controller

According to the Oxford English Dictionary, the word "fuzzy" is defined as "blurred, indistinct; imprecisely defined; confused, vague". Fuzzy systems are knowledge-based or rule-based systems. The heart of a fuzzy system is a knowledge base consisting of the so-called fuzzy IF-THEN rules. A fuzzy IF-THEN rule is an IF-THEN statement in which some words are characterized by continuous membership functions [59]. For example, if X is the speed of a car, Y is force application to the accelerator, then the following is a fuzzy IF-THEN rule [59]:

IF X is *high*, THEN Y is *less* (2 - 33)

where the words "high" and "less" are characterized by the membership functions shown in Fig. 2.9 and Fig. 2.10 respectively. A fuzzy system is constructed from a collection of fuzzy IF-THEN rules [59].

The core of a membership function for some fuzzy set A is defined as the region of the universe that is characterized by complete and full membership in the set A. That is, the core comprises those elements x of the universe such that $\mu_{\mathrm{A}}(x) = 1$ [60].

The support of a membership function for some fuzzy set A is defined as that region of the universe that is characterized by nonzero membership in the set A. That is, the support comprises those elements *x* of the universe such that $\mu_{\mathrm{A}}(x) > 0$ [60].

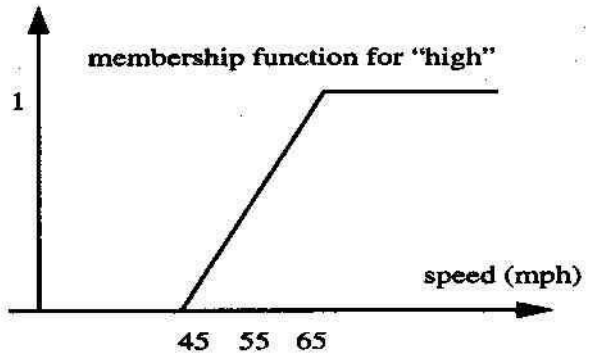


Fig. 2.9 Membership function for 'high" where the horizontal axis represents the speed of the car and the vertical axis represents the membership value for "high" [59]

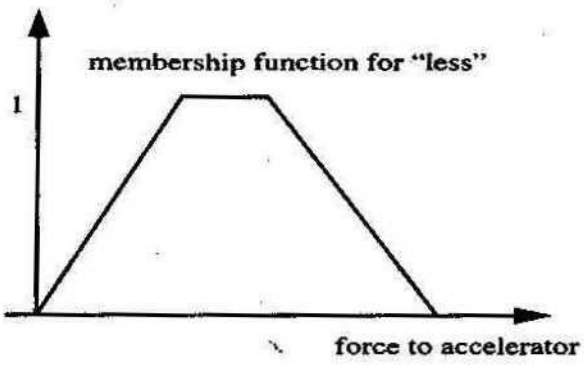


Fig. 2.10 Membership function for "less", where the horizontal axis represents the force applied to the accelerator and the vertical axis represents the membership value for "less" [59]

The boundaries of a membership function for some fuzzy set A are defined as that region of the universe containing elements that have a nonzero membership but not complete membership. That is, the boundaries comprise those elements x of the universe such that $0 < \mu_{\mathrm{A}}(x) < 1$. These elements of the universe are those with some degree of fuzziness, or only partial membership in the fuzzy set A [60].

A convex fuzzy set is described by a membership function whose membership values are strictly monotonically increasing, or whose membership values are monotonically decreasing, or whose membership values are strictly monotonically

increasing then strictly monotonically decreasing with increasing values for elements in the universe. Said another way, if, for any elements x, y, and z in a fuzzy set A, the relation $x < y < z$ implies that

$$\mu_{\mathrm{A}}(y) \geq \min[\mu_{\mathrm{A}}(x), \mu_{\mathrm{A}}(z)] \qquad (2 - 34)$$

then A is said to be a convex fuzzy set [60].

A special property of two convex fuzzy sets, say A and B , is that the intersection of these two convex fuzzy sets is also a convex fuzzy set. That is, for A and B, which are both convex, A ∩ B is also convex [60].

The crossover points of a membership function are defined as the elements in the universe for which a particular fuzzy set A has values equal to 0.5, i.e., for which $\mu_{\mathrm{A}}(x) = 0.5$ [60].

The height of fuzzy set A is the maximum value of the membership functions, i.e., $\mathrm{hgt(A)} = \max\{\mu_{\mathrm{A}}(x)\}$. If the $\mathrm{hgt(A)} < 1$, the fuzzy set is said to be subnormal [60].

The first step in building fuzzy system is to gather fuzzy IF-THEN rules based on human experiences or skills. The next step is combining the rules to the unique system [59].

There are three types of fuzzy system commonly used: Pure fuzzy systems, Takagi-Sugeno-Kang (TSK) fuzzy systems, and Fuzzy systems with fuzzifier and defuzzifier [59]. Basic configuration of fuzzy systems with fuzzifier and defuzzifier is shown in Fig. 2.11.

The fuzzifier is defined as a mapping from a real valued point $x^* \in U \subset R^n$ to a fuzzy set A' in U [59]. There are three criteria in designing fuzzifier [59]:

i. The fuzzifier should consider the fact that the input is at the crisp point
ii. If the input to fuzzy system is corrupted by noise, then the fuzzifier should help to supress the noise
iii. The fuzzifier should help to simplify the computations involved in the fuzzy inference engine.

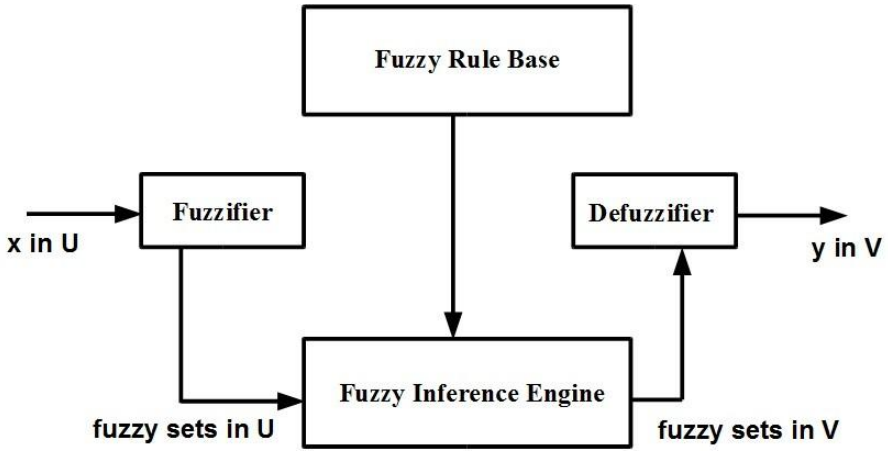


Fig. 2.11 Basic configuration of fuzzy systems with fuzzifier and defuzzifier [59]

There are three types of fuzzifier [59]: Singletion fuzzifier, Gaussian fuzzifier, and Triangle fuzzifier. Singleton fuzzifier can fulfill the three criteria above. It maps a real valued point $x^* \in U$ into a fuzzy singleton A' in U, which has the membership value 1 at x^* and zero at all other point in U [59]:

$$\mu_{A'}(x) = \begin{cases} 1 & if\ x = x^* \\ 0 & otherwise \end{cases} \qquad (2 - 35)$$

In fuzzy inference engine, the fuzzy logic base is used to combine IF-THEN rules in the fuzzy rules base to the mapping from a fuzzy set A' in U into a fuzzy set B' in V [59]. There several types of fuzzy inference engine [59]:

i. Product inference engine
ii. Minimum inference engine
iii. Lukasiewicz inference engine
iv. Zadeh inference engine
v. Dienes-Rescher inference engine

On the simplicity reason, product inference engine (PIE) and minimum inference engine (MIE) are commonly used as a fuzzy inference engine. For M rules and n membership functions, the PIE is formulated as [59]

$$\mu_{B^l}(y) = \max_{l=1}^{M} \left[\sup_{x \in U} \left(\mu_{A^l}(x) \prod_{i=1}^{n} \mu_{A_i^l}(x_i) \mu_{B^l}(y) \right) \right] \qquad (2 - 36)$$

and the MIE is formulated as [59]

$$\mu_{B^l}(y) = \max_{l=1}^{M} \left[\sup_{x \in U} \min\left(\mu_{A^l}(x), \mu_{A_1^l}(x_1), \ldots, \mu_{A_n^l}(x_n), \mu_{B^l}(y)\right) \right] \qquad (2 - 37)$$

Defuzzifier is defined as a mapping from a fuzzy set B' in $V \subset R$ (output of the fuzzy inference engine) to a crisp value $y^* \in V$ [59]. There are three criteria in designing defuzzifiers [59]:

i. Plausibility: the point y^* represents B' from an intuitive point of view
ii. Computational simplicity: particularly important for real time application
iii. Continuity: small change of B' should not make large change of y^*.

There are three types of defuzzifier [59]: center of gravity defuzzifier, center average defuzzifier, and maximum defuzzifier. Center average defuzzifier can fulfil all of the three criteria. If w_l is the height of l^{th} fuzzy membership function then for M fuzzy membership functions it can be formulated as [59]

$$y^* = \frac{\sum_{l=1}^{M} \bar{y}^l w_l}{\sum_{l=1}^{M} w_l} \qquad (2 - 38)$$

Eq. (2 - 38) shows that the crisp value of fuzzy output is the sum of core of l^{th}-fuzzy membership function multiplied by the height of l^{th}-fuzzy membership function over the sum of the height of l^{th}-fuzzy membership function.

2.12 Evolutionary Algorithms

Conventional search techniques, such as hill-climbing, are often incapable of optimizing non-linear multimodal functions. In such cases, a random search method might be required. However, undirected search techniques are extremely inefficient for large domain [61]. To overcome the problem, an artificial intelligence (AI) is introduced.

AI is the intelligence of machines and the branch of computer science that aims to create it. One of the subfield of AI is evolutionary computation that involves combinatorial optimization problems.

In AI, an evolutionary algorithm (EA) is a subset of evolutionary computation. An EA uses some mechanisms inspired by biological evolution, i.e. : reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions "live".

The popular type of EA is neural network, simulated annealing, ant colony algorithm and genetic algorithm (GA) as discussed in [62-77]. One seeks the solution of a problem in the form of strings of numbers (traditionally binary, although the best representations are usually those that reflect something about the problem being solved), by applying operators such as recombination and mutation. This type of EA is often used in optimization problems [78].

2.12.1 Genetic Algorithms

A GA is a directed random search technique, invented by Holland in 1975 as mentioned by Pham et al. [61], which can find the global optimal solution in complex multi-dimensional search spaces. A GA is modelled on natural evolution in that the operators it employs are inspired by the natural evolution process. These operators, known as genetic operators, manipulate individuals in a population over several generations to improve their fitness gradually. Individuals in a population are likened to chromosomes and usually represented as strings of binary numbers [61].

A simple GA is composed of three main components: initialization; evaluation; and genetic operators. There are three genetic operators: selection, crossover, and mutation [79].

The process of GA is shown in Fig. 2.12. After initialization of population with certain bit size and population size, then the fitness evaluations for each population are performed to select fit populations to be crossed (in crossover operation) and then

to be mutated (mutation operation). Next, the fitness evaluations for each population are done every time after mutation operation to find the best fitness for first generation. Following this, crossover and mutation are conducted for the next generations until getting the certain generation set by user. The solution is the best fit of all generations.

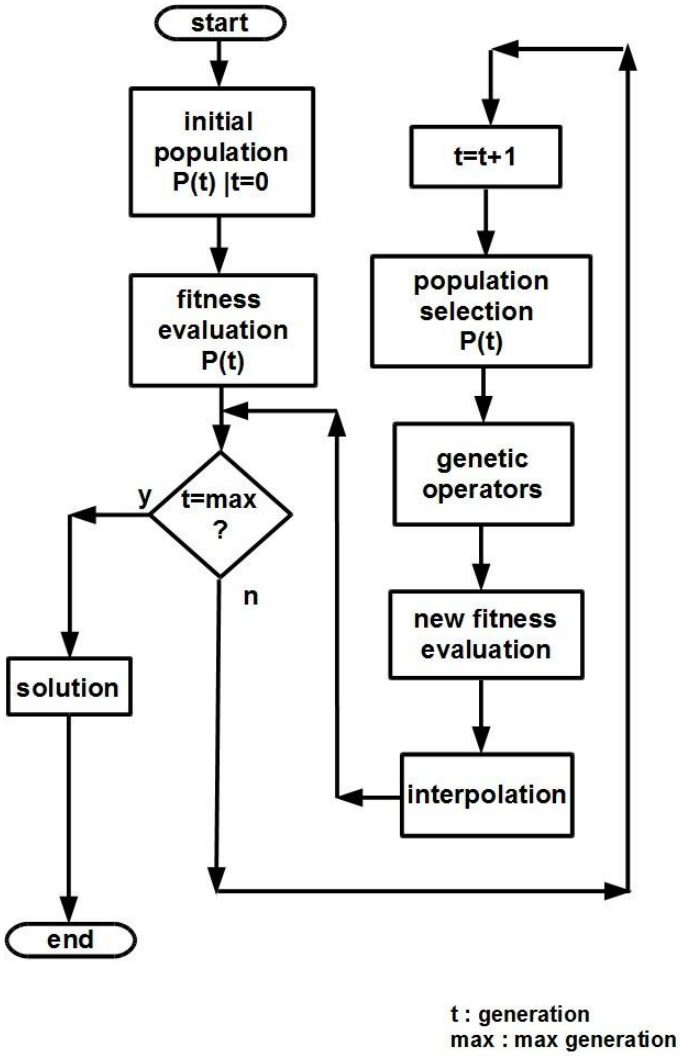


Fig. 2.12 Process flowchart of Genetic Algorithm

To generate good offspring, a good parent selection mechanism is necessary. This is a process used for determining the number of trials for one particular individual used in reproduction. The chance of selecting one chromosome as a parent should be directly proportional to the number of offspring produced [80].

There are three measures of performance of the selection algorithms [80]:

i. Bias: the absolute difference between individuals in actual and expected probability for selection. The best value is zero or zero bias, when an individual's probability equals its expected number of trials
ii. Spread: a range in the possible number of trials that an individual may achieve. The minimum spread is the smallest spread that theoretically permits zero bias.
iii. Efficiency: the overall time complexity of the algorithms.

The selection algorithm should thus be achieving a zero bias while is maintaining a minimum spread and not contributing to an increased time complexity of GA [80].

Many selection techniques employ roulette wheel mechanism (RWM). The basic roulette wheel selection method is a stochastic sampling with replacement (SSR). The segment size and selection probability remain the same throughout the selection phase. SSR tends to give zero bias but potentially inclines to a spread that is unlimited [80].

Stochastic sampling with partial replacement (SSPR) extends upon SSR by resizing a chromosome's segment if it is selected. Each time a chromosome is selected, the size of its segment is reduced by a certain factor. If the segment size becomes negative, then it is set to zero. This provides a reduction of spread but contributing to an increased time complexity [80].

Stochastic universal sampling (SUS) is another single-phase sampling algorithm with minimum spread, zero bias, and the time complexity. SUS uses an N_s equally spaced pointer, where N_s is the number of selection required. The population is shuffled randomly and a single random number in the range $\left[0, \frac{F_{sum}}{N_s}\right]$ is generated, ptr, where F_{sum} is the sum of individual's fitness functions. The N_s individuals are then chosen by generating the N_s pointers spaced by $1, [ptr, ptr + 1, \dots, ptr + N_s + 1]$, and selecting those individuals whose fitness span the positions of the pointers. In addition, as individuals are selected entirely on their position in the population, SUS has zero bias [80].

Crossover mechanism is shown in Fig. 2.13. A crossover point is randomly set. The portions of the two chromosomes beyond this cut-off point to the right are to be exchanged to form the offspring. An operation rate (p_c) with a typical value between 0.6 and 1.0 is normally used as the probability of crossover [80].

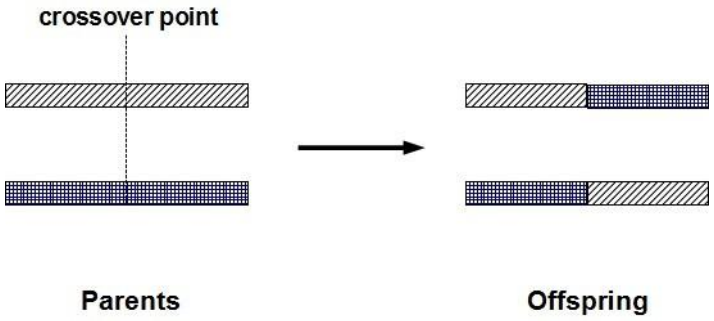


Fig. 2.13 Example of one-point (in the middle) of crossover [80]

Although the one-point crossover method was inspired by biological processes, it has one major drawback in that certain combinations of schema cannot be combined in some situations [81].

The mutation mechanism is shown in Fig. 2.14. This applied to each offspring individually after the crossover exercise. It alters each bit randomly with a small probability (p_m) with a typical value of less then 0.1 [80]. Mutation operation is used for avoiding the premature convergence in optimization process [82].

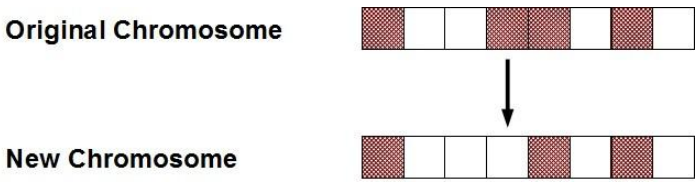


Fig. 2.14 Example of bit mutation on the fourth bit [80]

The choices of p_m and p_c as the control parameters can be a complex nonlinear optimization problem to solve. Furthermore, their setting are critically dependent upon the nature of the objective function. This selection issue still remains open to suggestion although some guidelines have been introduced by Grefenstette in 1986 and Dejong and Spears in 1990 [80]:

- For large population size (100)
  crossover rate: 0.6
  mutation rate: 0.001

- For small population size (30)
  crossover rate: 0.9
  mutation rate: 0.01

2.12.2 Parallel Genetic Algorithm

Brief mention should be made of the GA properties that will be important to this work. One can see that the GA already possesses an intrinsic parallelism architecture, in a nutshell, and hence less effort is required to construct a parallel computational framework. This provide the opportunity for the GA be fully exploited in its parallel structure to gain the required speed for practical use [80].

There are some GA-based parallel methods to enhance the computational speed. The methods of parallelization can be classified as: embarrassingly parallel algorithm [83], global parallel algorithm, migration parallel algorithm, and diffusion parallel algorithm. These categories reflect the different ways in which parallelism can be exploited in the GA as well as the nature of the population structure and recombination mechanisms used. A useful review on these techniques has been given by Man et al. [80].

For example, in embarrassingly parallel algorithm, the same evolutionary algorithm is run under different initial conditions in a parallel way. In this technique, once all the different configurations have been executed, the configuration showing the best behaviour is chosen [83].

Global parallel algorithm on the other hand treats the entire population as a single breeding mechanism. This can be implemented on the shared memory multiprocessor or distributed memory computer. In this case on a shared memory multiprocessor, chromosomes are stored in the shared memory. Each processor accesses the particular assigned chromosome and return the fitness functions without any conflicts. It should be noted that there is some synchronization needed between generation to generation. There is a need for the computational load among the processors to be balanced using a dynamic scheduling algorithm, e.g. guided self-schedule [80].

Another parallel processing mechanism for computing the GA is the migration parallel algorithm. The migration GA (coarse grained parallel GA) divides the population into a number of sub-populations, each of which is treated as a separate breeding unit under the control of conventional GA. The proliferation of good genetic material throughout the whole population is encouraged, and therefore the individual migration between sub-populations occurs from time to time [80].

Apart from the global and migration techniques, diffusion parallel algorithm (fine grained parallel GA) considers the population as a single continuous structure. Here, each individual is assigned to a geographic location on the population surface and usually placed in a 2-D grid, for the reason that the topology of the processing element in many massively parallel computers that are constructed in this form [80].

As noted in [80], the individuals are allowed to breed with individuals contained in a small local neighbourhood. Usually, this neighbourhood is chosen from the immediate adjacent individuals on the population surface and is motivated by the practical communication restrictions of parallel computers.

2.12.3 Hierarchical Genetic Algorithms

The following discussion on hierarchical genetic algorithm (HGA) is referred to the review given by Man et al. in [80]. On the biological inspiration, the genes can be classified into two different types: regulatory sequences (RSs) and structural genes (SGs). The SGs are coded for polypeptides or RNAs, while the RSs serve as the leaders that denote the beginning and ending of SGs, or participate in turning on or off the transcription on SGs, or function as initiation points for replication or recombination [80].

As has been mentioned in [80], one of the most surprising discoveries in the founding of molecular biology was that active and inactive genes exist in the SGs. Here, the active genes are separated into non-contiguous pieces along the parental DNA. Accordingly, the pieces that code mRNA are referred to as exons (active genes) and the non-coding pieces are referred as introns (inactive genes). During a transcription, there is a process of splicing.

On the algorithm application, to indicate the activation of the control gene, an integer "1" is assigned for each control gene that is being ignited where "0" is for turning off. When "1" is signalled, the associated parameter genes due to that particular active control gene are activated in the lower level structure. It should be noticed that the inactive genes always exist within the chromosome even when "0" appears. This hierarchical architecture implies that the chromosome contains more information that of the conventional GA structure. Hence, it is called *hierarchical genetic algorithm (HGA)* [80]. An example of HGA structure with 8-bit control genes and 8-bit parameter genes is shown in Fig. 2.15.

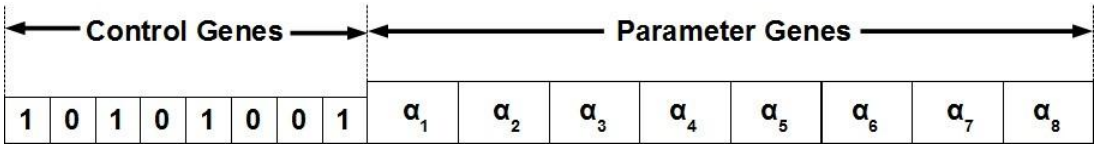


Fig. 2.15 An example of HGA structure with 8-bit control genes and 8-bit parameter genes

2.13 Controller Performance

There are some performance indexes usually used as a controller performance which are able to be classified as two groups: second order underdamped response and error analysis

The performance index in second order underdamped response are specified as follows [54]:

i. *Rise time,* t_r. The time required for the waveform to go from 0.1 of the final value to 0.9 of the final value.
ii. *Peak time,* t_p. The time required to reach the first, or maximum, peak.
iii. *Percent overshoot,* $\%O_S$. The amount that the waveform overshoots the steady state, or final value at the peak time, expressed as a percentage of the steady state value.
iv. *Settling time,* t_s. The time required for the transient's damped oscillations to reach and stay within $\pm$ 2 % of the steady-state value, or $\pm$ 5 % of the steady-state value [55].

The second-order underdamped response specification is shown in Fig. 2.16. Notice that all definitions are also valid for systems of order higher than 2, although analytical expressions for these parameters cannot be found unless the response of the higher-order system can be approximated as a second-order system [54].

There are several kinds of error analysis as a performance index, and the two of them are as follows:

1. Integral of absolute value of error (IAE) is defined as [84]:

$$IAE = \int_0^t |SP(t) - PV(t)|dt \qquad (2 - 39)$$

where SP is set-point and PV is process value.

IAE is appropriate for measuring the performance when the transient process is more important or when the input is variable.

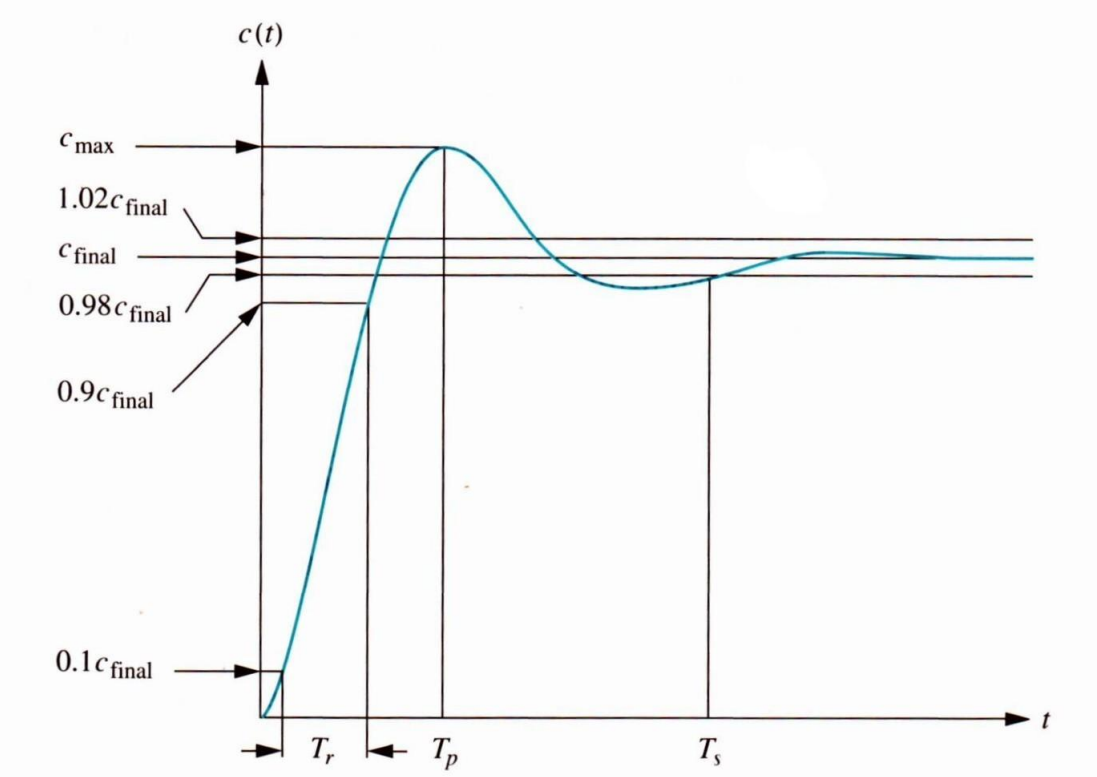


Fig. 2.16 Second-order underdamped response specifications [54]

2. Integral of time absolute value of error (IAE) is defined as [84]:

$$ITAE = t\int_{0}^{t}|SP(t) - PV(t)|dt \qquad (2 - 40)$$

where SP is set-point, PV is process value, and t is time (duration).
$ITAE$ is appropriate for measuring the performance when the steady process is important besides the transient process and when the input is constant.

2.14 Summary

The overview of some related previous work and the basic theory of electrical motor, system identification, control, fuzzy, and genetic algorithm has been presented in this chapter. The proposed method on the optimization algorithm has been introduced.

The optimization algorithm proposed in this chapter offers some promising ways for achieving the best controller for a DC servomotor.

In the following chapter, detailed discussion on the model development, the controller's design and the optimization algorithm development, and the verification of the controllers and the optimization algorithm via simulation to determine the correctness of the model, the controllers and the optimization algorithm will be presented.

CHAPTER 3

SIMULATION AND HARDWARE EXPERIMENT

3.1 Introduction

This chapter presents the methodology devised for the experiments setup. It starts with the modelling of a DC motor from a known source with known specifications. The associated controllers were then designed and applied to the model using MATLAB/SIMULINK. There were six speed controllers: (1) proportional-integral (PI), (2) proportional-integral-derivative (PID), (3) fuzzy logic controller (FLC), (4) fuzzy logic-based self-tuning PI (FLBPI), (5) fuzzy logic-based self-tuning PID (FLBPID), (6) and fuzzy logic parallel integral controller (FLIC). There were three position controllers: (1) fixed proportional controller, (2) FLC, and (3) variable proportional controller. The performance of the controllers were compared and the controllers in which the performance are better than PI and PID controllers would be selected to be optimized using GA.

It would be time consuming and risky if the GA process is applied to the real hardware. Therefore, from the hardware servomotor system, a gray box *s*-modeling of the system was first build. The controllers were then applied to the transfer function model for evaluation. Once the controller's performances are understood, then a real-time implementation on an experiment rig that constitutes the servo speed and position controller system consisting of the servomotor and load, measuring and controlling devices would be conducted.

The next step is designing GA to optimize the speed controllers. Before optimizing the controllers, the GA simulation is conducted to get the ideal specifications. In the GA simulation, it is designed to get the maximum value of the function. Based on the simulation result, a new method of GA is proposed, i.e. the

semi-parallel operation GA (SPOGA) with the change of the operation process. The initial population and solution processes are also changed.

The SPOGA is then applied to the transfer function and hardware experimental rig. The performance of SPOGA optimized controllers are compared to non-SPOGA optimized controllers and conventional controllers.

3.2 Hardware Implementation

The experiment on DC servomotor for the speed and position was conducted using SIMULINK where the block diagram is shown in Fig. 3.1. The speed control loop is in the position control loop [28]. Basically, the position control loop is executed until reaching the position set-point while the speed is limited to the speed set-point. Practically, position control is better to be sensorless because it will reduce cost and size and increase the reliability of the overall system [85].

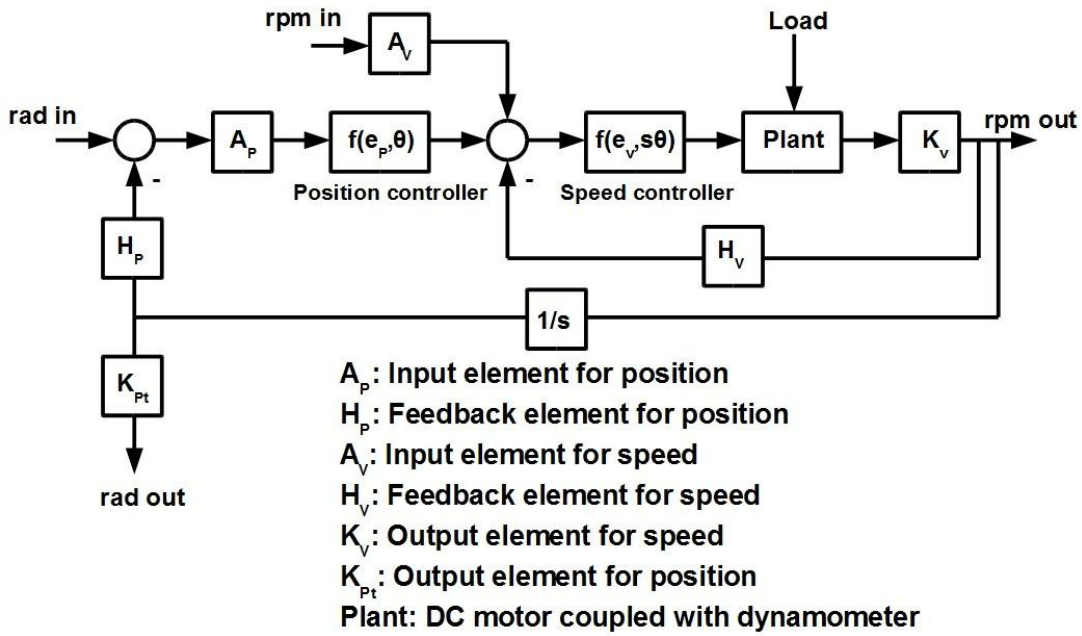


Fig. 3.1 Structure of feedback controller [28]

The input, feedback, and output elements for position and speed are implemented in the SIMULINK diagram with specifications as follows:

A_P=1

A_v=0.002

K_v=9.5455

Hv=0.002

H_P=0.005

K_{pt}=0.005

These specifications have been based on realistic assumptions.

Position control was done based on the position set-point and integration of speed sensor as a process value. The position controller was designed to control the radian position where the maximum position is less then 2π or 6.28 rad. In the experiment, the range of position is [0.00, 6.00] rad. Based on the experiment on the hardware device, the range of speed is [150.00, 426.47] rpm when no load and in the full load condition (1 Nm), the range of speed is [150.00, 278.29] rpm. Hopefully, the position will reach the maximum position set-point with the minimum of speed set-point within 90 sec. Based on the gear specification in the RS Component web, there are three reduction gears with ratio 12/70 each. Therefore, the overall gear ratio is 0.005 and as depicted in Fig. 3.1, H_P would be assigned the value 0.005, and similarity for K_{pt}.

The input range of position controller is [0.00, 6.00]. The range of position set-point is [0.00, 6.00] rad. Therefore, the value of A_P is unity. The input range of speed controller is [0.00, 10.00]. The speed sensor has 0.002 volts/rpm. The range of speed set-point in the experiment is [0.00, 400.00] rpm. Therefore, the value of A_V is 0.002 and it needs conditioning of input constant. The value of H_V is the same as A_V to compare the values of set-point and process value that yields an actuating signal.

The hardware implementation for the block diagram is as follows:

Plant/DC motor: 175 W, 1500 rpm, 240 V, 1.1 A

Load : Dynamometer load controller

<u>Input elements:</u>

Tacho-generator with 500 rpm/volt

ADC: 1 channel [0, 10] volts

Filtering : FIR with 30 points

Output elements:

DAC: 2 channels [0, 4] volts

Differential amp: HA-17741

Power amp: Chopper/Inverter and IGBT

Controller elements:

Computer with Intel Pentium Dual Core T2080 processor, Windows XP SP3, MATLAB/SIMULINK software.

The DAQ USB-1208FS has 250 samples per second and it depends on a PC that is used. Using computer and software as specified above, it has 1000 samples per second. In other words, the sampling rate is 1 ms.

It is to note that at this point, from the experiment the appropriate sampling period for the control algorithms with regards to the time constant of the plant in the open loop condition is 10 ms. The more detail discussion on the experiment will be presented in Chapter 5.

The experimental rig constituting the servo speed and position controller system consists of the servomotor and load, measuring and controlling devices. The servo system contains a DC motor driven by an IGBT chopper inverter. The measuring device is the speed sensor (tacho-generator), ADC and a digital filter i.e., finite impulse response (FIR), while the controlling devices are DAC, differential amplifier, and the IGBT inverter circuit. The measuring devices provide status of the output responses of the speed and position where the information about the speed and position is fed through signal conditioning circuit and anti-aliasing filter for analysis and calculation of the control signal. The speed and position requirements proportional to the manipulated variable of the controller's output are fed to a computer.

The block diagram of hardware design is shown in Fig. 3.2 where the power amplifier and differential amplifier are presented more detail in Fig. 3.3 and Fig. 3.4.

The output range of controller is [0.00, 20.00] volts. This follows the input range of Chopper/Inverter Control Unit in the power amplifier circuit diagram (Fig. 3.3)

where the input range is [-10.00, +10.00] volts. This range is fed by the output of differential amplifier in Fig. 3.4 which has a formula [86]:

$$V_{OUT} = \left(\frac{R_2}{R_1}\right)(V_2 - V_1) \qquad (3-1)$$

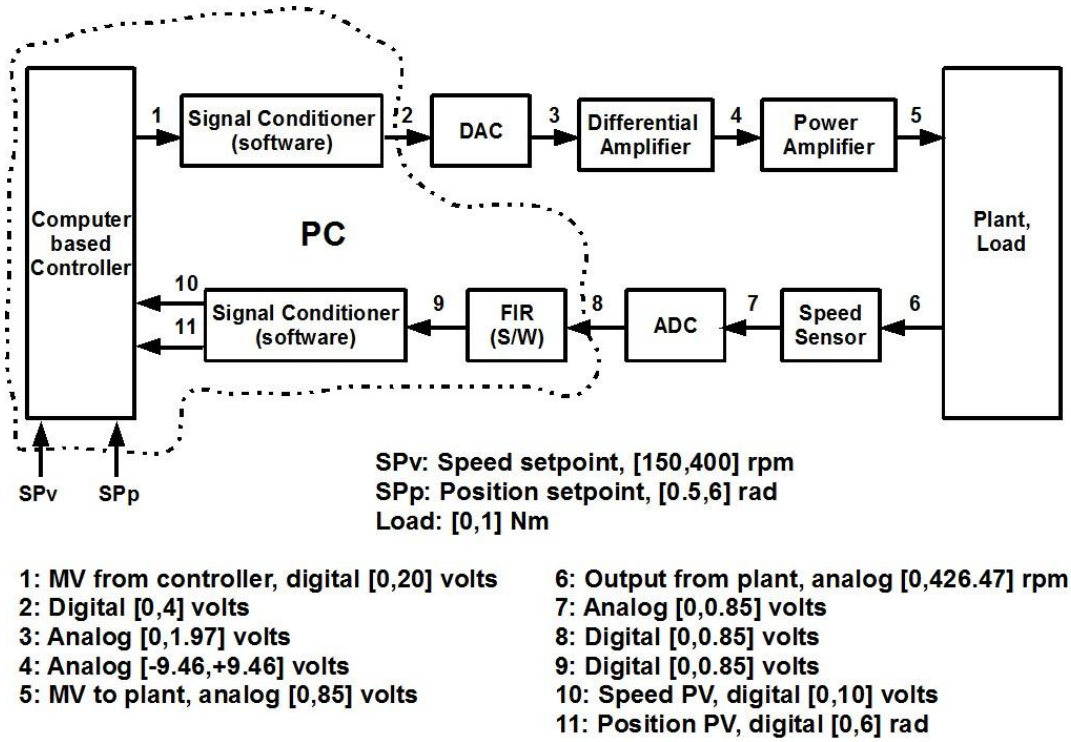


Fig. 3.2 Block diagram of hardware design

The gain of differential amplifier is chosen to be 5 (five) regarding to the output range of DAC which is about 2 volts. The values of V_1 and V_2 are fed from a DAC which has two channels 12 bit each. The block diagram of USB-1208FS DAQ is shown in Fig. 3.5. The output of ADC or input of DAC are connected to the computer using full speed USB 2.0 compliant interface which is also compatible with USB 1.1.

Signal conditioner software is needed in communication between computer and DAC. Refer to the Fig. 3.6 and Fig. 3.7, if **in** is the output of computer as a controller to the signal conditioner, and **v1** and **v2** are the output of signal conditioner to DAC, then the flowchart is shown in Fig. 3.6.

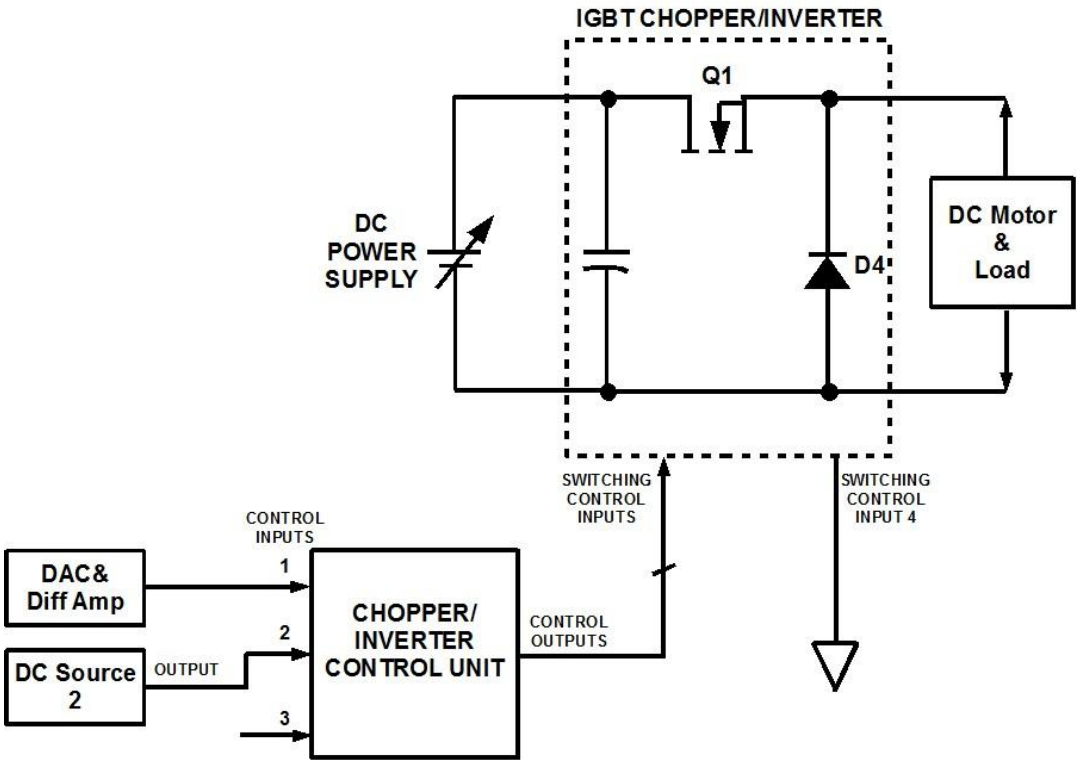


Fig. 3.3 Power amplifier circuit diagram

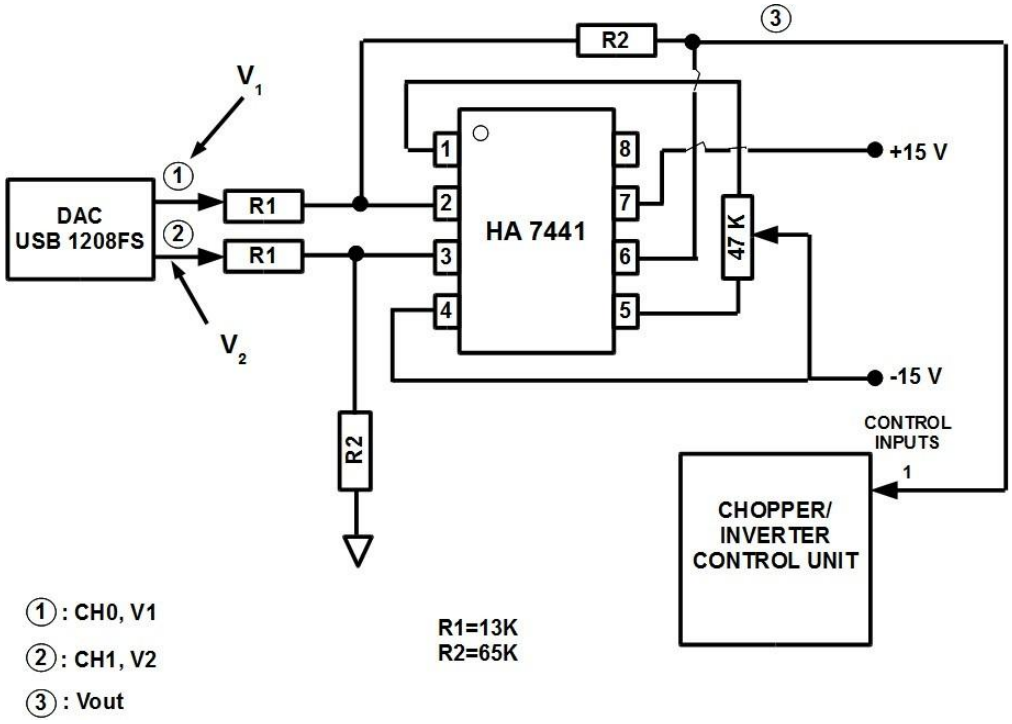


Fig. 3.4 Differential amplifier circuit diagram

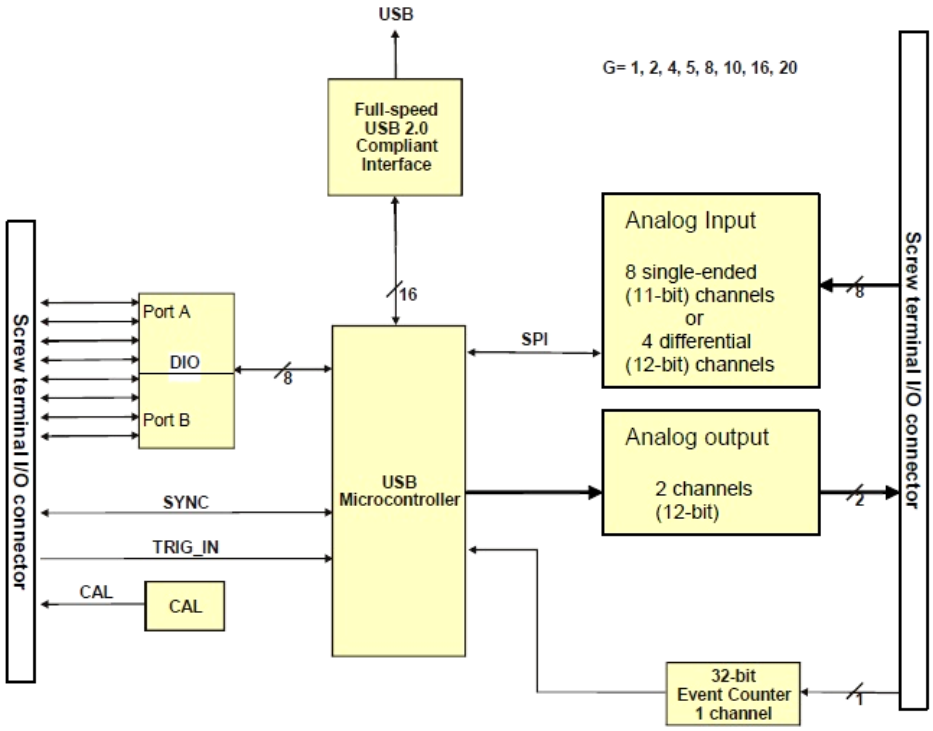


Fig. 3.5 USB-1208FS functional block diagram [87]

As seen in Fig. 3.6, with the range of **in** is [0.00, 10.00], the range of **v1** will be [4.00, 0.00] and **v2** will be zero. With the range of **in** is [10.00, 20.00], the range of **v2** will be [0.00, 4.00] and **v1** will be zero as shown in Fig. 3.7.

The output of speed sensor has the range of [0.00, 0.85] volts and can directly connected to the ADC which is within the range input of ADC, [0.00, 10.00] volts. The output of ADC is digital 0.85 and filtered using 30-point FIR for noise reduction. It is a digital software filter based on moving average (MA) filter. If N is the number of points, X_i is input at i^{th} point, then the output at i^{th} point, Y_i, is given by

$$Y_i = \frac{\sum_{k=0}^{N-1} X_{i-k}}{N} \qquad (3-2)$$

Experimentally, the value of N is 30. This was done by searching the minimum value of N in the multiples of 5 in which the maximum deviation of noise is less than the maximum steady state error, i.e. 2 % [54]. The experiment result will be presented in Chapter 5.

There are two output transducers: speed output transducer, and position transducer. There are two speed transducers sequentially:

i. Zero tolerance regarding to noise: when the magnitude is less than or equal to 0.05 volts, then it is assumed to be zero
ii. Converter volt to rpm: regarding to the speed sensor, the magnitude in volt is converted to speed by multiplying with 500.

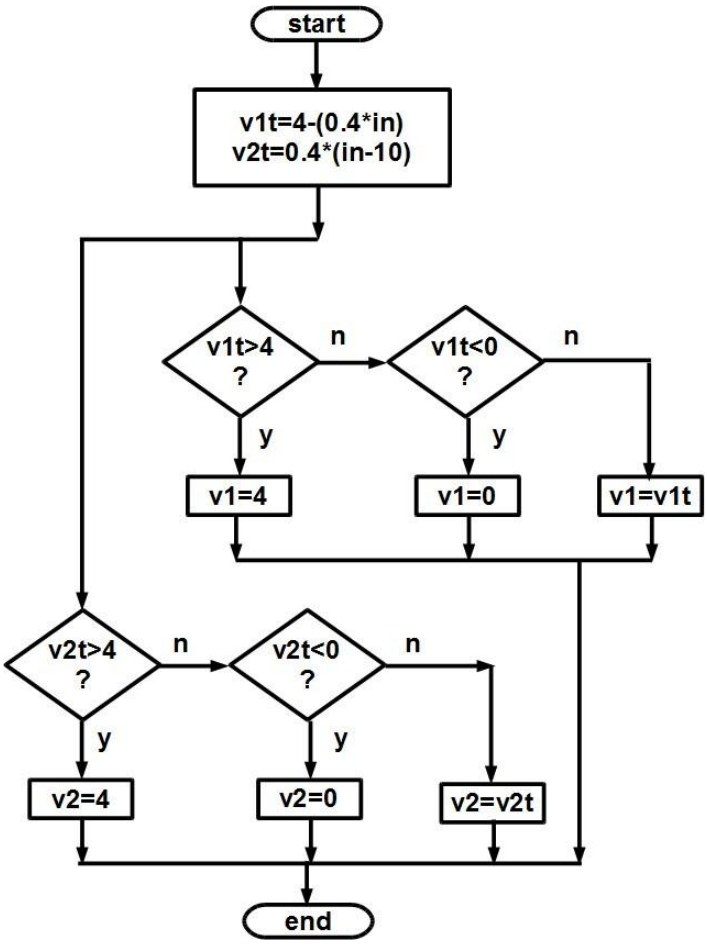


Fig. 3.6 Signal conditioner flowchart for DAC

The position transducer converts the rpm to rad/sec by multipying with 0.1048 and then multipying with 0.005 as a gear ratio. The final value is then integrated to convert the speed to position.

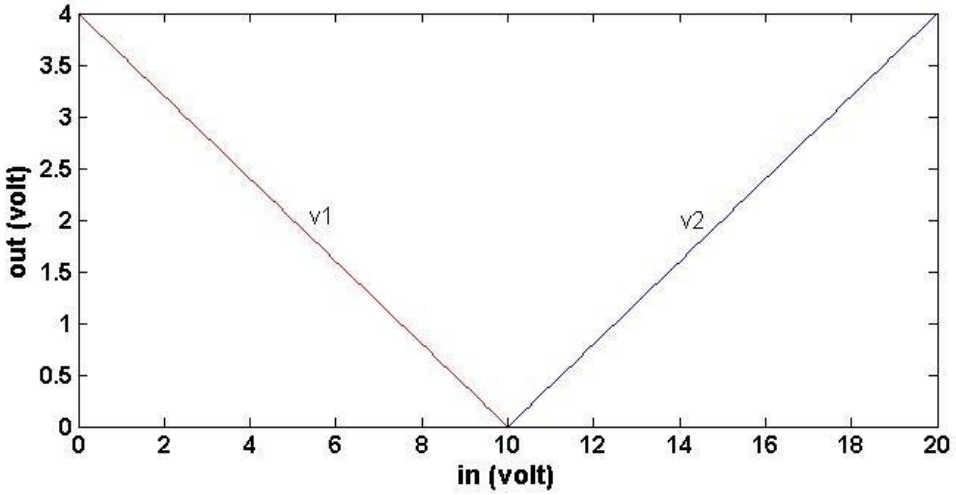


Fig. 3.7 Input-output of signal conditioner

3.3 Input-Output Modelling of A DC Servomotor

In the optimization process of a servomotor, it requires time and has big risk. To investigate this issue, a real plant, constituting of a DC motor and its controller is modeled and simulated to allow detail analysis of its control system. The flowchart of input-output modeling process is shown in Fig. 3.8.

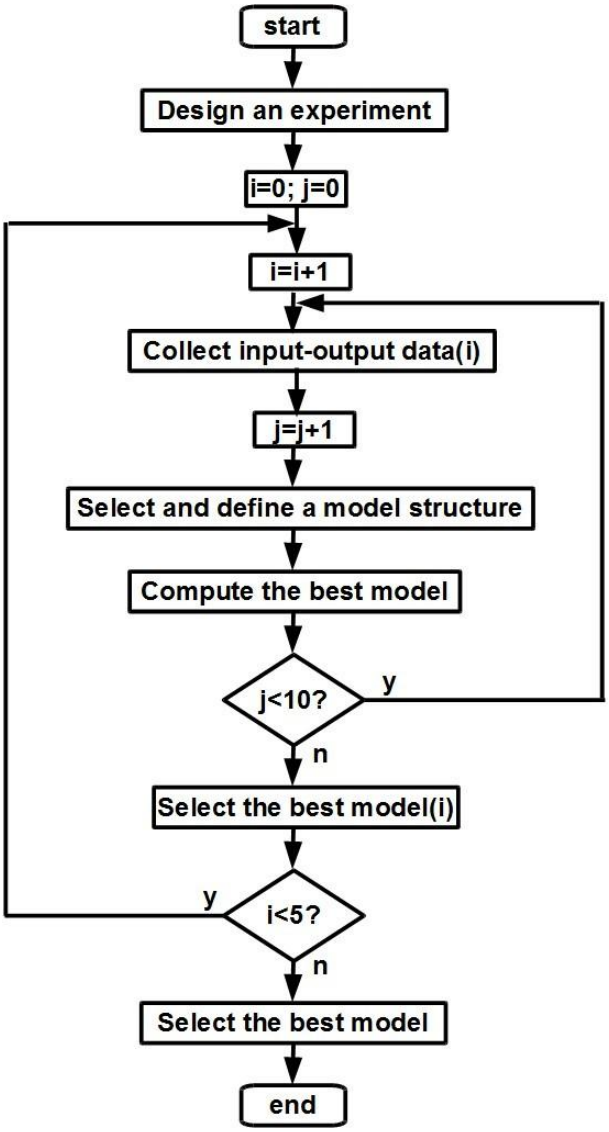


Fig. 3.8 Process of input-output modeling

3.3.1 Designing An Experiment

The DC motor used in experiment is rated 175 W, 1,500 rpm, 240 V, 1.1 A. The DC motor is coupled with a dynamometer. These components are combined as a gray box to be identified as a transfer function in *s*-domain.

Open loop characteristic of the gray box is tested using SIMULINK which is applied to the plant through DAQ, 30-point-FIR filter, chopper/inverter control unit, and IGBT. The input voltage is varied randomly in the range of [0.00, 20.00] volts for 100 seconds with about 32 sequences. There are five patterns of input signal shown in

Fig. 3.9 to Fig. 3.13. These patterns are generated randomly using SIMULINK/MATLAB command.

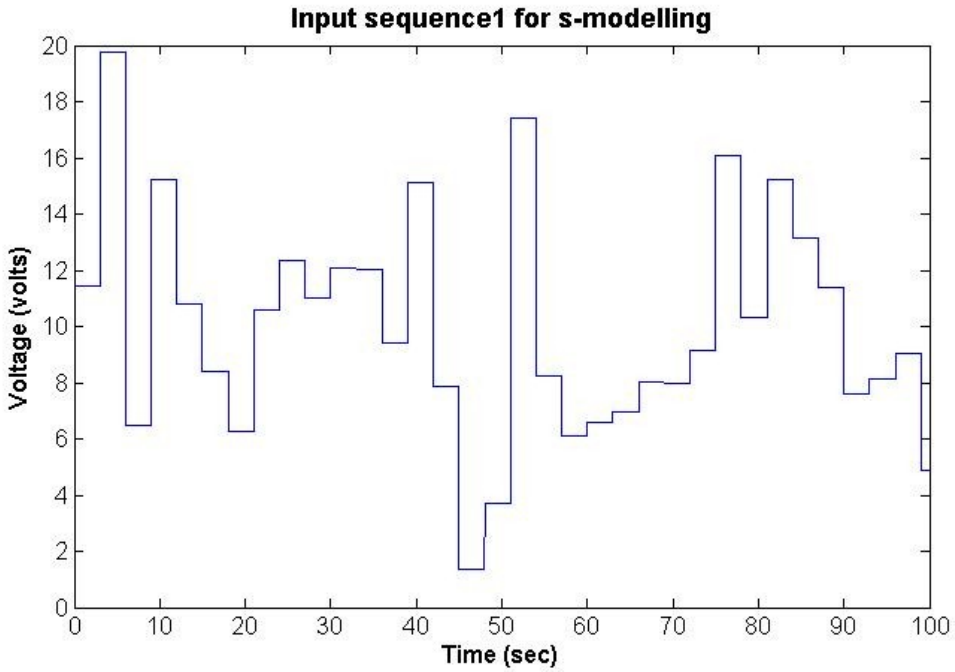


Fig. 3.9 Input sequence 1 for input-output modelling

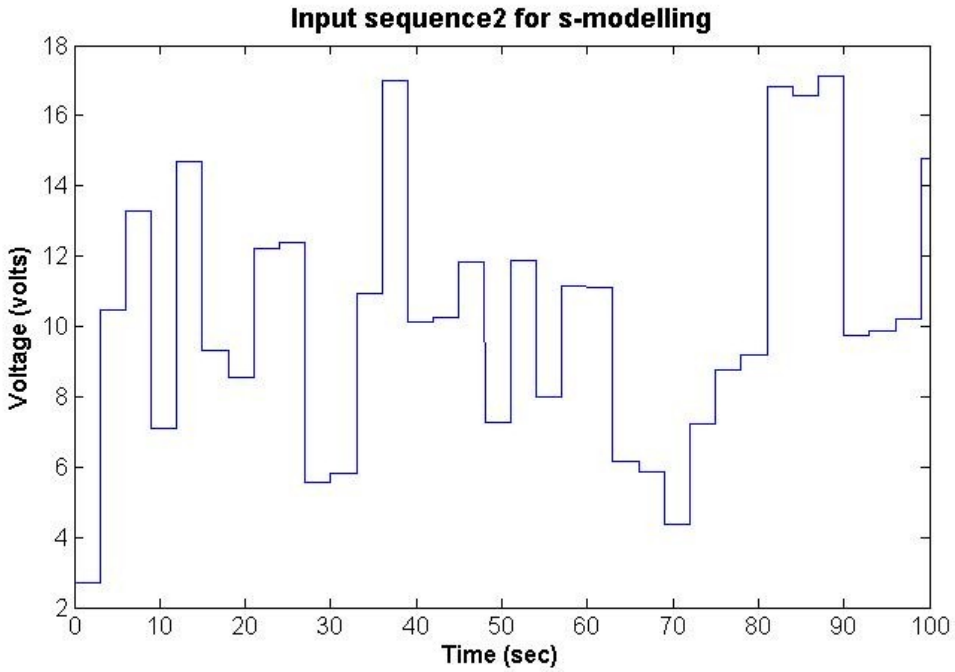


Fig. 3.10 Input sequence 2 for input-output modelling

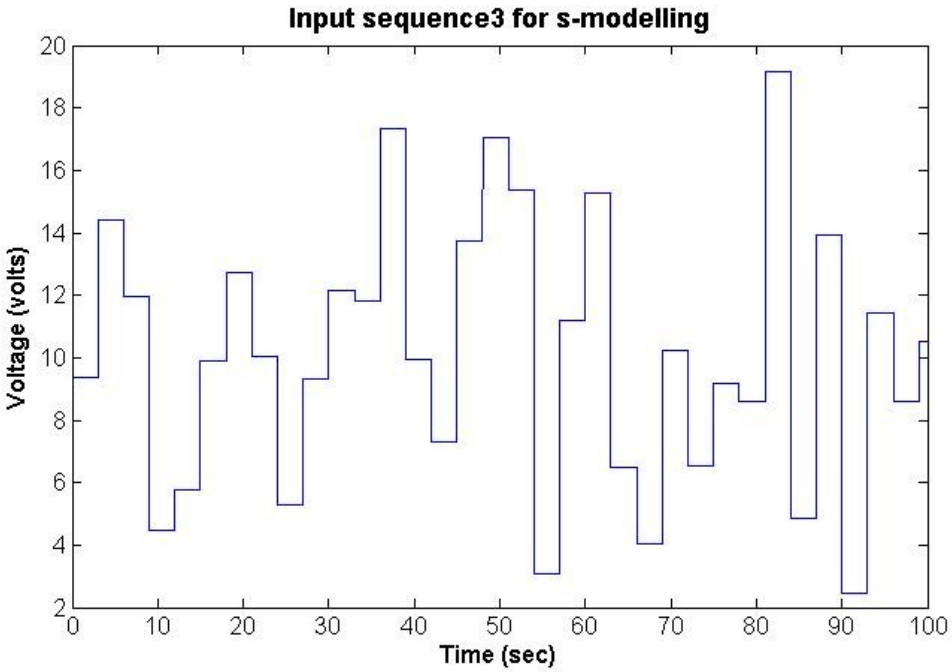


Fig. 3.11 Input sequence 3 for input-output modelling

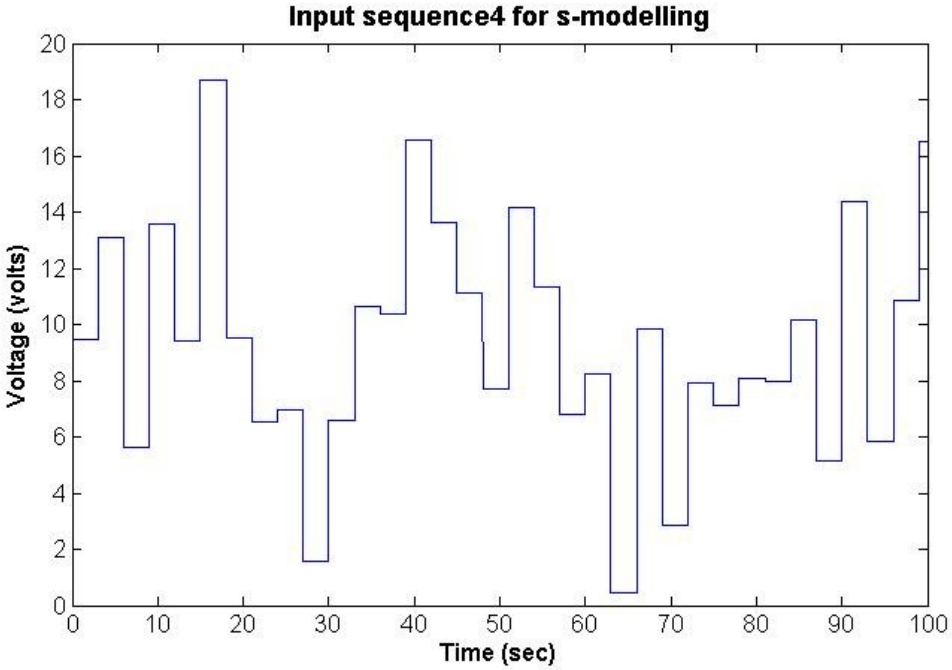


Fig. 3.12 Input sequence 4 for input-output modelling

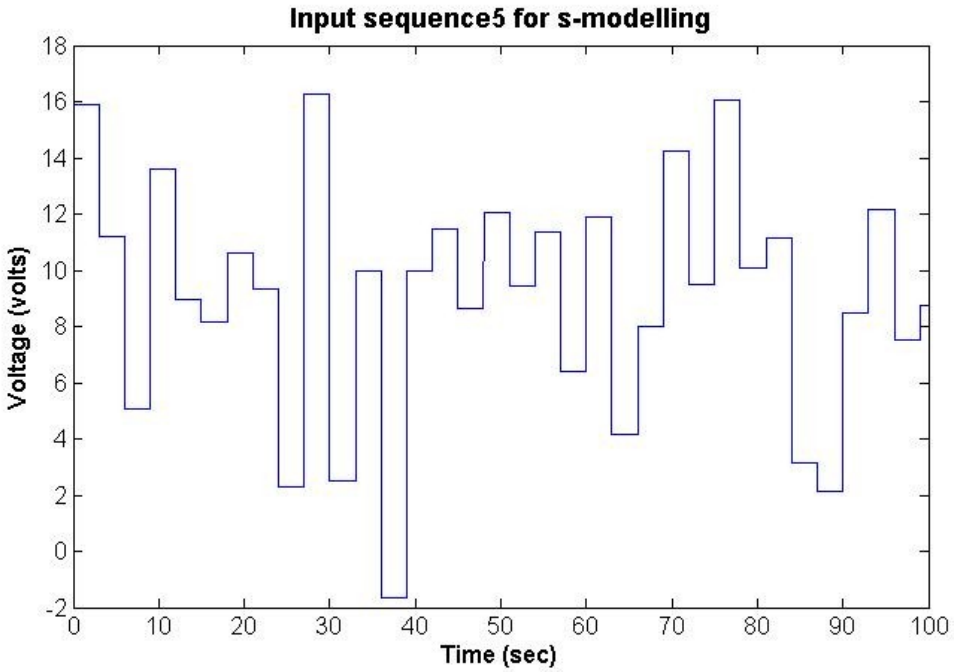


Fig. 3.13 Input sequence 5 for input-output modelling

3.3.2 Collecting Input-Output Data

The DC motor is run for 100 sec with variations of input voltage in Fig. 3.9 to Fig. 3.13. The sampling time is 0.01 sec. The result (output voltage) and the input are then saved in the workspace. The data for input-output modeling is obtained from the MATLAB command below:

```
data=iddata(workspace_input,workspace_output,0.01);
```

3.3.3 Selecting and Defining A Model Structure

There are six process models to be compared and selected the best one: process model using 1st order transfer function without zero, process model using 1st order transfer function with zero, process model using 2nd order transfer function without zero, process model using 2nd order transfer function with zero, process model using 3rd

order transfer function without zero, and process model using 3rd order transfer function with zero.

If P is the process model, n is the number of poles (1,2 or 3), D is to include a time-delay term (optional), Z is to include a process zero (optional) and U is to indicate possible complex-valued (underdamped) poles (optional), then using MATLAB command, the general command to get the process model is as follows:

model=pem(data, 'PnD$_t$ZU$_c$');

3.3.4 Computing The Best Model

To select the best model, both the real time system and the six type input-output model systems are fed with the same input pattern and the output are compared. An input-output model system type with the best fitness (i.e. the largest fitness value) is selected and the fitness value is obtained from Eq (2 - 22).

3.3.5 Selection of The Best Model

To select the best model, the previous steps are repeated 10 times for each type of input sequence, and the result are ten *s*-models with the best fitness. The best model is selected from the best *s*-model among the ten best *s*-models. These steps are then repeated for five different types of input sequence. The best model is selected among the five best *s*-models.

3.4 Simulation and Experiment Design of Speed and Position Control

The speed and position control was build based on the structure of feedback controller in Fig. 2.7. The block diagram of simulation experiment in SIMULINK is shown in Fig. 3.14, and for hardware experiment is shown in Fig. 3.15

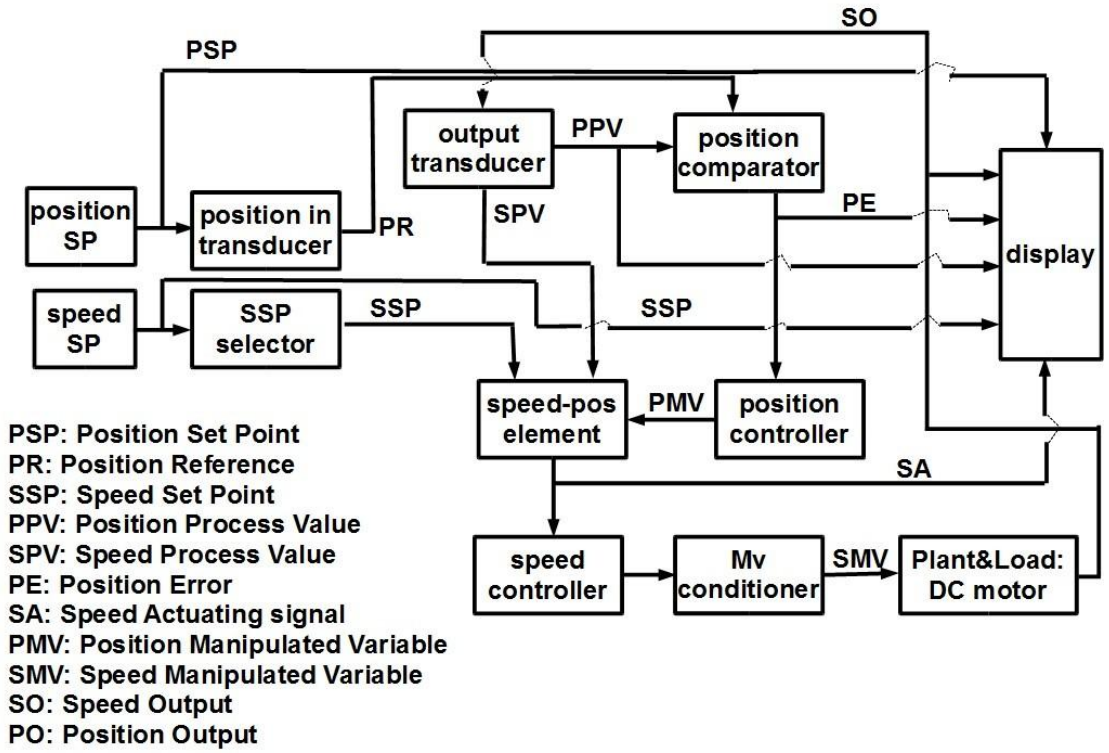


Fig. 3.14 Block diagram of simulation experiment in SIMULINK platform

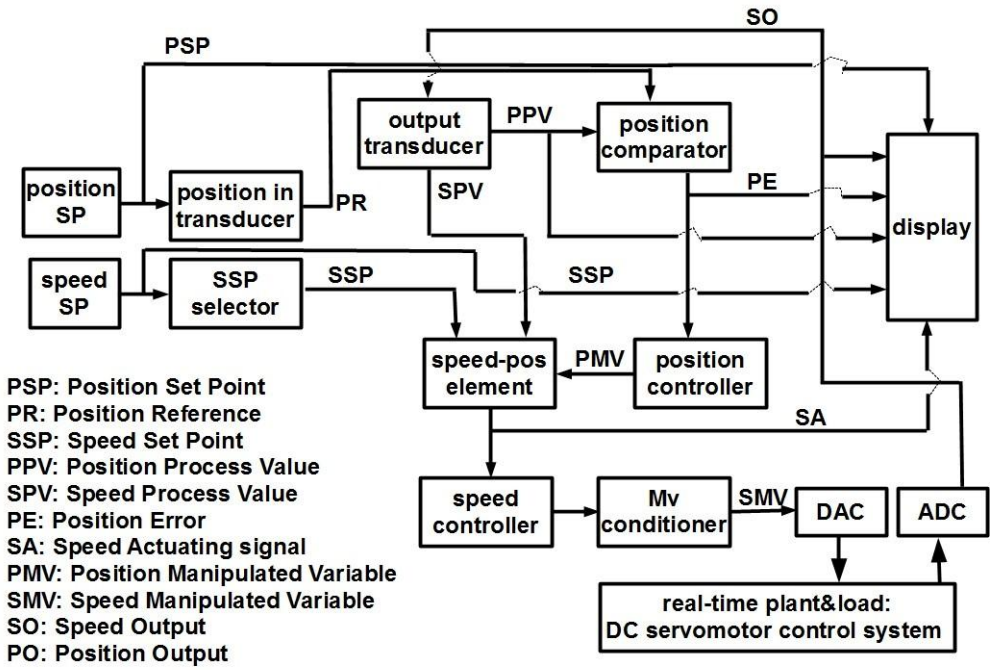


Fig. 3.15 Block diagram of hardware experiment in SIMULINK platform

The SSP selector is used for selecting the two types of input: constant input according to the value given by user, and variations of input with the range of

[0.00, 400.00] rpm for the experiment with variations of speed set-point. The pattern of speed set-point for variations of input is shown in Fig. 3.16

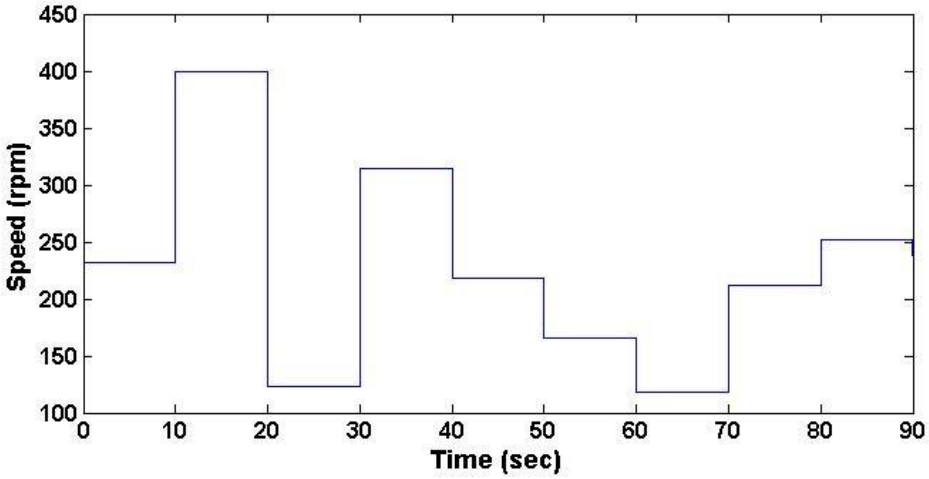


Fig. 3.16 Pattern of variations of speed set-point

The speed-pos element is used for : conditioning the actual speed set-point in the speed controller based on input speed set-point and position manipulated variable, speed comparator, i.e. comparison between speed set-point and speed process value, and conditioning the input constant of controller.

In the first function of speed-pos element, if ASSP is actual speed set-point, ISSP is input speed set-point (set by user) and PMV is position manipulated variable, then the flowchart is shown in Fig. 3.17.

In the second function of speed-pos element, if A_{SSP} is actual speed set-point and S_{PV} is speed process value, then the output of speed comparator, S_{EC}, is formulated in Eq. (3 - 3)

$$S_{EC} = A_{SSP} - S_{PV} \qquad (3 - 3)$$

In the third function of speed-pos element, if A_{SSP} is actual set-point and S_{EC} is output of speed comparator, then the input to the speed controller, S_E , is formulated in Eq. (3 - 4)

$$S_E = \frac{10 * S_{EC}}{A_{SSP}} \qquad (3 - 4)$$

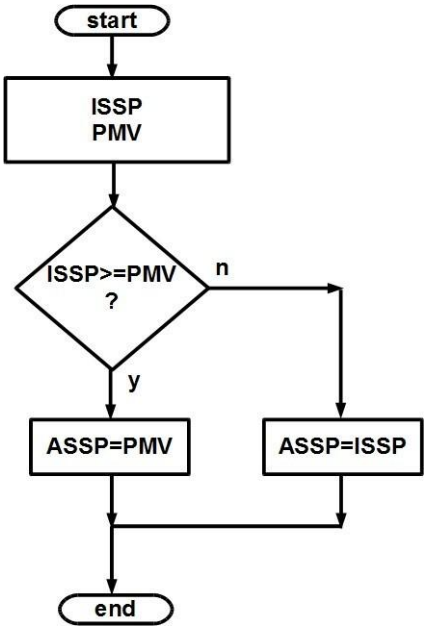


Fig. 3.17 Flowchart of conditioner of actual speed set-point

Position comparator is used for comparison between position set-point and position process value. If P_{SP} is position set-point and P_{PV} is position process value, then the output of position comparator, P_E, is formulated in Eq. (3 - 5)

$$P_E = P_{SP} - P_{PV} \qquad (3 - 5)$$

The output transducer contains filtering (FIR) and controller conditioning. The USB-1208FS was used as an ADC and DAC. In the simulation experiment, the Mv conditioner has the value of unity. In the hardware experiment, the Mv conditioner is used as a conditioner to the DAC as shown in Fig. 3.6 and Fig. 3.7.

The position controller is used for controlling the position in one direction only. Therefore, the appropriate controller is proportional based. There were three kinds of position controller in the experiment:

i. Fixed proportional controller
ii. Variable proportional controller
iii. Fuzzy logic controller

The speed controller is used for controlling the speed in one direction only. There were nine kinds of speed controller in the experiment:

i. PI controller
ii. PID controller
iii. FLC (Fuzzy Logic Controller)
iv. FLBPI (Fuzzy Logic Based self tuning PI) controller
v. FLBPI-GA (Fuzzy Logic Based self tuning PI that optimized by GA/SPOGA) controller
vi. FLBPID (Fuzzy Logic Based self tuning PID) controller
vii. FLBPID-GA (Fuzzy Logic Based self tuning PID that optimized by GA/SPOGA) controller
viii. FLIC (Fuzzy Logic parallel Integral Controller)
ix. FLIC-GA (Fuzzy Logic parallel Integral Controller that optimized by GA/SPOGA).

3.4.1 Simulation and Experiment Design of Position Controller

The three kinds of position controller were implemented based on the type of speed controller. The fixed proportional position controller was conducted with conventional speed controllers. The variable proportional position controller in which the idea is based on [27] was done with hybrid-fuzzy speed controllers. The fuzzy logic position controller was conducted with a fuzzy logic speed controller.

Similarly, the parameter of fixed proportional position controller (proportional constant, K_{Pp}), was optimized via experiment method. It was conducted after optimizing each type of the conventional speed controllers.

The parameter of variable proportional controller (proportional constant, K_{Pp}) was optimized using experiment method. It was conducted after optimizing each type of the hybrid-fuzzy speed controllers. If M_p is the output of variable proportional controller, K_{Pp} is proportional constant of position controller, then the output of position controller, P_{MV}, can be formulated as follows:

$$P_{MV} = K_{Pp} * M_p \qquad (3 - 6)$$

If P_E is position error, and K_{Pv} is variable proportional constant, then M_p can be formulated as

$$M_p = K_{Pv} * P_E \qquad (3-7)$$

If P_R is position reference, then the values of K_{Pv} are shown in Table 3.1.

Table 3.1 Values of K_{Pv}

P_R	K_{PV}
[0.5 , 1.0)	0.08
[1.0 , 1.5)	0.25
[1.5 , 2.5]	0.50
(2.5 , 3.5]	1.00
(3.5 , 6.0]	1.10

In FLC, as a fuzzifier is the singleton mode (Eq. (2 - 35)) with two inputs, namely, error and change of error, seven triangular membership functions each, and one output with four triangular membership functions. For the choice based on suitability and practicality, as an inference engine is the Mamdani product (Eq. (2 - 36)) and as a defuzzifier is the center average (eq. (2 - 38)) . The input membership functions and output membership functions are shown in Fig. 3.18 and Fig. 3.19 respectively.

If E, D, and U are error input, change of error input, and output of FLC for position controller, then the rule table formulated is shown in Table 3.2.

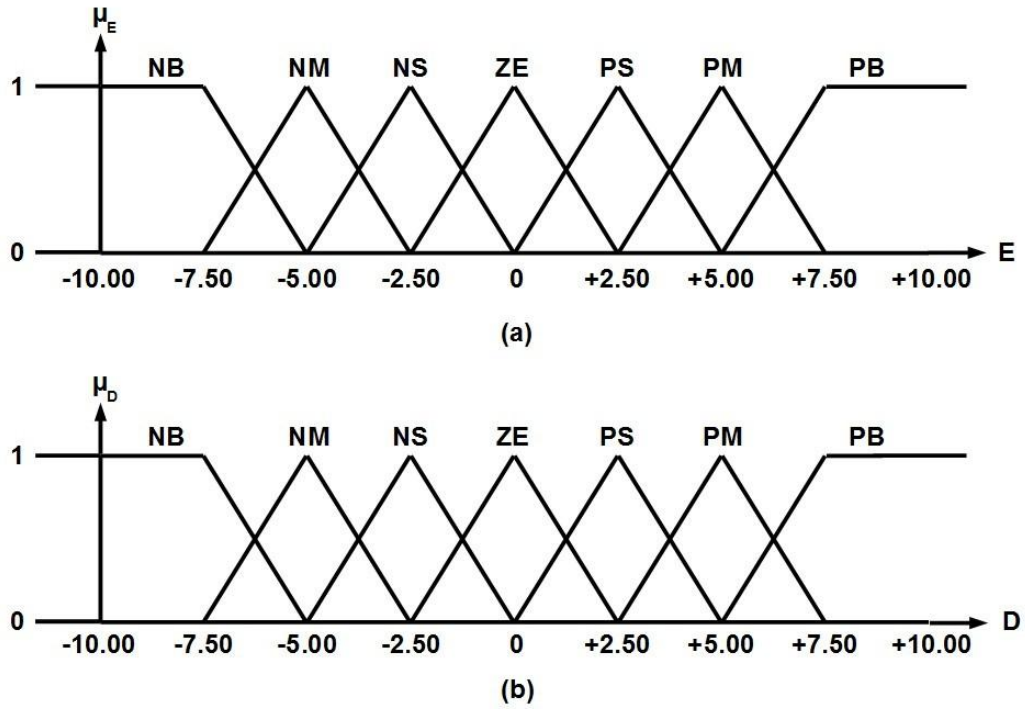


Fig. 3.18 Fuzzy input membership functions for position controller: (a) error; (b) change of error

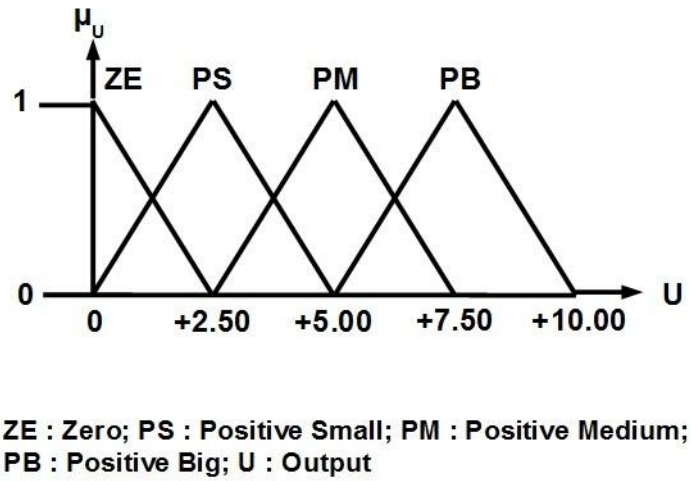


Fig. 3.19 Fuzzy output membership function for position controller

Table 3.2 Rules of FLC for position controller

E	**D**						
	NB	**NM**	**NS**	**ZE**	**PS**	**PM**	**PB**
NB	ZE	ZE	ZE	ZE	ZE	ZE	ZE
NM	ZE	ZE	ZE	ZE	ZE	ZE	ZE
NS	ZE	ZE	ZE	ZE	ZE	ZE	ZE
ZE	ZE	ZE	ZE	ZE	PS	PS	PS
PS	PS	PS	PS	PS	PM	PM	PM
PM	PS	PM	PM	PM	PB	PB	PB
PB	PS	PM	PB	PB	PB	PB	PB

NB: Negative Big; NM: Negative Medium; NS: Negative Small; ZE: Zero
PS: Positive Small; PM: Positive Medium; PB: Positive Big

3.4.2 Simulation and Experiment Design of Conventional Speed Controllers

There were two kinds of conventional speed controller in the experiment:

i. PI controller
ii. PID controller

In PI controller experiment, the parallel combination of PI controller was used which has the basic formulation in time domain as in Eq. (2 - 23), without the derivative part. In discrete form with sampling period of 0.01 sec, Eq. (2 - 26), to Eq. (2 - 28) are modified to be as follows:

$$m(0.01k) = m_P(0.01k) + m_I(0.01k) \quad (3 - 8)$$

$$m_P(0.01k) = K_P e(0.01k) \quad (3 - 9)$$

$$m_I(0.01k) = 0.01K_I \sum_k e(0.01k) + m_I(0) \quad (3 - 10)$$

In the experiment on PID controller, the parallel combination PID controller was used which has the basic formulation in time domain as in Eq. (2 - 23). In discrete form with sampling period of 0.01 sec, Eq. (2 - 26) to Eq. (2 - 29) are modified to be as follows:

$$m(0.01k) = m_P(0.01k) + m_I(0.01k) + m_D(0.01k) \quad (3 - 11)$$

$$m_P(0.01k) = K_P e(0.01k) \quad (3 - 12)$$

$$m_I(0.01k) = \left(0.01K_I \sum_k e(0.01k)\right) + m_I(0) \quad (3 - 13)$$

$$m_D(0.01k) = \frac{K_D}{0.01}[e(0.01k) - e(0.01k - 0.01)] \quad (3 - 14)$$

In SIMULINK platform, there is a facility to use continuous programming and the software can automatically does the conversion from discrete or digital form.

The parameters of PI controller was optimized using Ziegler-Nichols (ultimate cycle) method using Eq. (2 - 31), and the parameters of PID controller was optimized using Eq. (2 - 32).

Both PI and PID experiments were provided with anti integral windup using saturation feedback as in Fig. 2.8. Experimentally, the value of *Ka* is $\frac{100}{K_I}$.

3.4.3 Simulation and Experiment Design of Fuzzy Logic Controller (FLC)

In the experiment of speed controller using FLC, as the fuzzifier is a singleton mode, Eq. (2 - 35), with two inputs, namely, error and change of error, seven triangular membership functions each, and one output rate with seven triangular membership functions. As an inference engine is the Mamdani product, Eq. (2 - 36), and as a defuzzifier is the center average, Eq. (2 - 38). The input and output membership functions are shown in Fig. 3.20 and Fig. 3.21 respectively.

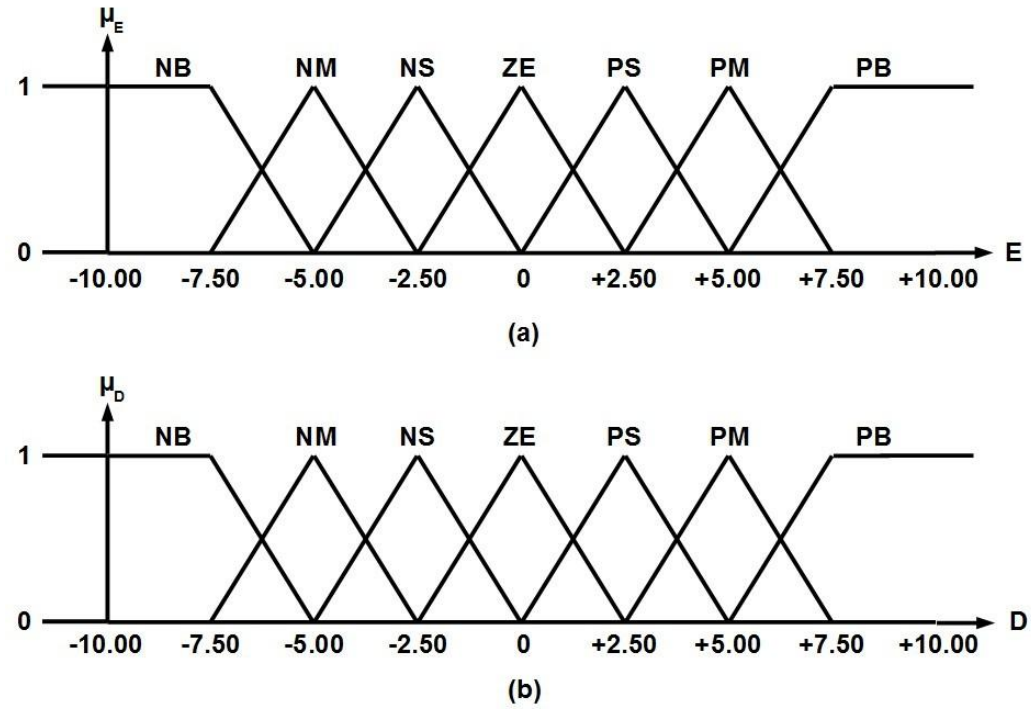


Fig. 3.20 Fuzzy input membership functions for speed controller: (a) error; (b) change of error

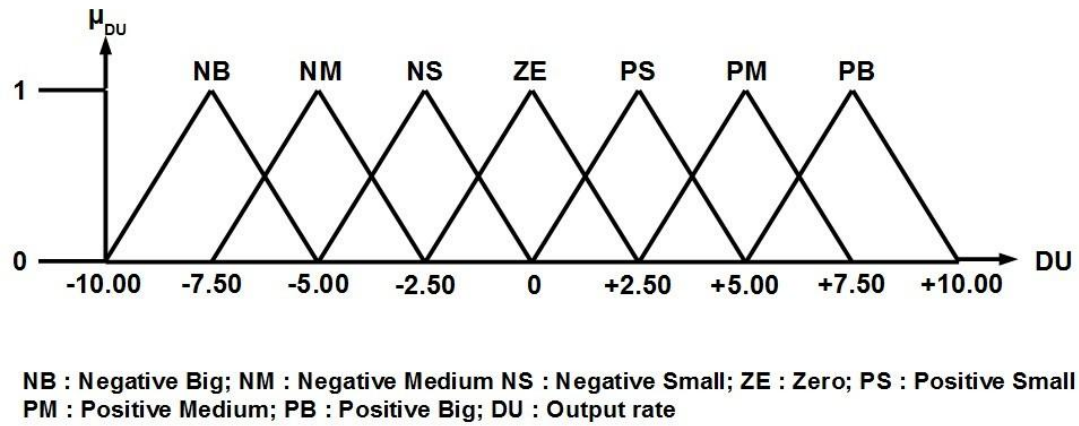


Fig. 3.21 Fuzzy output rate membership function for speed controller

If E, D, and U are error input, change of error input, and output rate of FLC for speed controller, then the rule table in Table 3.3 can be coded into the fuzzy rule string $H_{(w,x,y)}$ which is formulated in the form of integer matrix [43],

$$H_{(w,x,y)} = \begin{bmatrix} h_{1,1} & \cdot & \cdot & \cdot & h_{1,j} \\ \cdot & \cdot & & & \cdot \\ \cdot & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ h_{i,1} & \cdot & \cdot & \cdot & h_{i,j} \end{bmatrix} \quad (3 - 15)$$

where $h_{i,j} \in [1, y]$ and $\forall i \leq w, j \leq x$ and the $i - j$ element implies the following rule [43]:

$R_{i,j}$: If E is E_i and D is D_j then U is U_k

In the experiment, E_1=D_1=U_1=NB, E_2=D_2=U_2=NM, E_3=D_3=U_3=NS, E_4=D_4=U_4=ZE, E_5=D_5=U_5=PS, E_6=D_6=U_6=PM, E_7=D_7=U_7=PB, i = 4, j = 2, k = 4, w = x = y = 7.

If M is the output of controller to be fed to the signal conditioner, then it can be formulated as

$$M(k) = \left(T_s \sum_k U(k) \right) + M(0) \quad (3 - 16)$$

Table 3.3 Rules of FLC for speed controller

E	**D**						
	NB (*D1*)	**NM (*D2*)**	**NS (*D3*)**	**ZE (*D4*)**	**PS (*D5*)**	**PM (*D6*)**	**PB (*D7*)**
NB (*D1*)	NB	NM	NM	NM	NM	NM	NM
NM (*D2*)	NM	NM	NM	NM	NM	NM	NM
NS (*D3*)	NS	NS	NS	NS	NS	NS	NS
ZE (*D4*)	Z	Z	Z	Z	PS	PM	PB
PS (*D5*)	Z	PS	PM	PB	PB	PB	PB
PM (*D6*)	Z	PS	PM	PB	PB	PB	PB
PB (*D7*)	Z	PS	PM	PB	PB	PB	PB

The output of controller was provided with anti integral windup using saturation feedback as in Fig. 2.8. Experimentally, the value of *Ka* is $\frac{100}{K_I}$.

3.4.4 Simulation and Experiment Design of Hybrid-Fuzzy Controller

There were three kinds of hybrid controller in the experiment: FLBPI, FLBPID, and FLIC. FLBPID uses the basis of FLBPI with additional of fixed value of K_D. The block diagram of fuzzy-logic-based self tuning PI for speed controller is shown in Fig. 3.22. In this figure, $p_v(k)$ is the process value of speed, r(k) is the reference value or set-point of speed, h is the output of fuzzy controller, and m(k) is the output of controller.

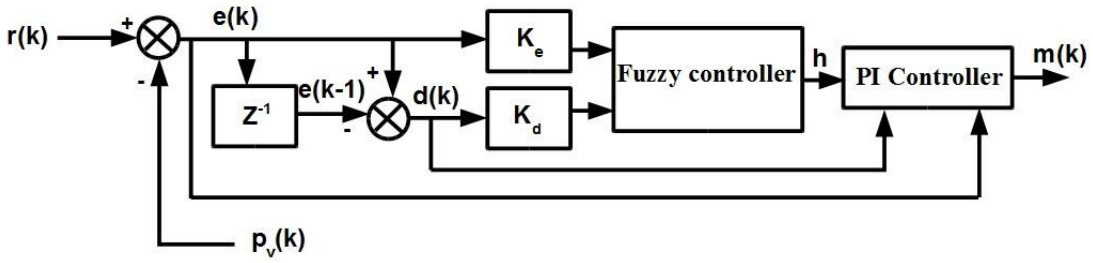


Fig. 3.22 Block diagram of fuzzy-logic-based self-tuning PI for the speed controller [45]

The fuzzy sets and their corresponding membership functions for input error, e(k) and change of error, d(k) and output (h) are shown in Fig. 3.23. The rules for FLBPI and FLBPID (K_P and K_I) are shown in Table 3.4. The value of K_D is constant based on Ziegler-Nichols method in PID tuning.

If K_{Pm} is the maximum value of K_P (experiment), K_{Im} is the maximum value of K_I (experiment), and K_{Dm} is the maximum value of K_D (Ziegler-Nichols), then the values of K_P, K_I, and K_D for hybrid controllers are obtained from Eq. (3 - 17) for FLBPI and Eq. (3 - 18) for FLBPID.

$$K_P = h.K_{Pm};\ K_I = h^2.K_{Im} \qquad (3 - 17)$$

$$K_P = h.K_{Pm};\ K_I = h^2.K_{Im};\ K_D = K_{Dm} \qquad (3 - 18)$$

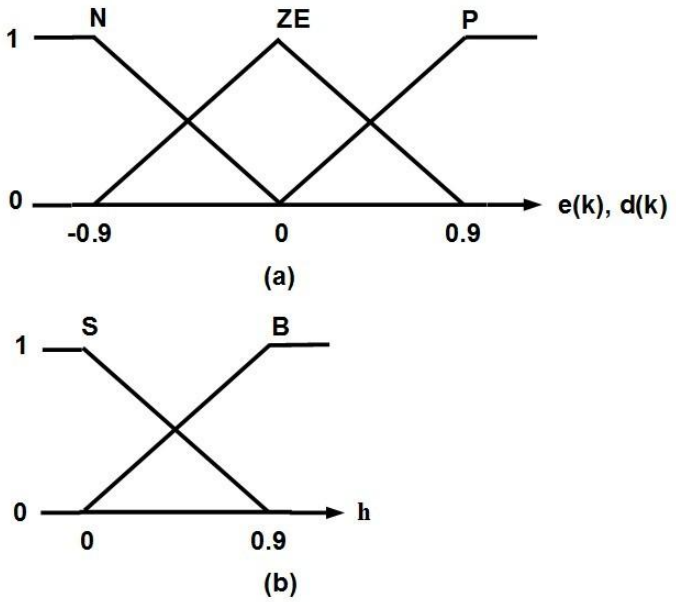


Fig. 3.23 Fuzzy sets and their corresponding membership functions: (a) Input, (b) Output [45]

The FLIC experiment was based on the FLC experiment for speed and the output is paralleled with integral controller which was optimized using experiment method. The structure of FLIC is shown in Fig. 3.24.

Table 3.4 Fuzzy rules base for K_P and K_I in FLBPI [45]

E	**D**		
	N	**ZE**	**P**
N	S	B	S
ZE	S	B	S
P	S	B	S

N: Negative; ZE: Zero; P: Positive; S: Small; B: Big

In the experiment of FLIC, the structure of input and output membership functions are the same as in FLC. The input membership functions is shown in Fig. 3.20 and output membership functions is shown in Fig. 3.25.

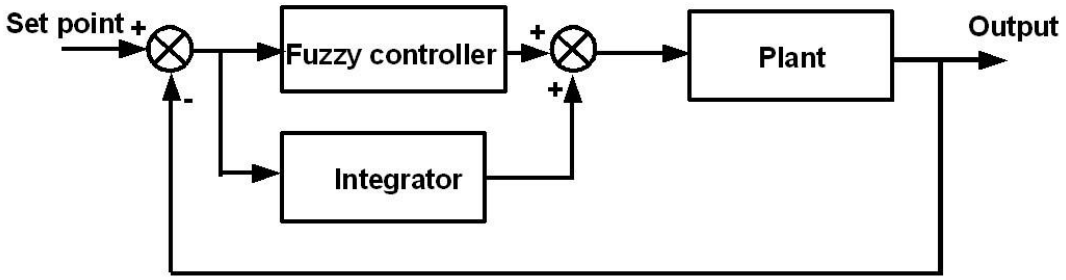
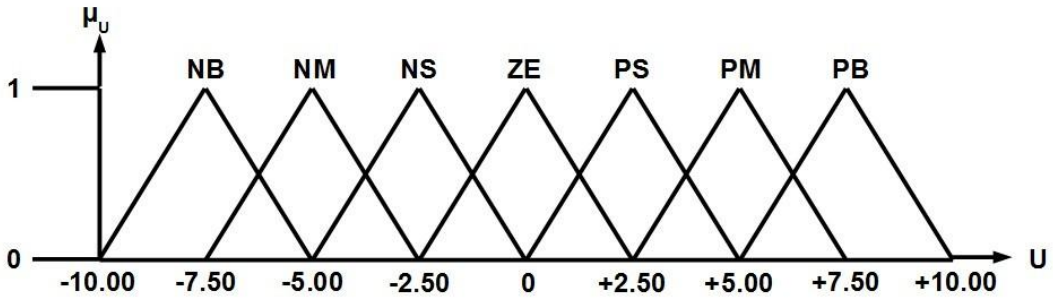


Fig. 3.24 Structure of FLIC



NB : Negative Big; NM : Negative Medium NS : Negative Small; ZE : Zero; PS : Positive Small PM : Positive Medium; PB : Positive Big; U: Output

Fig. 3.25 Fuzzy output membership functions in FLIC

The output of FLC part is clamped to the range of [0.00, 10.00]. If E, D, and U are error input, change of error input, and output of FLC in FLIC, then the rule table in Table 3.3 can be coded into the fuzzy rule string $H_{(w,x,y)}$ which is formulated in the form of integer matrix as shown in Eq. (3 - 15) [43].

If M is the output of controller to be fed to the signal conditioner, e is error signal, K_I is integral constant, T_s is sampling rate, and U is fuzzy output, then the manipulated variable can be formulated as

$$M(k) = \left(K_I T_s \sum_k e(k) \right) + U(k) + M(0) \qquad (3 - 19)$$

The output of integral controller was provided with anti integral windup using saturation feedback as in Fig. 2.8. Experimentally, the value of Ka is $\frac{100}{K_I}$.

3.5 Design of Genetic Algorithm

The process of GA is shown in Fig. 2.12. The first step of GA process is initialization as shown in Fig. 3.26. The maximum population size, i.e. the maximum number of chromosome is $2^{bitlength}$. If the number of chromosome is set to be more than the maximum population size, then it will be clamped to the maximum population size. The random number with the range [0.00, 1.00] is generated for each bit each chromosome. If the number is more than or equal to 0.5 then the bit value is 1. Otherwise, the bit value is 0.

After having initialized the population, the next step is selection process using SUS Roulette wheel as shown in Fig. 3.27. This selection is used for selecting the good fitness to be a new population. In the figure, N_f is the total fitness value. Start from the first position to be occupied, given a random value, z, in the range of [0, N_f] to the first position. Start from the first chromosome, if the fitness value of the first chromosome is larger or the same as z, then the first position will be occupied by the first chromosome. Otherwise, the fitness value of the first chromosome will be combined with the second chromosome and so on until the combined fitness value is larger or equal to z. The first position will then be occupied by the last combination chromosome where the fitness value is larger or equal to z. The procedures are repeated for the second, the third and until the last position of population is attained.

After the selection process, the next step is running the GA operations: crossover and mutation. The flowchart of crossover process is shown in Fig. 3.28. First of all, the two chromosomes are selected and coupled randomly. Then a random value with the range of [0.00, 1.00] is given to the couple of chromosomes. If the random value is smaller or equal to the crossover rate then the crossover operation as in Fig. 2.13 will occur. Otherwise, no crossover operations. The crossover point is selected randomly. This process is repeated until all of the chromosomes in the population are coupled.

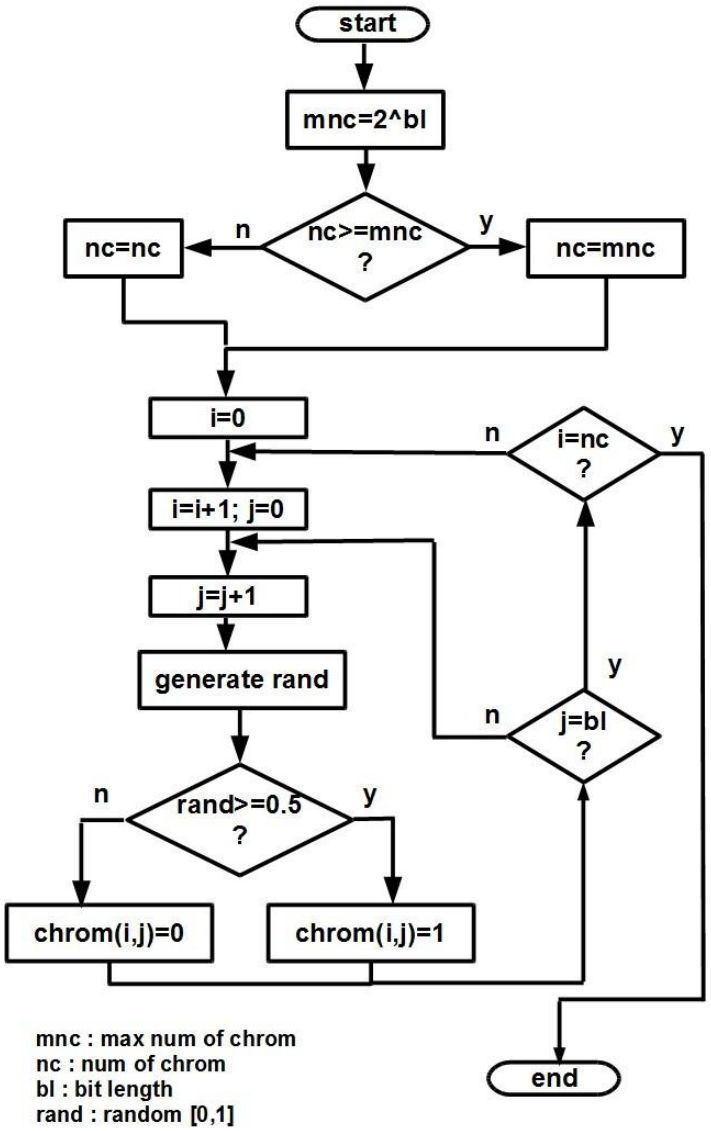


Fig. 3.26 Population initialization using random generation

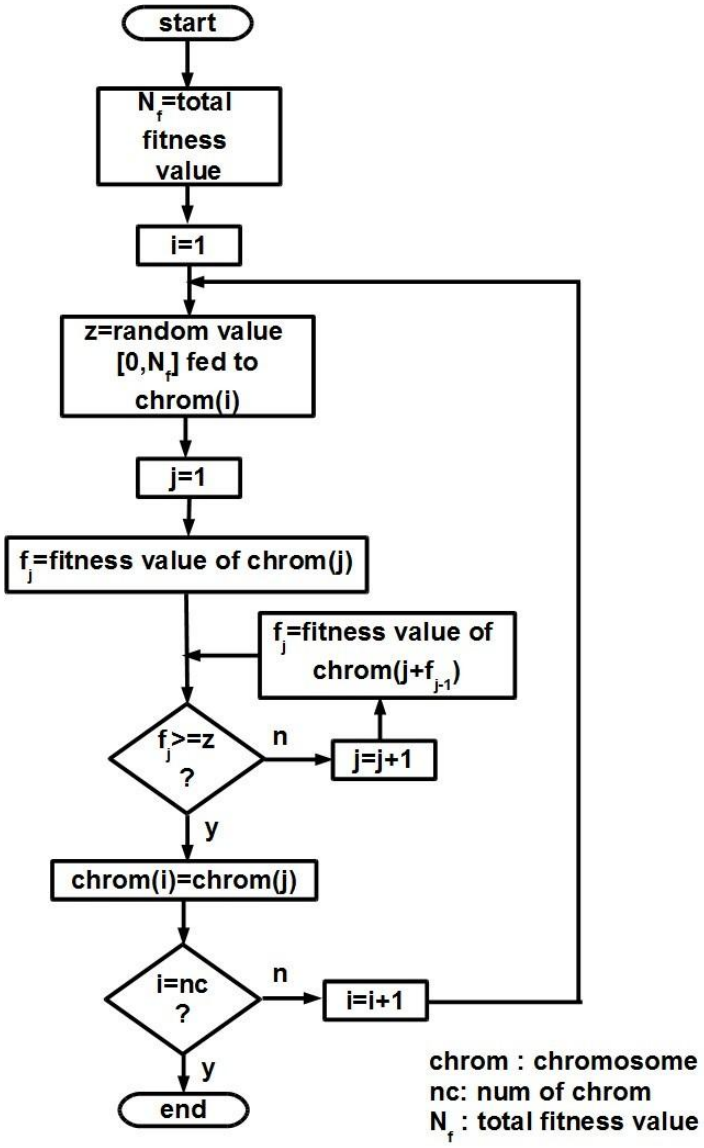


Fig. 3.27 Flowchart of SUS Roulette Wheel selection

The flowchart of mutation process is shown in Fig. 3.29. This process starts with generating a random number in the range of [0.00, 1.00] to each chromosome. If the number is smaller or equal to the mutation rate, then the mutation operation as in Fig. 2.14 will occur. Otherwise, no mutation operation.

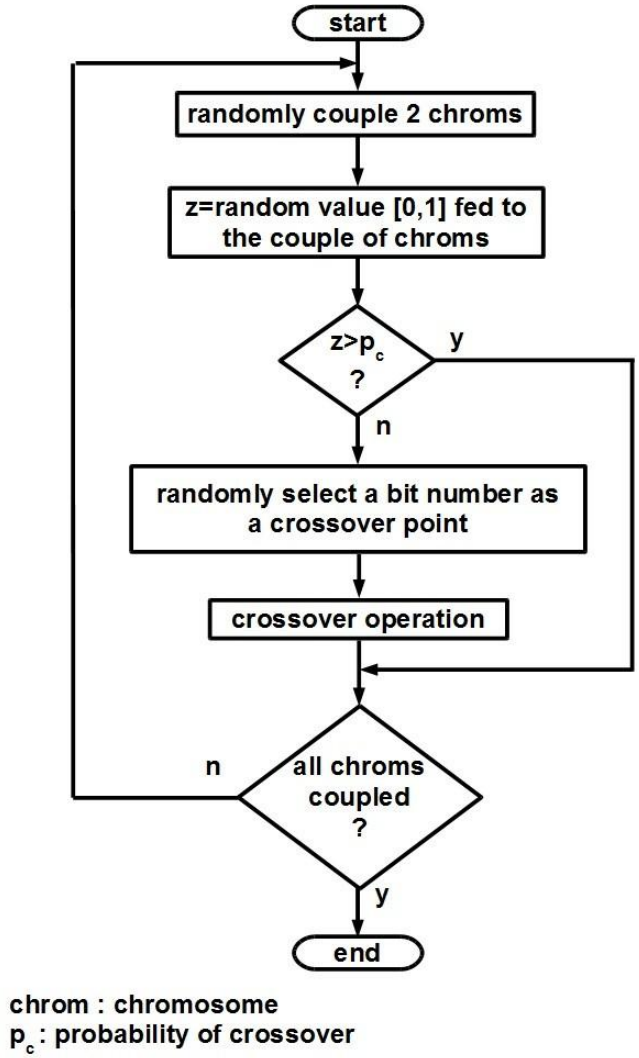


Fig. 3.28 Flowchart of one-point crossover process

The parameters of GA is shown in Table 3.5. The bit length is determined based on the problem to be solved. The crossover rate and mutation rate is determined based on the population size as explained in 2.6.1. Based on literature review, the minimum number of generation is 2 [80] and the maximum is unlimited. In this experiment, the number of generation is set to be 20 [88].

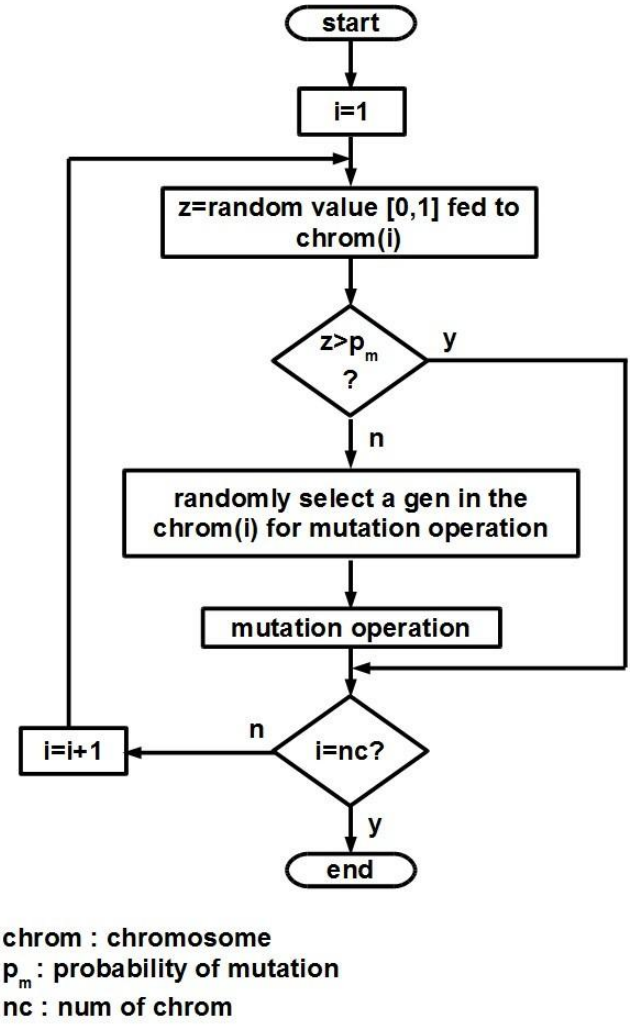


Fig. 3.29 Flowchart of mutation process

Table 3.5 Parameters of GA

NO.	SYMBOL	NAME OF PARAMETER
1	bl	Bit length
2	N_c	Population size
3	N_g	Number of generation
4	p_c	Crossover rate
5	p_m	Mutation rate

The population size is determined based on the bit length and accuracy of the result of the GA process. To understand the population trends, a simulation is needed.

3.5.1 Simulation of GA

Simulation of GA was done to determine the ideal population size according to the bit length with the number of generation is 20.

The simulation is done for conventional GA to get the maximum value of the function as in Eq. (3 - 20) and (3 - 21).

$$f = \max\left(f_1(x) + f_2(x) + f_3(x) + f_4(x)\right) \quad (3 - 20)$$

where

$$f_1(x) = -x^2 + 2x$$

$$f_2(x) = -x^2 + 4x \quad (3 - 21)$$

$$f_3(x) = -x^2 + 6x$$

$$f_4(x) = -x^2 + 8x$$

The process of GA is as in Fig. 2.12 with the max generation of 20. The initialization process is as in Fig. 3.26 with 40 bit length. This value was chosen based on the function in Eq. (3 - 21) that there were four functions where each function had the range of [0.00, 10.00] with the resolution of 0.01. If bl is the bit length, m_d is the maximum value of the decimal number, and r is the resolution, then the bit length for each function is obtained from the formula

$$bl = \log_2\left(\frac{m_d}{r}\right) \quad (3 - 22)$$

From the specifications, using Eq. (3 - 22) gives the following result: 10 bit per function or the total is 40 bit.

The minimum population size is statistically 30, then was increased to 40, 50, and so on until fulfilling a criteria. When the population size is less then 100, the crossover rate is 0.9 and to avoid the premature convergence, the mutation rate is set to 0.01 [80] and 0.1 [88]. When the population size is 100, the crossover rate is 0.6 and to avoid the premature convergence, the mutation rate is set to be 0.01 [80]

The performance index for this experiment is as follows.
The result of GA to solve the Eq. (3 - 20) is compared to the result of manual calculation. In the manual calculation, each function in Eq. (3 - 21) is derived and

then find the *x* such that the value of the derived function is zero. The result value of *x* is {1, 4, 9, 16}. Using these values to solve Eq. (3 - 20) yields 30. The % error is 100 times the deviation between the GA result and the manual result over the manual result.

There are two criteria for this experiment: minimum criteria and good criteria. The minimum criteria is as follows:
If the % error is less than or equal to 5 then the GA result is called *true*. Otherwise, the GA result is called *false*. The experiment was repeated 100 times and then the accuracy was calculated by counting the false result within the 100 experiments. The accuracy is 100 % if there are no false. The ideal population size for this criteria is the minimum population size in the resolution of 10 with 100 % accuracy and less then 2 % average error. The good criteria is as follows:
If the % error is less than or equal to 2 then the GA result is called *true*. Otherwise, the GA result is called *false*. The experiment was repeated 100 times and then the accuracy was calculated by counting the false result within the 100 experiments. The accuracy is 100 % if there are no false. The ideal population size for this criteria is the minimum population size in the resolution of 10 with 100 % accuracy and less than 2 % average error.

3.5.2 The Structure of Semi-Parallel Operation Genetic Algorithm (SPOGA)

Applying one-point crossover operation as in the biological inspiration in the conventional GA for solving the function in Eq. (3 - 20) and Eq. (3 - 21) will raise some problems. Applying parallel genetic algorithms will be more complicated. HGA is appropriate for optimizing membership functions of FLC with the consequence that the crossover point will be more or less than 0.5. It is expected that any new type of GA would solve the problems.

A new GA-based optimization algorithm is proposed in the thesis. The underlying specific mechanism of applying this algorithm in control problem has not been reported. The thrust of the idea for proposing this algorithm in this thesis comes from the conventional GA in particular the intrinsic parallelism architecture of the GA and

the subpartition of chromosomes in HGA, with the chromosome separated into some sub-chromosome according with the problem to be solved as shown in Fig. 3.30.

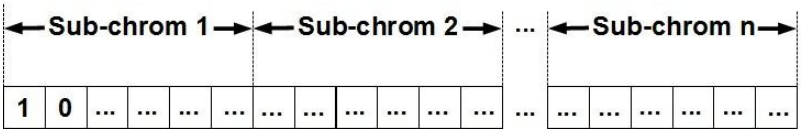


Fig. 3.30 Chromosome structure of SPOGA, typically six bit per sub-chromosome

The selection operation is conventional, but the crossover and mutation operation is parallel process between each sub-chromosome. Noticably, the structure is different from parallel genetic algorithms and therefore it is called *semi-parallel operation genetic algorithm (SPOGA).*

The operation of SPOGA is the same as operation in the HGA. In the HGA, the selection process is done as in the conventional GA, but for crossover and mutation, the control genes and the parameter genes are done separately [43]. In the SPOGA, the crossover and mutation operation is done separately between each sub-chromosome.

It is expected that SPOGA can reduce the population size with still using one-point crossover operation and when it is applied to optimize the membership functions of FLC, the crossover point will still remain 0.5. The process flowchart of SPOGA is shown in Fig. 3.31.

Based on the Fig. 3.31, there are three boxes with dotted line indicating the new proposed methods. Twisted ring counter principle is applied as an initial population instead of using random number principle. SPOGA operator is the specific difference between SPOGA and conventional GA. The solution process is done by searching the chromosome with the best fitness value among the all chromosomes in all generations, this method is based on elitism process as in [19].

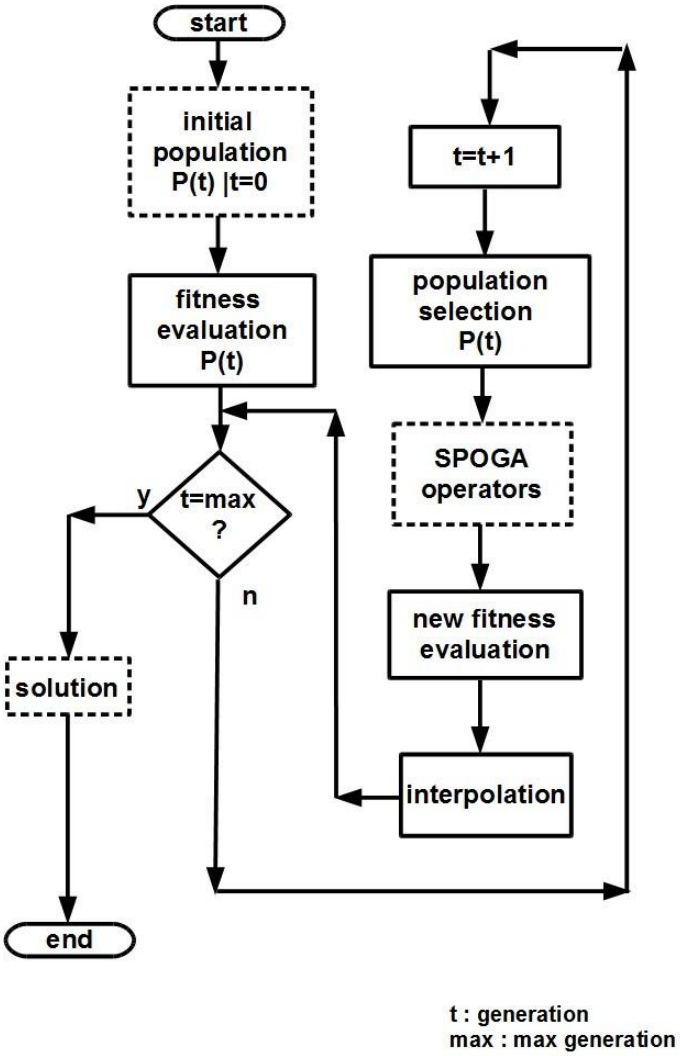


Fig. 3.31 Process flowchart of SPOGA

3.5.3 Simulation of SPOGA

Simulation of SPOGA was done to determine the ideal population size according to the bit length with the number of generation is 20. The simulation was done to get the maximum value of the function as in Eq. (3 - 20) and (3 - 21).

The process of SPOGA is as in Fig. 3.31 with the max generation of 20. The initialization process is shown in Fig. 3.32 with 40 bit length. This value was chosen based on as in Section 3.6.1.

Fig. 3.32 shows the process of population initialization using b_l-bit mod-$2b_l$ N_c-step ring counter. For example, if $b_l=10$ $N_c=30$, then the initial population is shown in Table 3.6.

The minimum population size is statistically 30, then was increased to 40, 50, until fulfilling the criteria. When the population size is less then 100, the crossover rate is 0.9 and to avoid the premature convergence, the mutation rate is set to 0.01 [80] and 0.1 [88]. When the population size is 100, the crossover rate is 0.6 and to avoid the premature convergence, the mutation rate is set to 0.01 [80]. The performance index was done as in Section 3.6.1.

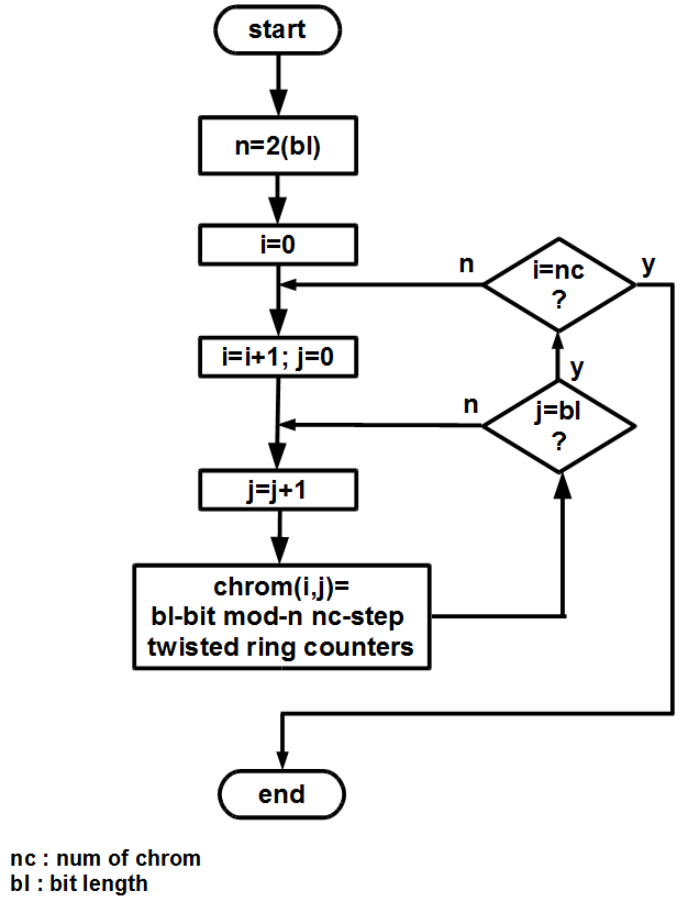


Fig. 3.32 Initial population for SPOGA using twisted ring counters

Table 3.6 Initial population of SPOGA for 10 bit length, 30 population size

No.	Chromosome									
1	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	1
3	0	0	0	0	0	0	0	1	1	1
4	0	0	0	0	0	0	1	1	1	1
5	0	0	0	0	0	1	1	1	1	1
6	0	0	0	0	1	1	1	1	1	1
7	0	0	0	1	1	1	1	1	1	1
8	0	0	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	0
12	1	1	1	1	1	1	1	1	0	0
13	1	1	1	1	1	1	1	0	0	0
14	1	1	1	1	1	1	0	0	0	0
15	1	1	1	1	1	0	0	0	0	0
16	1	1	1	1	0	0	0	0	0	0
17	1	1	1	0	0	0	0	0	0	0
18	1	1	0	0	0	0	0	0	0	0
19	1	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	1
22	0	0	0	0	0	0	0	0	1	1
23	0	0	0	0	0	0	0	1	1	1
24	0	0	0	0	0	0	1	1	1	1
25	0	0	0	0	0	1	1	1	1	1
26	0	0	0	0	1	1	1	1	1	1
27	0	0	0	1	1	1	1	1	1	1
28	0	0	1	1	1	1	1	1	1	1
29	0	1	1	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1	1	1	1

3.6 Design and Application of SPOGA to Optimize Hybrid-Fuzzy Controller

There were three kinds of hybrid-fuzzy controller to be optimized by SPOGA: FLBPI, FLBPID, and FLIC.

3.6.1 Design and Application of SPOGA to Optimize FLBPI

There are two parameters to be optimized in FLBPI: K_{Pm} and K_{Im}. The membership functions and rules of fuzzy logic part is unnecessary to be optimized since the number of membership functions is small and the rules are simple. Experimentally, the input scales of fuzzy logic part have no effect to be optimized and the output scale has to be 1. Therefore, the I/O scales were set to be 1.

Experimentally, the range of K_{Pm} is [0.00, 23.50] and the range of K_{Im} is [0.00, 130.00]. Using Eq. (3 - 22) with resolution of 0.01, the bit length of sub-chromosome for K_{Pm} is 12 and for K_{Im} is 14. The bit length of chromosome is 26. Based on the simulation result in Section 3.6.3 that will be presented in Chapter 4, the population size is about 4/3 times the bit length of chromosome, or 35. The number of generation is 20.

The initial population of each sub-chromosome is shown in Fig. 3.32. In the figure, bl is 12 for K_{Pm} and 14 for K_{Im}. The nc is 35 for both and fulfil the requirement that nc≥n. The result is 12-bit mod-24 35- step ring counter for K_{Pm} and 14-bit mod-28 35- step ring counter for $K_{Im.}$ as shown in Table 3.7.

SUS Roulette wheel was used for the selection operation where the flowchart is shown in Fig. 3.27. The selection work based on the fitness evaluation where the flowchart is shown in Fig. 3.33. Note that the fitness evaluation process used for SPOGA is the same is the GA process. Before selection process, each chromosome in the population is decoded. The result of decoding is then fed to the FLBPI system to get the values of K_{Pm} and $K_{Im.}$ The FLBPI system is then run to control the speed of *s*-modelled DC motor to test the performance of speed controller with regards to the chromosome.

The set-point signal for the speed test run is shown in Fig. 3.34. There are 4 areas in the test. The area 1 is in the set-point of 150 rpm unloaded, the area 2 is in the set-point of 400 rpm unloaded, the area 3 is in the set-point of 250 rpm unloaded, and area 4 is in the set-point of 250 rpm loaded.

As in Fig. 3.33, there are two possibilities of condition when doing the test run: error and success. If the condition is 'error', then the ITAE value is 10^{30}. Else if the condition is 'success', then the ITAE is the sum of ITAE in area 1, ITAE in area 2, ITAE in area 3, and ITAE in area 4.

Table 3.7 Initial population in FLBPI

No.	Sub-Chrom K_{Pm}	Sub-Chrom K_{Im}
1	0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 1
2	0 0 0 0 0 0 0 0 0 0 1 1	0 0 0 0 0 0 0 0 0 0 0 0 1 1
3	0 0 0 0 0 0 0 0 0 1 1 1	0 0 0 0 0 0 0 0 0 0 0 1 1 1
4	0 0 0 0 0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0 0 0 1 1 1 1
5	0 0 0 0 0 0 0 1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 1 1 1 1
6	0 0 0 0 0 0 1 1 1 1 1 1	0 0 0 0 0 0 0 0 1 1 1 1 1 1
7	0 0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 1 1
8	0 0 0 0 1 1 1 1 1 1 1 1	0 0 0 0 0 0 1 1 1 1 1 1 1 1
9	0 0 0 1 1 1 1 1 1 1 1 1	0 0 0 0 0 1 1 1 1 1 1 1 1 1
10	0 0 1 1 1 1 1 1 1 1 1 1	0 0 0 0 1 1 1 1 1 1 1 1 1 1
11	0 1 1 1 1 1 1 1 1 1 1 1	0 0 0 1 1 1 1 1 1 1 1 1 1 1
12	1 1 1 1 1 1 1 1 1 1 1 1	0 0 1 1 1 1 1 1 1 1 1 1 1 1
13	1 1 1 1 1 1 1 1 1 1 1 0	0 1 1 1 1 1 1 1 1 1 1 1 1 1
14	1 1 1 1 1 1 1 1 1 1 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1
15	1 1 1 1 1 1 1 1 1 0 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 0
16	1 1 1 1 1 1 1 1 0 0 0 0	1 1 1 1 1 1 1 1 1 1 1 1 0 0
17	1 1 1 1 1 1 1 0 0 0 0 0	1 1 1 1 1 1 1 1 1 1 1 0 0 0
18	1 1 1 1 1 1 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1 1 0 0 0 0
19	1 1 1 1 1 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1 0 0 0 0 0
20	1 1 1 1 0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 0 0 0 0 0 0
21	1 1 1 0 0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 0 0 0 0 0 0 0
22	1 1 0 0 0 0 0 0 0 0 0 0	1 1 1 1 1 1 0 0 0 0 0 0 0 0
23	1 0 0 0 0 0 0 0 0 0 0 0	1 1 1 1 1 0 0 0 0 0 0 0 0 0
24	0 0 0 0 0 0 0 0 0 0 0 0	1 1 1 1 0 0 0 0 0 0 0 0 0 0
25	0 0 0 0 0 0 0 0 0 0 0 1	1 1 1 0 0 0 0 0 0 0 0 0 0 0
26	0 0 0 0 0 0 0 0 0 0 1 1	1 1 0 0 0 0 0 0 0 0 0 0 0 0
27	0 0 0 0 0 0 0 0 0 1 1 1	1 0 0 0 0 0 0 0 0 0 0 0 0 0
28	0 0 0 0 0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0
29	0 0 0 0 0 0 0 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 1
30	0 0 0 0 0 0 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 1 1
31	0 0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 1 1 1
32	0 0 0 0 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 0 1 1 1 1
33	0 0 0 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 1 1 1 1
34	0 0 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0 1 1 1 1 1 1
35	0 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 1 1

There were three additional criteria in calculating the fitness value: maximum overshoot of all areas, maximum settling time of all areas, and maximum steady state error of all areas. The maximum overshoot criteria is set to be 10 %. The maximum settling time criteria is set to be 2 seconds. The maximum steady state criteria is set to be 2 %. If all of the criteria are fulfilled, then the fitness value is $\frac{10^3}{ITAE}$, otherwise, the fitness value is $\frac{0.8}{ITAE}$.

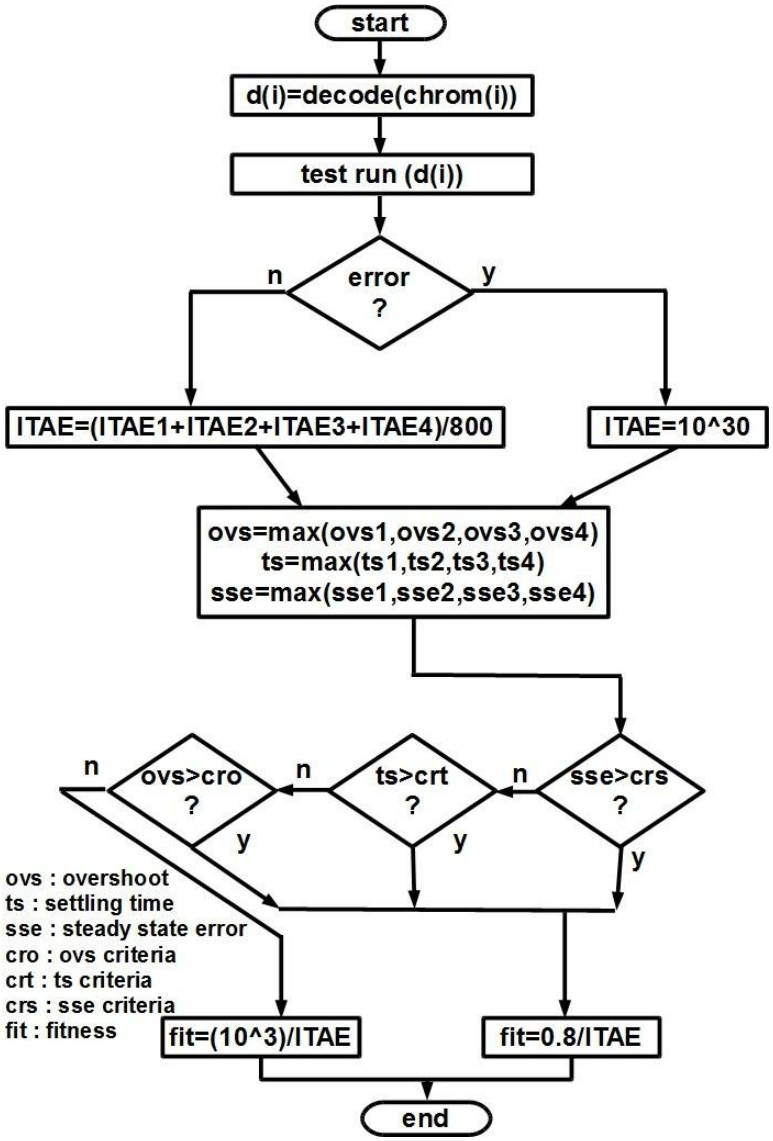


Fig. 3.33 Flowchart of fitness evaluation for SPOGA

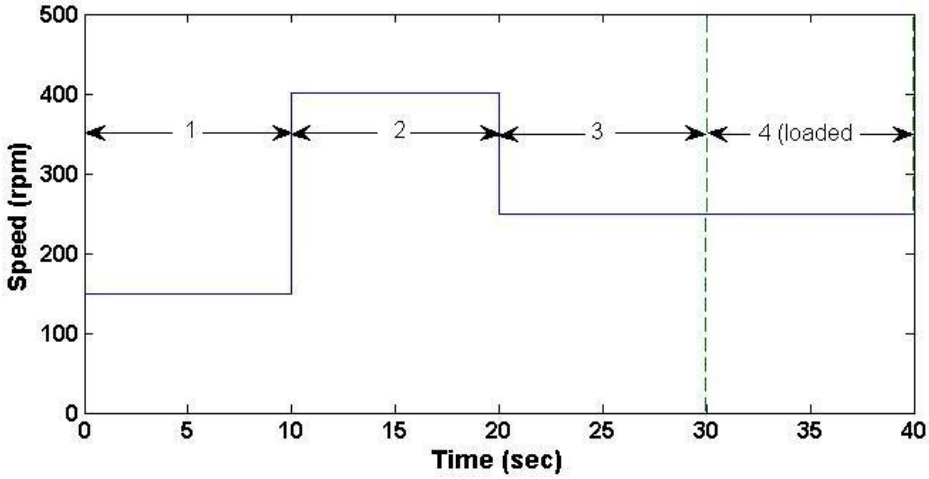


Fig. 3.34 Set-point signal for the speed test run in fitness evaluation

The flowchart of chromosome decoding for FLBPI is shown in Fig. 3.35. The bit length of chromosome is 26. The bit 14 to 25 or the chromosome{1} to chromosome{12} are 12 bit for K_{Pm} , and the bit 0 to 13 or the chromosome{13} to chromosome{26} are 14 bit for K_{Im}. The value of K_{Pm} is the conversion from 12 bit biner number to real number in the range of [0.00, 23.50]. The value of K_{Im} is the conversion from 14 bit biner number to real number in the range of [0.00, 130.00].

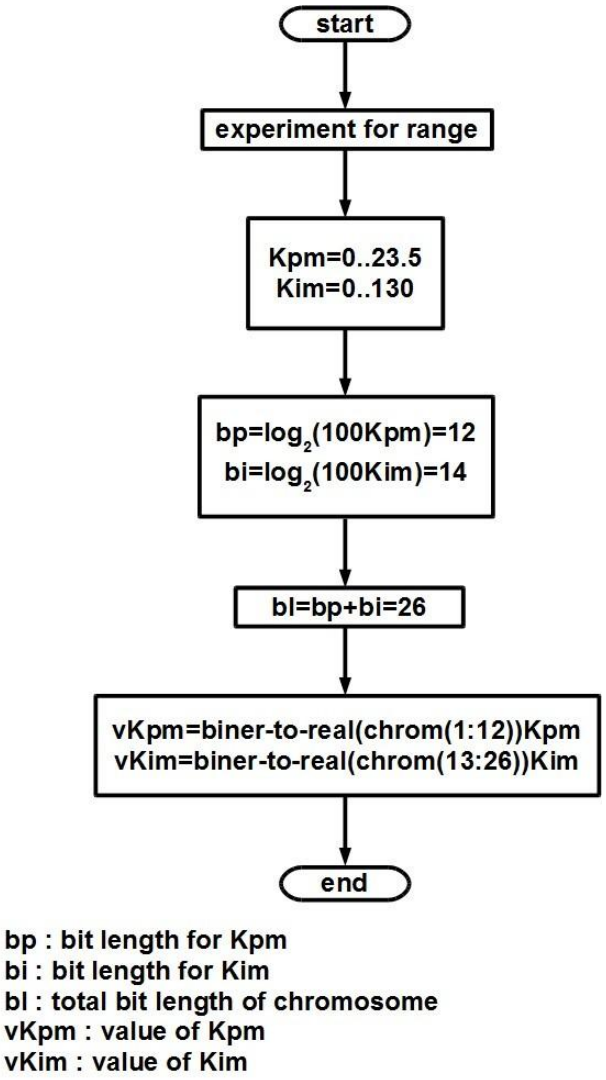


Fig. 3.35 Flowchart of chromosome decoding for FLBPI of SPOGA process

The crossover operation is basically the same as in Fig. 3.28 but the process is done separately for each sub-chromosome as shown in Fig. 3.36. The result is two new sub-chromosomes for K_{Pm} (12 bit) and K_{Im} (14 bit) to the further process, i.e. mutation as shown in Fig. 3.37. As in crossover operation, the mutation operation is basically the same as in Fig. 3.29 but the process is done separately for each sub-chromosome. The result is two new sub-chromosomes for K_{Pm} (12 bit) and K_{Im}

(14 bit) and then combined in one chromosome 26 bit to the further process, i.e. new fitness evaluation as in Fig. 3.33. The operation process is repeated until 20 generations. Finally, the solution process is done by searching the chromosome with the best fitness value among the all chromosomes in all generations as shown in Fig. 3.38.

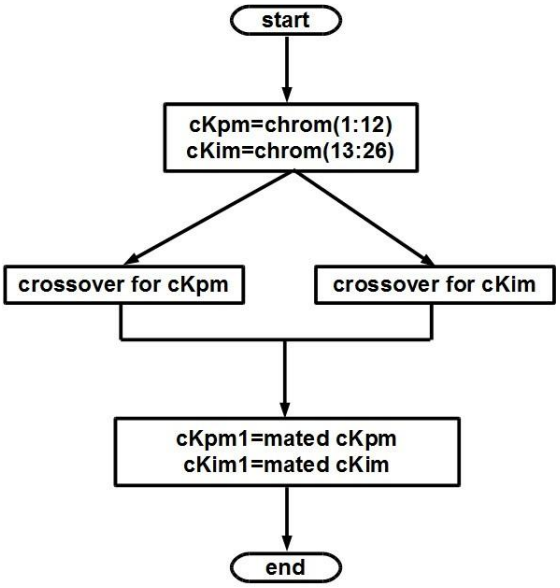


Fig. 3.36 Flowchart of crossover process for FLBPI of SPOGA process

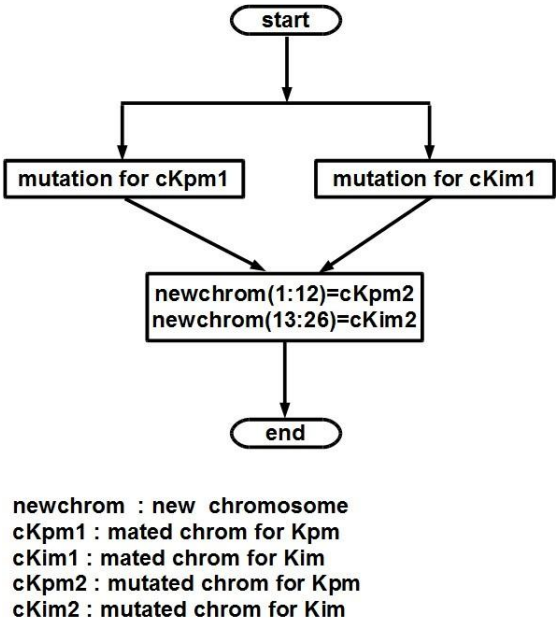


Fig. 3.37 Flowchart of mutation process for FLBPI of SPOGA process

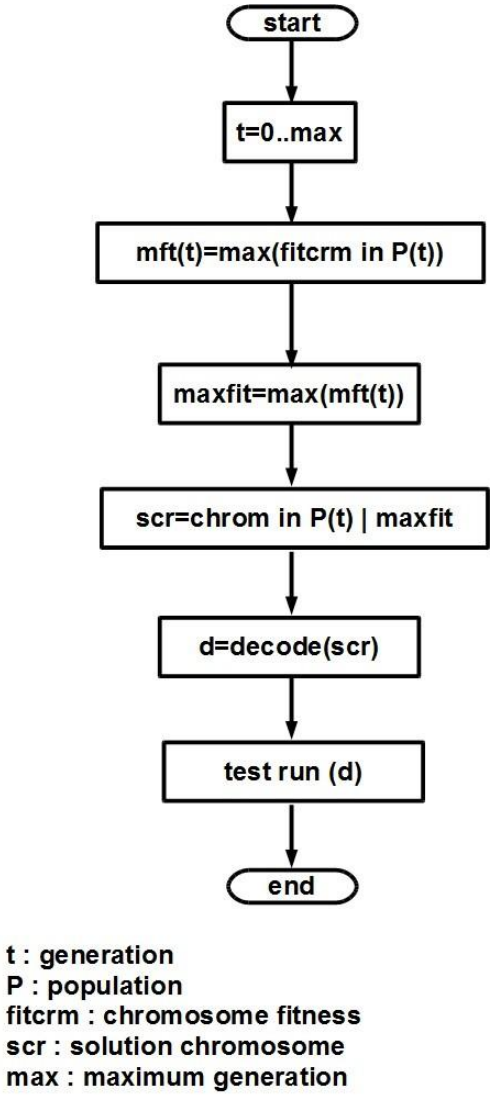


Fig. 3.38 Solution chromosome as a result of SPOGA process

3.6.2 Design and Application of SPOGA to Optimize FLBPID

There are three parameters to be optimized in FLBPID: K_{Pm}, K_{Dm}, and K_{Im}. The membership functions and rules of fuzzy logic part is unnecessary to be optimized for the number of membership functions is small and the rules is simple. Experimentally, the input scales of fuzzy logic part have no effect to be optimized and the output scale has to be 1. Therefore, the I/O scales were set to be 1.

Experimentally, the range of K_{Pm} is [0.00, 23.50] and the range of K_{Im} is [0.00, 110.00]. The range of K_{Dm} is determined based on the result of PID tuning

using Ziegler-Nichols method as in Eq. (2 - 32). This was conducted since the derivative component of PID controller is very sensitive to any disturbances [89]. The range of K_{Dm} was set not too far from the result of Ziegler Nichols, that is [0.00, 0.20]. The result of Ziegler Nichols method in tuning PID would be presented in Chapter 4. Using Eq. (3 - 22) with resolution of 0.01, the bit length of sub-chromosome for K_{Pm} is 12, K_{Dm} is 5, and for K_{Im} is 14. The bit length of chromosome is 31. Based on the simulation result in Section 3.6.3 that will be presented in Chapter 4, the population size is about 4/3 times the bit length of chromosome, or 42. The number of generation is 20.

The initial population of each sub-chromosome is shown in Fig. 3.32. In the figure, the bl is 12 for K_{Pm}, 5 for K_{Dm} , and 14 for K_{Im}. The nc is 42 for both and fulfil the requirement that nc≥n. The result is 12-bit mod-24 42- step ring counter for K_{Pm}, 5-bit mod 10 42- step ring counter for K_{Dm}, and 14-bit mod-28 42- step ring counter for $K_{Im.}$, as shown in Table 3.8.

SUS Roulette wheel was used for selection operation where the flowchart is shown in Fig. 3.27. The selection work based on the fitness evaluation where the flowchart is shown in Fig. 3.33. Before the selection process, each chromosome in the population is decoded. The result of decoding is then fed to the FLBPID system to get the values of K_{Pm}, K_{Dm} , and $K_{Im.}$ The FLBPID system is then run to control the speed of *s*-modelled DC motor to test the performance of speed controller with regards to the chromosome. The set-point signal for the speed test run is shown in Fig. 3.32.

The flowchart of chromosome decoding for FLBPID is shown in Fig. 3.39. The bit length of chromosome is 31. The bit 19 to 30 or the chromosome{1} to chromosome{12} are 12 bit for K_{Pm} , the bit 14 to 18 or the chromosome{13} to chromosome{17} are 5 bit for K_{Dm}, and the bit 0 to 13 or chromosome{18} to chromosome{31} are 5 bit for K_{Im}. The value of K_{Pm} is the conversion from 12 bit biner number to real number in the range [0.00, 23.50]. The value of K_{Im} is the conversion from 14 bit biner number to real number in the range [0.00, 130.00]. The value of K_{Dm} is the conversion from 5 bit biner number to real number in the range [0.00, 0.20].

Table 3.8 Initial population in FLBPID

No.	Sub-Chrom K_{Pm}	Sub-Chrom K_{Dm}	Sub-Chrom K_{Im}
1	0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 1
2	0 0 0 0 0 0 0 0 0 0 0 1 1	0 0 0 1 1	0 0 0 0 0 0 0 0 0 0 0 0 1 1
3	0 0 0 0 0 0 0 0 0 0 1 1 1	0 0 1 1 1	0 0 0 0 0 0 0 0 0 0 0 1 1 1
4	0 0 0 0 0 0 0 0 0 1 1 1 1	0 1 1 1 1	0 0 0 0 0 0 0 0 0 0 1 1 1 1
5	0 0 0 0 0 0 0 0 1 1 1 1 1	1 1 1 1 1	0 0 0 0 0 0 0 0 0 1 1 1 1 1
6	0 0 0 0 0 0 1 1 1 1 1 1 1	1 1 1 1 0	0 0 0 0 0 0 0 0 1 1 1 1 1 1
7	0 0 0 0 0 1 1 1 1 1 1 1 1	1 1 1 0 0	0 0 0 0 0 0 0 1 1 1 1 1 1 1
8	0 0 0 0 1 1 1 1 1 1 1 1 1	1 1 0 0 0	0 0 0 0 0 0 1 1 1 1 1 1 1 1
9	0 0 0 1 1 1 1 1 1 1 1 1 1	1 0 0 0 0	0 0 0 0 0 1 1 1 1 1 1 1 1 1
10	0 0 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 0	0 0 0 0 1 1 1 1 1 1 1 1 1 1
11	0 1 1 1 1 1 1 1 1 1 1 1 1	0 0 0 0 1	0 0 0 1 1 1 1 1 1 1 1 1 1 1
12	1 1 1 1 1 1 1 1 1 1 1 1 1	0 0 0 1 1	0 0 1 1 1 1 1 1 1 1 1 1 1 1
13	1 1 1 1 1 1 1 1 1 1 1 1 0	0 0 1 1 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1
14	1 1 1 1 1 1 1 1 1 1 1 0 0	0 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1
15	1 1 1 1 1 1 1 1 1 0 0 0	1 1 1 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 0
16	1 1 1 1 1 1 1 1 0 0 0 0	1 1 1 1 0	1 1 1 1 1 1 1 1 1 1 1 1 0 0
17	1 1 1 1 1 1 1 0 0 0 0 0	1 1 1 0 0	1 1 1 1 1 1 1 1 1 1 1 0 0 0
18	1 1 1 1 1 1 0 0 0 0 0 0	1 1 0 0 0	1 1 1 1 1 1 1 1 1 1 0 0 0 0
19	1 1 1 1 1 0 0 0 0 0 0 0	1 0 0 0 0	1 1 1 1 1 1 1 1 1 0 0 0 0 0
20	1 1 1 1 0 0 0 0 0 0 0 0	0 0 0 0 0	1 1 1 1 1 1 1 1 0 0 0 0 0 0
21	1 1 1 0 0 0 0 0 0 0 0 0	0 0 0 0 1	1 1 1 1 1 1 1 0 0 0 0 0 0 0
22	1 1 0 0 0 0 0 0 0 0 0 0	0 0 0 1 1	1 1 1 1 1 1 0 0 0 0 0 0 0 0
23	1 0 0 0 0 0 0 0 0 0 0 0	0 0 1 1 1	1 1 1 1 1 0 0 0 0 0 0 0 0 0
24	0 0 0 0 0 0 0 0 0 0 0 0	0 1 1 1 1	1 1 1 1 0 0 0 0 0 0 0 0 0 0
25	0 0 0 0 0 0 0 0 0 0 0 0 1	1 1 1 1 1	1 1 1 0 0 0 0 0 0 0 0 0 0 0
26	0 0 0 0 0 0 0 0 0 0 0 1 1	1 1 1 1 0	1 1 0 0 0 0 0 0 0 0 0 0 0 0
27	0 0 0 0 0 0 0 0 0 1 1 1 1	1 1 1 0 0	1 0 0 0 0 0 0 0 0 0 0 0 0 0
28	0 0 0 0 0 0 0 0 1 1 1 1 1	1 1 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0
29	0 0 0 0 0 0 0 1 1 1 1 1 1	1 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 1
30	0 0 0 0 0 0 1 1 1 1 1 1 1	0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 1 1
31	0 0 0 0 0 1 1 1 1 1 1 1 1	0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 1 1 1
32	0 0 0 0 1 1 1 1 1 1 1 1 1	0 0 0 1 1	0 0 0 0 0 0 0 0 0 0 1 1 1 1
33	0 0 0 1 1 1 1 1 1 1 1 1 1	0 0 1 1 1	0 0 0 0 0 0 0 0 0 1 1 1 1 1
34	0 0 1 1 1 1 1 1 1 1 1 1 1	0 1 1 1 1	0 0 0 0 0 0 0 0 1 1 1 1 1 1
35	0 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 1	0 0 0 0 0 0 0 1 1 1 1 1 1 1
36	1 1 1 1 1 1 1 1 1 1 1 1 1	1 1 1 1 0	0 0 0 0 0 0 1 1 1 1 1 1 1 1
37	1 1 1 1 1 1 1 1 1 1 1 1 0	1 1 1 0 0	0 0 0 0 0 1 1 1 1 1 1 1 1 1
38	1 1 1 1 1 1 1 1 1 1 1 0 0	1 1 0 0 0	0 0 0 0 1 1 1 1 1 1 1 1 1 1
39	1 1 1 1 1 1 1 1 1 1 0 0 0	1 0 0 0 0	0 0 0 1 1 1 1 1 1 1 1 1 1 1
40	1 1 1 1 1 1 1 1 1 0 0 0 0	0 0 0 0 0	0 0 1 1 1 1 1 1 1 1 1 1 1 1
41	1 1 1 1 1 1 1 0 0 0 0 0 0	0 0 0 0 1	0 1 1 1 1 1 1 1 1 1 1 1 1 1
42	1 1 1 1 1 1 0 0 0 0 0 0 0	0 0 0 1 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1

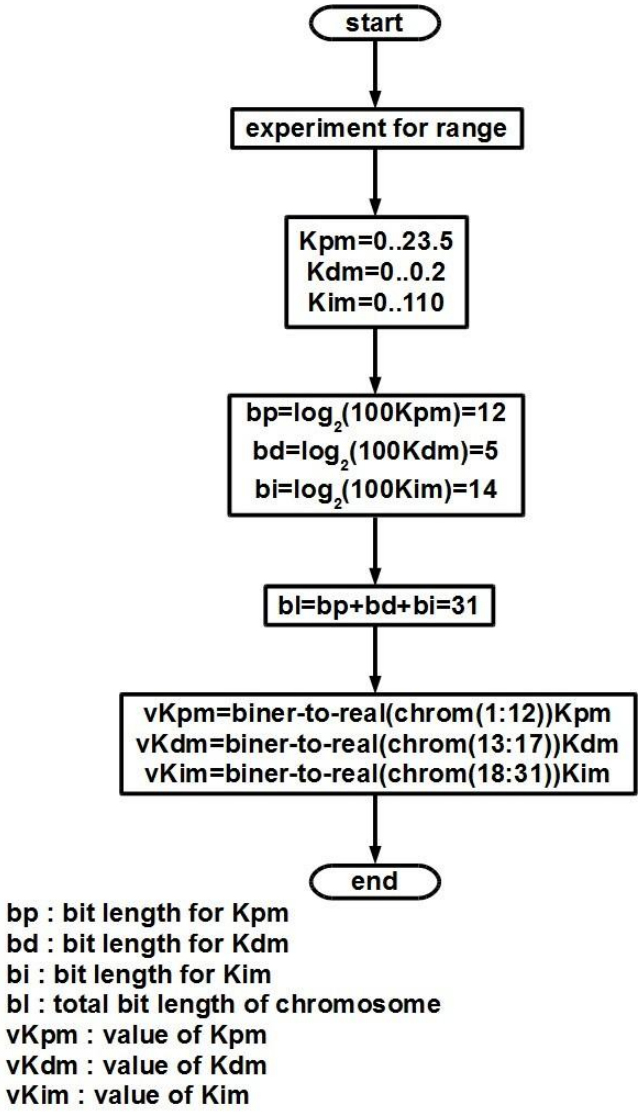


Fig. 3.39 Flowchart of chromosome decoding for FLBPID of SPOGA process

The crossover operation is basically the same as in Fig. 3.28 but the process is done separately for each sub-chromosome as shown in Fig. 3.40. The result is three new sub-chromosomes for K_{Pm} (12 bit), K_{Dm} (5 bit), and K_{Im} (14 bit) to the further process, i.e. mutation as shown in Fig. 3.41. As in crossover operation, the mutation operation is basically the same as in Fig. 3.29 but the process is done separately for each sub-chromosome. The result is three new sub-chromosomes for K_{Pm} (12 bit), K_{Dm} (5 bit), and K_{Im} (14 bit) and then combined in one chromosome 31 bit to the further process, i.e. new fitness evaluation as in Fig. 3.33. The operation process is repeated until 20 generations. Finally, the solution process is done by searching the chromosome with the best fitness value among the all chromosomes in all generations as shown in Fig. 3.38

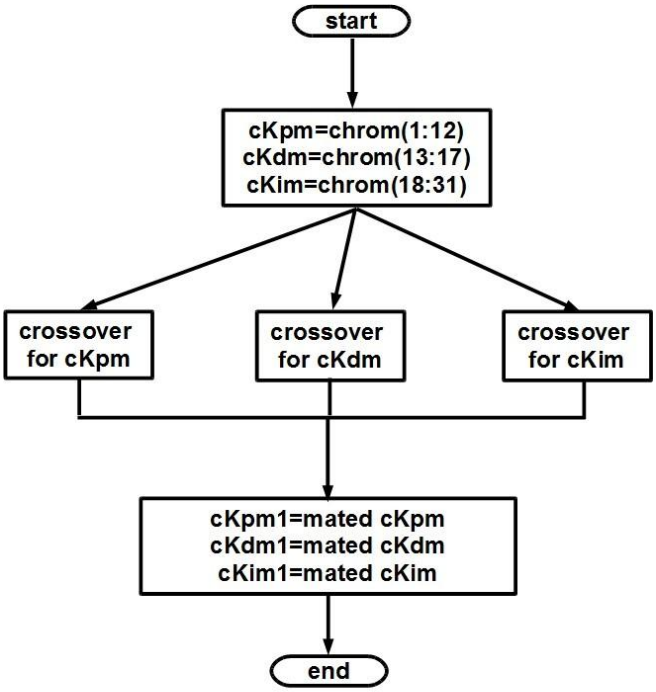


Fig. 3.40 Flowchart of crossover process for FLBPID of SPOGA process

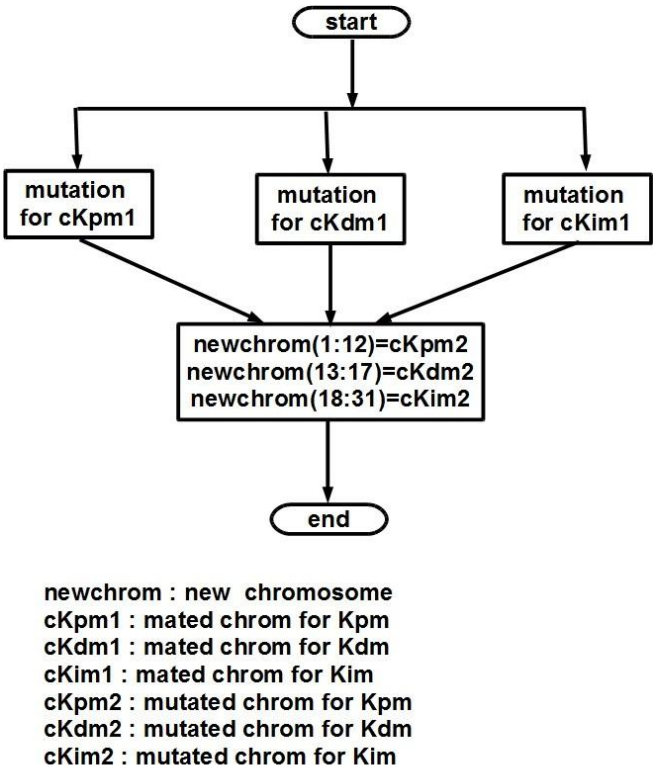


Fig. 3.41 Flowchart of mutation process for FLBPID of SPOGA process

3.6.3 Design and Application of SPOGA to Optimize FLIC

Regarding to the structure of FLIC as shown in Fig. 3.24, there are two steps in optimizing FLIC using SPOGA: optimizing the membership functions and rules of FLC part, and optimizing I/O scales of FLC part and integral constant of integral controller part.

3.6.3.1 Optimizing Membership Function and Rules

There are three parameters to be optimized in membership functions: the error membership functions, E, the change of error membership functions, D, and the output membership functions, U. In this process, the I/O constant of FLC part is set to be 1 and the integral constant of integral controller part is set to be 0.

Initially, there are seven triangular membership functions for input and output of FLC as shown in Fig. 3.20 and Fig. 3.21. Consequently, there are three sub-chromosomes with 7 bit each, for E, D, and U. The bit length of chromosome is 21. Based on the simulation result in Section 3.6.3 that will be presented in Chapter 4, the population size is about 4/3 times the bit length of chromosome, or 30, since statistically, the minimum population size is 30. The number of generation is 20. The initial population of each sub-chromosome is shown in Fig. 3.32. The result is 7-bit mod-14 30- step ring counter for E, D, and U, as shown in Table 3.9.

For consistency, SUS Roulette wheel was used for the selection operation where the flowchart is shown in Fig. 3.27. The selection work based on the fitness evaluation where the flowchart is shown in Fig. 3.33. The maximum overshoot criteria was set to 100 %, while the maximum settling time criteria is set to be 10 seconds, and the maximum steady state error criteria is set to be 100 %. Before the selection process, each chromosome in the population is decoded. The result of the decoding is then fed to the FLIC system to get the E, D, and U. The FLIC system is then run to control the speed of *s*-modelled DC motor to test the performance of speed controller with regards to the chromosome. The set-point signal for the speed test run is shown in Fig. 3.34.

Table 3.9 Initial population for membership functions of FLIC

No.	Sub-Chrom E	Sub-Chrom D	Sub-Chrom U
1	0 0 0 0 0 0 1	0 0 0 0 0 0 1	0 0 0 0 0 0 1
2	0 0 0 0 0 1 1	0 0 0 0 0 1 1	0 0 0 0 0 1 1
3	0 0 0 0 1 1 1	0 0 0 0 1 1 1	0 0 0 0 1 1 1
4	0 0 0 1 1 1 1	0 0 0 1 1 1 1	0 0 0 1 1 1 1
5	0 0 1 1 1 1 1	0 0 1 1 1 1 1	0 0 1 1 1 1 1
6	0 1 1 1 1 1 1	0 1 1 1 1 1 1	0 1 1 1 1 1 1
7	1 1 1 1 1 1 1	1 1 1 1 1 1 1	1 1 1 1 1 1 1
8	1 1 1 1 1 1 0	1 1 1 1 1 1 0	1 1 1 1 1 1 0
9	1 1 1 1 1 0 0	1 1 1 1 1 0 0	1 1 1 1 1 0 0
10	1 1 1 1 0 0 0	1 1 1 1 0 0 0	1 1 1 1 0 0 0
11	1 1 1 0 0 0 0	1 1 1 0 0 0 0	1 1 1 0 0 0 0
12	1 1 0 0 0 0 0	1 1 0 0 0 0 0	1 1 0 0 0 0 0
13	1 0 0 0 0 0 0	1 0 0 0 0 0 0	1 0 0 0 0 0 0
14	0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0
15	0 0 0 0 0 0 1	0 0 0 0 0 0 1	0 0 0 0 0 0 1
16	0 0 0 0 0 1 1	0 0 0 0 0 1 1	0 0 0 0 0 1 1
17	0 0 0 0 1 1 1	0 0 0 0 1 1 1	0 0 0 0 1 1 1
18	0 0 0 1 1 1 1	0 0 0 1 1 1 1	0 0 0 1 1 1 1
19	0 0 1 1 1 1 1	0 0 1 1 1 1 1	0 0 1 1 1 1 1
20	0 1 1 1 1 1 1	0 1 1 1 1 1 1	0 1 1 1 1 1 1
21	1 1 1 1 1 1 1	1 1 1 1 1 1 1	1 1 1 1 1 1 1
22	1 1 1 1 1 1 0	1 1 1 1 1 1 0	1 1 1 1 1 1 0
23	1 1 1 1 1 0 0	1 1 1 1 1 0 0	1 1 1 1 1 0 0
24	1 1 1 1 0 0 0	1 1 1 1 0 0 0	1 1 1 1 0 0 0
25	1 1 1 0 0 0 0	1 1 1 0 0 0 0	1 1 1 0 0 0 0
26	1 1 0 0 0 0 0	1 1 0 0 0 0 0	1 1 0 0 0 0 0
27	1 0 0 0 0 0 0	1 0 0 0 0 0 0	1 0 0 0 0 0 0
28	0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0
29	0 0 0 0 0 0 1	0 0 0 0 0 0 1	0 0 0 0 0 0 1
30	0 0 0 0 0 1 1	0 0 0 0 0 1 1	0 0 0 0 0 1 1

The flowchart of chromosome decoding for FLIC is as shown in Fig. 3.42. The bit length of chromosome is 21. The bit 14 to 20 or the chromosome{1} to chromosome{7} are 7 bit for E, the bit 7 to 13 or the chromosome{8} to chromosome{14} are 7 bit for D, and the bit 0 to 6 or chromosome{15} to chromosome{21} are 7 bit for U. The related membership functions are shown in Fig. 3.43 where μ_i is i^{th}-fuzzy membership function, α_{ia}, α_{ib}, and α_{ic} are i^{th}-fuzzy boundaries.

Note that when the membership function is deleted, then the existing neighbour membership functions will change their boundary of fuzzy membership functions such that the crossover point is still 0.5.

The crossover operation is basically the same as in Fig. 3.28 but the process is done separately for each sub-chromosomes as shown in Fig. 3.44. The result is three new sub-chromosomes for E (7 bit), D (7 bit), and U (7 bit) to the further process, i.e.

mutation as shown in Fig. 3.45. As in crossover operation, the mutation operation is basically the same as in Fig. 3.29 but the process is done separately for each sub-chromosome. The result is three new sub-chromosomes for E (7 bit), D (7 bit), and U (7 bit) and then combined in one 21-bit size chromosome to the further process, i.e. new fitness evaluation as in Fig. 3.33. The operation process is repeated until 20 generations. Finally, the solution process is done by searching the chromosome with the best fitness value among the all chromosomes in all generations as shown in Fig. 3.38.

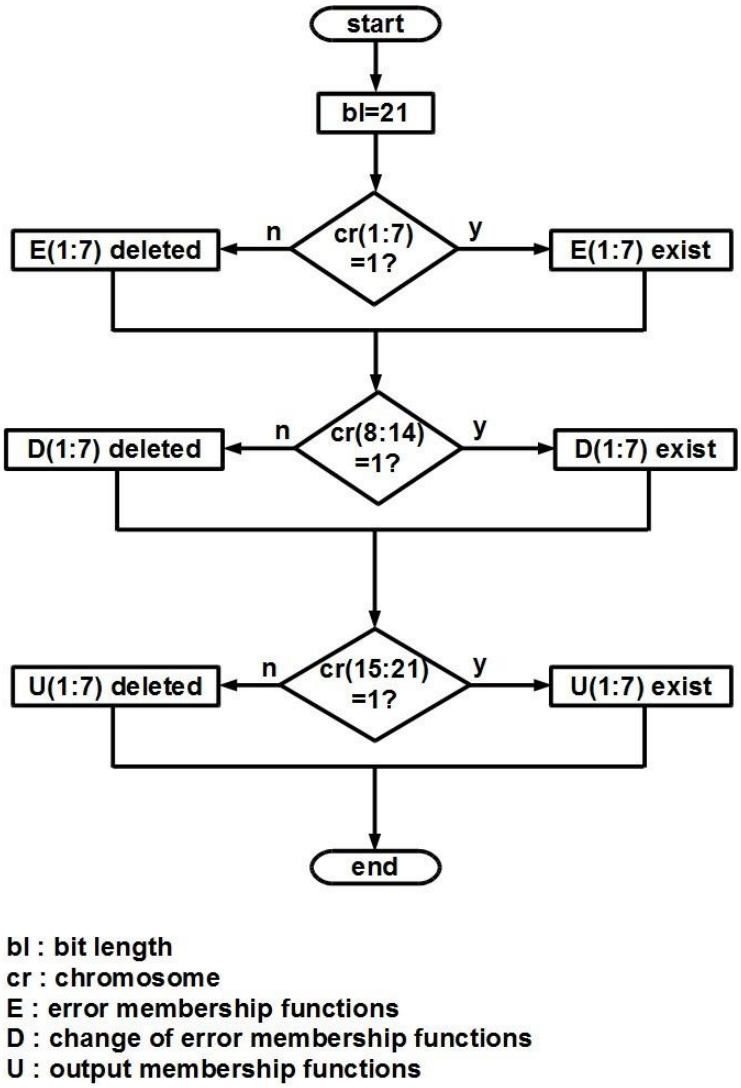


Fig. 3.42 Flowchart of chromosome decoding of SPOGA process for membership functions in FLIC

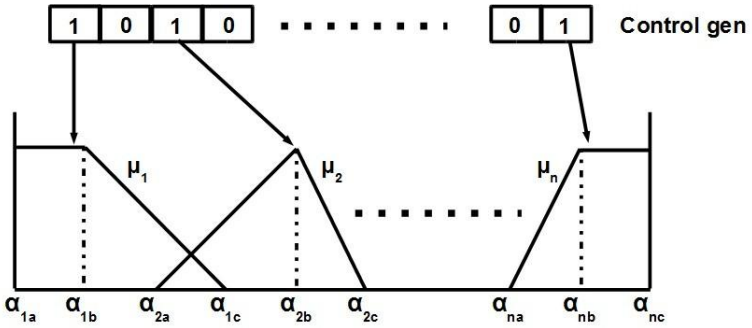


Fig. 3.43 Fuzzy membership functions related to chromosome

There are only one operation in the rules optimization, i.e. mutation operation with the mutation rate of 0.01. The rules to be optimized is as in Eq. (3 - 15) using delta shift operation which alters each element in the fuzzy rule chromosome as follows [80]:

$$h_{i,j} = h_{i+\Delta i, j+\Delta j} \qquad (3 - 23)$$

where Δi, Δj have equal chance to be +1 or -1 with a probability of 0.01 [80].

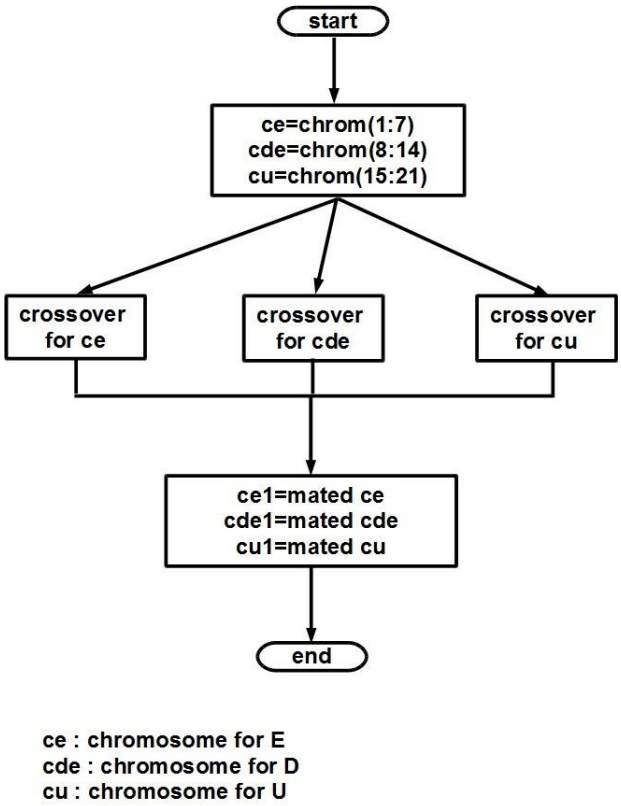


Fig. 3.44 Flowchart of crossover process in SPOGA process for membership functions in FLIC

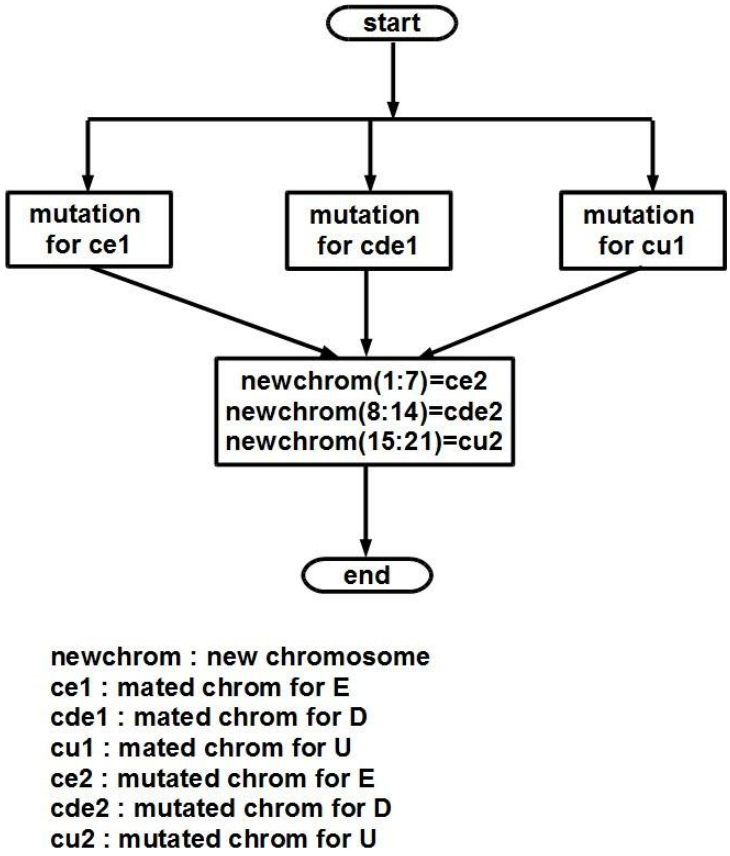


Fig. 3.45 Flowchart of mutation process in SPOGA process for membership functions in FLIC

3.6.3.2 Optimizing I/O Scales and Integral Constant

There are three parameters to be optimized: the input scale for fuzzy error input, K_e, the output scale for fuzzy output, K_u, and the integral constant, K_I. This is done with the optimized membership functions and rules. Experimentally, the input scale for fuzzy change of error input, K_{ce}, have no effect to be optimized, thus, it is set to be 1.

Experimentally, the range of K_e is [0.00, 1.50], the range of K_u is [0.00, 0.50]. The range of K_i is [0.00, 1.00]. Using Eq. (3 - 22) with resolution of 0.01, the bit length of sub-chromosome for K_e is 8, K_u is 6, and for K_I is 7. The bit length of chromosome is 21. Based on the simulation result in Section 3.6.3 that will be presented in Chapter 4, the population size is about 4/3 times the bit length of chromosome, but statistically, the minimum population size is 30. The number of generation is 20.

The initial population of each sub-chromosome is shown in Fig. 3.32. In the figure, the bl is 8 for K_e, 6 for K_u , and 7 for K_I. The nc is 30 for all and fulfil the requirement that nc≥n. The result is 8-bit mod-16 30- step ring counter for K_e, 6-bit mod 12 30- step ring counter for K_u, and 7-bit mod-14 30- step ring counter for K_I, as shown in Table 3.10.

Table 3.10 Initial population for I/O scales and integral constant

No.	Sub-Chrom K_e	Sub-Chrom K_u	Sub-Chrom K_I
1	0 0 0 0 0 0 0 1	0 0 0 0 0 1	0 0 0 0 0 0 1
2	0 0 0 0 0 0 1 1	0 0 0 0 1 1	0 0 0 0 0 1 1
3	0 0 0 0 0 1 1 1	0 0 0 1 1 1	0 0 0 0 1 1 1
4	0 0 0 0 1 1 1 1	0 0 1 1 1 1	0 0 0 1 1 1 1
5	0 0 0 1 1 1 1 1	0 1 1 1 1 1	0 0 1 1 1 1 1
6	0 0 1 1 1 1 1 1	1 1 1 1 1 1	0 1 1 1 1 1 1
7	0 1 1 1 1 1 1 1	1 1 1 1 1 0	1 1 1 1 1 1 1
8	1 1 1 1 1 1 1 1	1 1 1 1 0 0	1 1 1 1 1 1 0
9	1 1 1 1 1 1 1 0	1 1 1 0 0 0	1 1 1 1 1 0 0
10	1 1 1 1 1 1 0 0	1 1 0 0 0 0	1 1 1 1 0 0 0
11	1 1 1 1 1 0 0 0	1 0 0 0 0 0	1 1 1 0 0 0 0
12	1 1 1 1 0 0 0 0	0 0 0 0 0 0	1 1 0 0 0 0 0
13	1 1 1 0 0 0 0 0	0 0 0 0 0 1	1 0 0 0 0 0 0
14	1 1 0 0 0 0 0 0	0 0 0 0 1 1	0 0 0 0 0 0 0
15	1 0 0 0 0 0 0 0	0 0 0 1 1 1	0 0 0 0 0 0 1
16	0 0 0 0 0 0 0 0	0 0 1 1 1 1	0 0 0 0 0 1 1
17	0 0 0 0 0 0 0 1	0 1 1 1 1 1	0 0 0 0 1 1 1
18	0 0 0 0 0 0 1 1	1 1 1 1 1 1	0 0 0 1 1 1 1
19	0 0 0 0 0 1 1 1	1 1 1 1 1 0	0 0 1 1 1 1 1
20	0 0 0 0 1 1 1 1	1 1 1 1 0 0	0 1 1 1 1 1 1
21	0 0 0 1 1 1 1 1	1 1 1 0 0 0	1 1 1 1 1 1 1
22	0 0 1 1 1 1 1 1	1 1 0 0 0 0	1 1 1 1 1 1 0
23	0 1 1 1 1 1 1 1	1 0 0 0 0 0	1 1 1 1 1 0 0
24	1 1 1 1 1 1 1 1	0 0 0 0 0 0	1 1 1 1 0 0 0
25	1 1 1 1 1 1 1 0	0 0 0 0 0 1	1 1 1 0 0 0 0
26	1 1 1 1 1 1 0 0	0 0 0 0 1 1	1 1 0 0 0 0 0
27	1 1 1 1 1 0 0 0	0 0 0 1 1 1	1 0 0 0 0 0 0
28	1 1 1 1 0 0 0 0	0 0 1 1 1 1	0 0 0 0 0 0 0
29	1 1 1 0 0 0 0 0	0 1 1 1 1 1	0 0 0 0 0 0 1
30	1 1 0 0 0 0 0 0	1 1 1 1 1 1	0 0 0 0 0 1 1

As with the other cases, SUS Roulette wheel was used for selection operation where the flowchart is shown in Fig. 3.27. The selection work based on the fitness evaluation where the flowchart is shown in Fig. 3.33. The maximum overshoot criteria is set to be 25 %. The maximum settling time criteria is set to be 4 seconds. The maximum steady state error criteria is set to be 2 %. Before selection process, each chromosome in the population is decoded. The result of decoding is then fed to the FLIC system to get the values of K_e, K_u , and K_I. The FLIC system is then run to control the speed of *s*-modelled DC motor to test the performance of speed controller

with regards to to the chromosome. The set-point signal for the speed test run is shown in Fig. 3.34.

The flowchart of chromosome decoding for FLIC is shown in Fig. 3.46. The bit length of chromosome is 21. The bit 13 to 20 or the chromosome{1} to chromosome{8} are 8 bit for K_e , the bit 7 to 12 or the chromosome{9} to chromosome{14} are 6 bit for K_u, and the bit 0 to 6 or chromosome{15} to chromosome{21} are 7 bit for K_I. The value of K_e is the conversion from 8 bit biner number to real number in the range [0.00, 1.50]. The value of K_u is the conversion from 6 bit biner number to real number in the range [0.00, 0.50]. The value of K_I is the conversion from 7 bit binary number to real number in the range [0.00, 1.00].

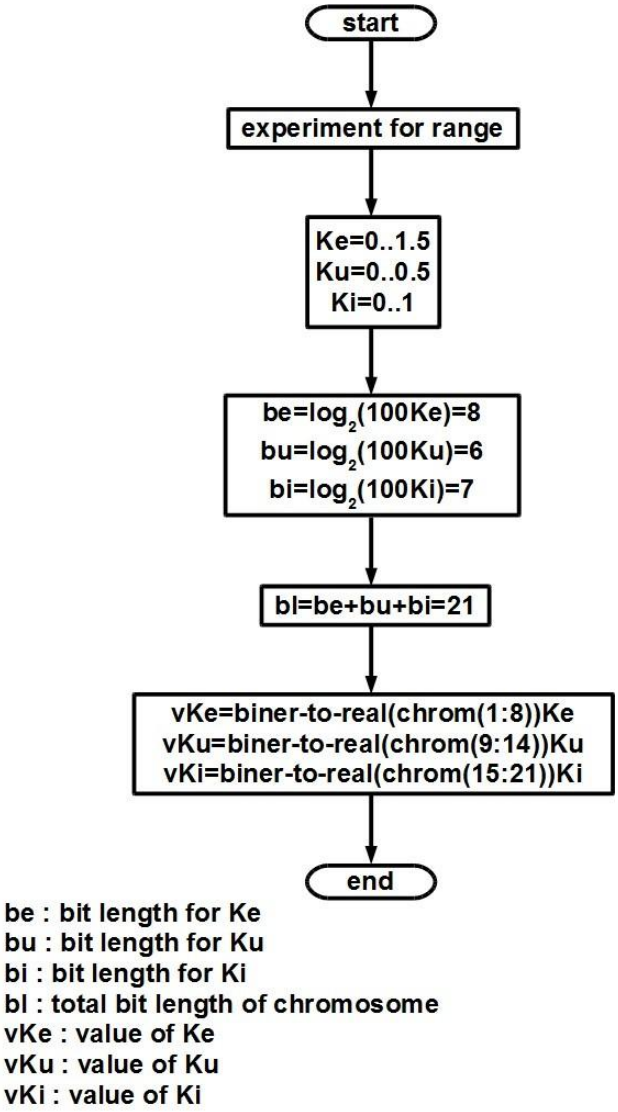


Fig. 3.46 Flowchart of chromosome decoding in SPOGA process for I/O/ scales and integral constant

The crossover operation is basically the same as in Fig. 3.28 but the process is done separately for each sub-chromosome as shown in Fig. 3.47. The result is three new sub-chromosomes for K_e (8 bit), K_u (6 bit), and K_I (7 bit) to the further process, i.e. mutation as shown in Fig. 3.28. As in crossover operation, the mutation operation is basically the same as in Fig. 3.29 but the process is done separately for each sub-chromosome. The result is three new sub-chromosomes for K_e (8 bit), K_u (6 bit), and K_I (7 bit) and then combined in one chromosome 21 bit for the further process, i.e. new fitness evaluation as in Fig. 3.33. The operation process is repeated until 20 generations. Finally, the solution process is done by searching the chromosome with the best fitness value among the all chromosomes in all generations as shown in Fig. 3.38.

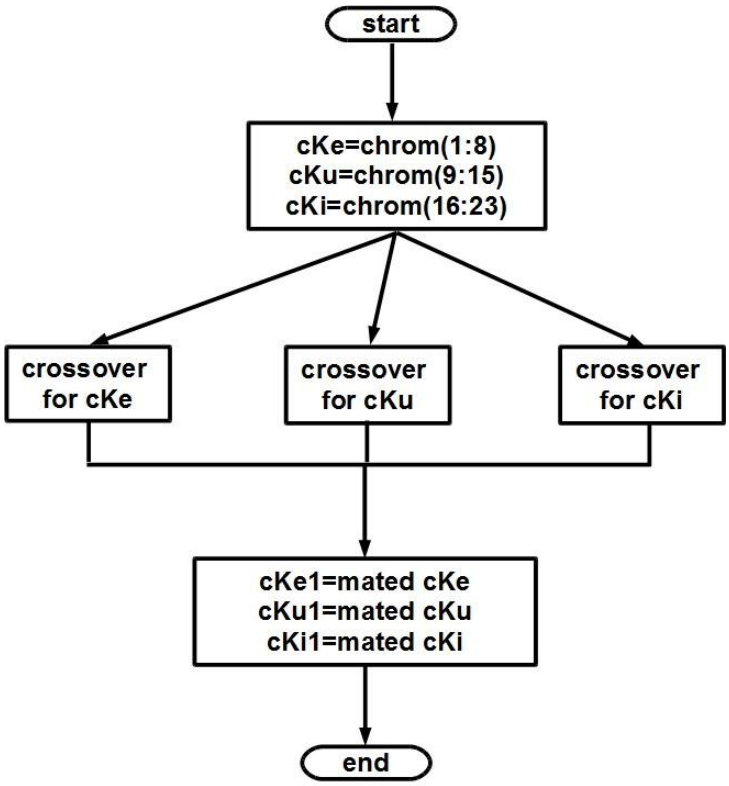


Fig. 3.47 Flowchart of crossover process in SPOGA process for I/O/ scales and integral constant

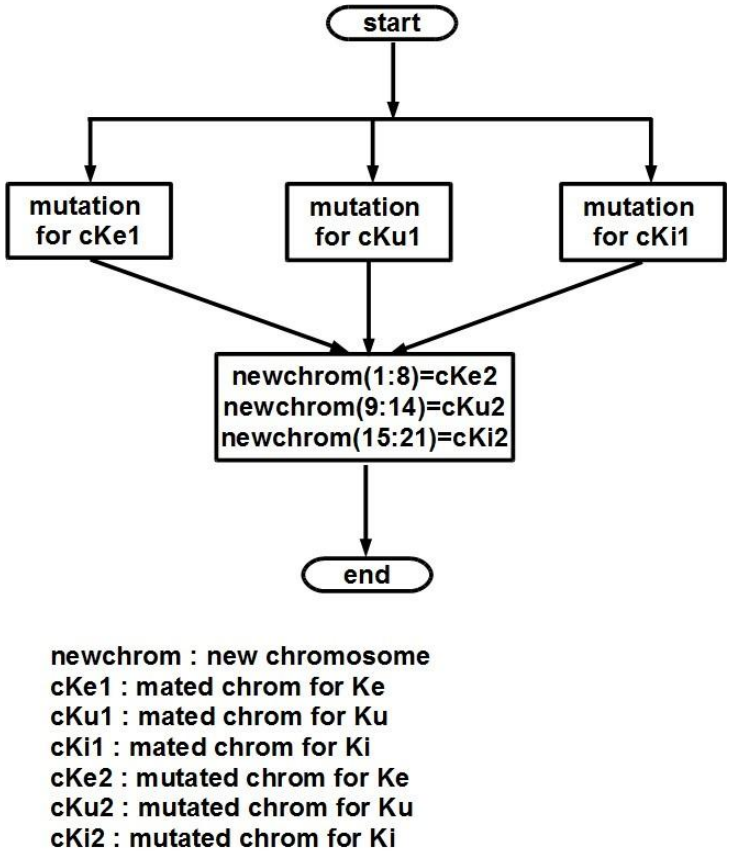


Fig. 3.48 Flowchart of mutation process in SPOGA process for I/O/ scales and integral constant

3.7 Performance Comparisons and Evaluations

There are two types of experiment to compare the nine types of controller as explained in Section 3.5. The types of experiment are simulation and hardware experiments and are divided into four conditions as follows:

1. Extreme conditions:
   a. Experiment of speed and position controller with the set-point of 150 rpm, 6 rad, unloaded
   b. Experiment of speed and position controller with the set-point of 400 rpm, 0.5 rad, unloaded

2. Moderate conditions:

   Experiment of speed and position controller with the set-point of 275 rpm, 3.5 rad, unloaded

3. Variable load conditions:
   a. Experiment of speed and position controller with the set-point of 250 rpm, 5 rad, unloaded, full-loaded after 15 sec
   b. Experiment of speed and position controller with the set-point of 250 rpm, 5 rad, full-loaded, unloaded after 15 sec
4. Variable set-point:
   a. Experiment of speed controller with the variable set-point of speed in the range of [0.00, 400.00] rpm, unloaded
   b. Experiment of speed and position controller with the set-point of 275 rpm, 2 rad, 5 rad after 45 sec, unloaded

The set-points in the experiment of speed and position controllers with the set-point of 150 rpm, 6 rad, unloaded (type 1a) are shown in Fig. 3.49 and Fig. 3.50. The performance items used as a performance index for Fig. 3.49 are:

i. % overshoot ($\%O_s$)
ii. Settling time (t_s)
iii. $ITAE$ for the first 8 sec only since the open loop time constant of plant is 0.5 sec. Based on the Eq. (2 - 40), the formula is as follows:

$$ITAE_{vp} = t\int_0^8 |SP(t) - PV(t)|dt \qquad (3 - 24)$$

The performance items used as a performance index for Fig. 3.50 are:

i. % steady state error for position ($\%S_p$)
ii. $ITAE$ for overall 90 seconds. Based on the Eq. (2 - 40), the formula is as follows:

$$ITAE_p = t\int_0^{90} |SP(t) - PV(t)|dt \qquad (3 - 25)$$

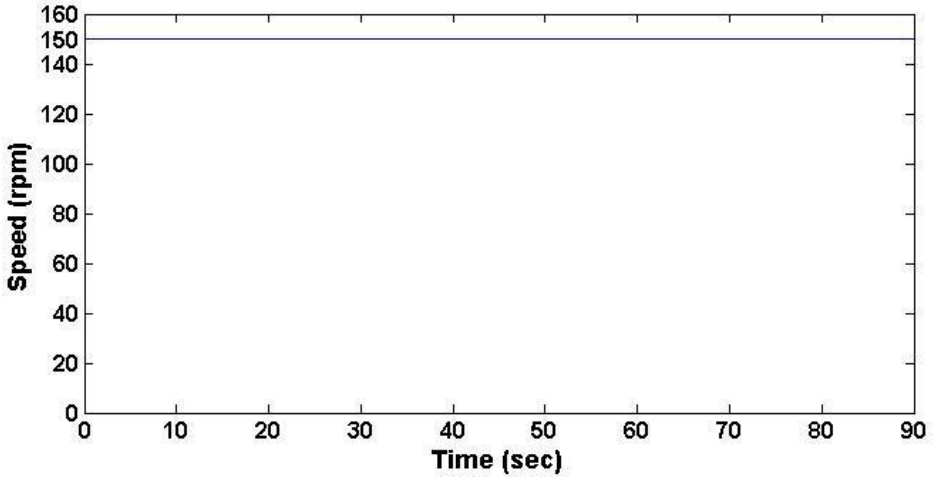


Fig. 3.49 Set-point of speed in the experiment of type 1a.

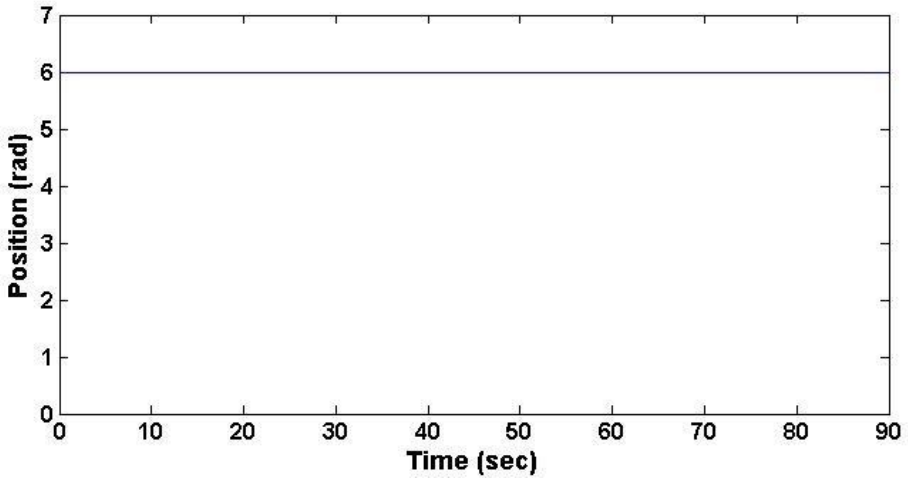


Fig. 3.50 Set-point of position in the experiment of type 1a.

As implied in Eq. (3 - 24) and Eq. (3 - 25), the best is the minimum value. Making a comparison is better to use a number which is easy to understand and assigned it as the best is the maximum value. Consequently, Eq. (3 - 24)and Eq. (3 - 25) can be changed to fitness functions based on Eq. (2 - 1). In the experiment of type 1a, f_{vp} is the fitness function for the first 8-second starting speed based on $ITAE_{vp}$, f_p is the fitness function for position based on $ITAE_p$, for N types of controller, are formulated as

$$f_{vp}(i) = \frac{\left(ITAE_{vp,max} - ITAE_{vp}(i)\right)}{\sum_{i=1}^{N}\left(ITAE_{vp,max} - ITAE_{vp}(i)\right)} \qquad (3 - 26)$$

$$f_p(i) = \frac{\left(ITAE_{p,max} - ITAE_p(i)\right)}{\sum_{i=1}^{N}\left(ITAE_{p,max} - ITAE_p(i)\right)} \quad (3\text{ - }27)$$

Based on Eq. (3 - 26) and Eq. (3 - 27) from Eq. (3 - 24) and Eq. (3 - 25), if f_{os} is the fitness function based on % overshoot, f_{ts} is the fitness function based on settling time, and f_{sp} is the steady state error for position control, then for N types of controller, the fitness functions can be formulated as follows:

$$f_{os}(i) = \frac{\left(\%O_{s,max} - \%O_s(i)\right)}{\sum_{i=1}^{N}\left(\%O_{s,max} - \%O_s(i)\right)} \quad (3\text{ - }28)$$

$$f_{ts}(i) = \frac{\left(t_{s,max} - t_s(i)\right)}{\sum_{i=1}^{N}\left(t_{s,max} - t_s(i)\right)} \quad (3\text{ - }29)$$

$$f_{sp}(i) = \frac{\left(S_{p,max} - S_p(i)\right)}{\sum_{i=1}^{N}\left(S_{p,max} - S_p(i)\right)} \quad (3\text{ - }30)$$

If fit_v is the total function fitness for speed control, fit_p is the total fitness function for position control, and fit_{1a} is the total fitness function for speed and position control in the experiment 1a, then these fitness functions can be formulated as follows:

$$fit_v = \frac{f_{vp} + f_{os} + f_{ts}}{3} \quad (3\text{ - }31)$$

$$fit_p = \frac{f_p + f_{sp}}{2} \quad (3\text{ - }32)$$

$$fit_{1a} = \frac{2fit_v + fit_p}{3} \quad (3\text{ - }33)$$

It is noted that Eq. (3 - 33) is built based on the fact that the SPOGA is not used for optimizing the position controller. Therefore, the speed fitness is more important than the position fitness.

The set-points in the experiment of speed and position controller with set-point of 400 rpm, 0.5 rad, unloaded (type 1b) are shown in Fig. 3.51 and Fig. 3.52. The are no performance items used as a performance index for Fig. 3.51. The performance items used as a performance index for Fig. 3.52 are:

i. % steady state error ($\%S_p$)

ii. $ITAE$ for overall 90 seconds with the formula as in Eq. (3 - 25).

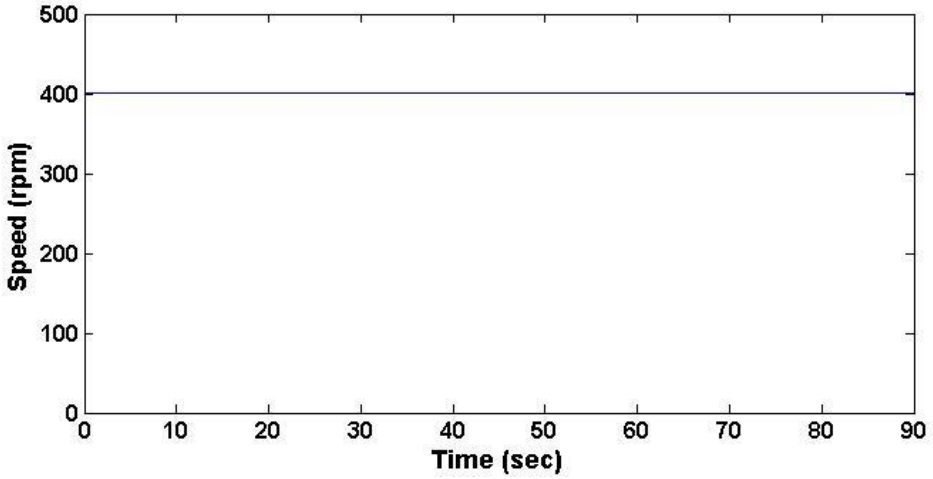


Fig. 3.51 Set-point of speed in the experiment of type 1b.

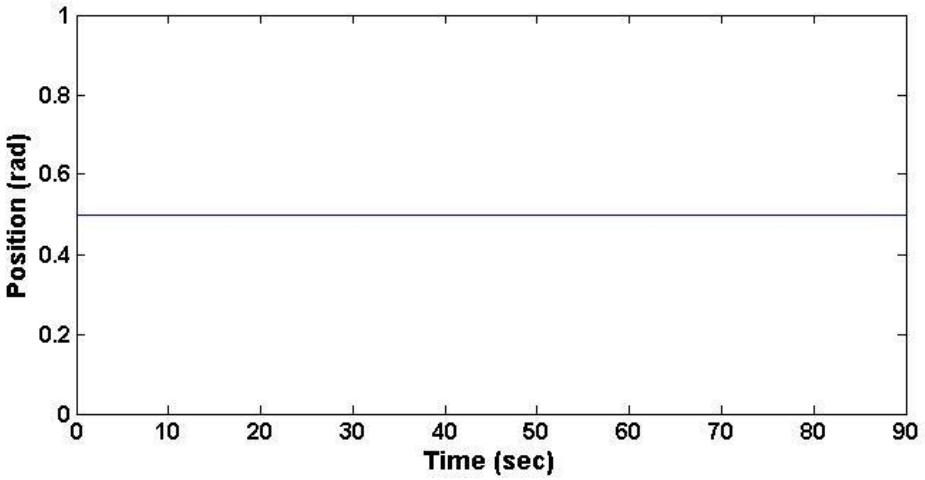


Fig. 3.52 Set-point of position in the experiment of type 1b.

The fitness function for position based on $ITAE_p$ is formulated as in Eq. (3 - 27). The total fitness function for position control is formulated as in Eq. (3 - 32). If fit_{1b} is the total fitness function for speed and position control in the experiment 1b, then it can be formulated as follows:

$$fit_{1b} = fit_p \qquad (3 - 34)$$

The set-points in the experiment of speed and position controller with set-point of 275 rpm, 3.5 rad, unloaded (type 2) are shown in Fig. 3.53 and Fig. 3.54. The performance items used as a performance index for Fig. 3.53 are:

i. % overshoot ($\%O_s$)

ii. Settling time (t_s)

iii. $ITAE$ for the first 8 sec with the formula as in Eq. (3 - 24).

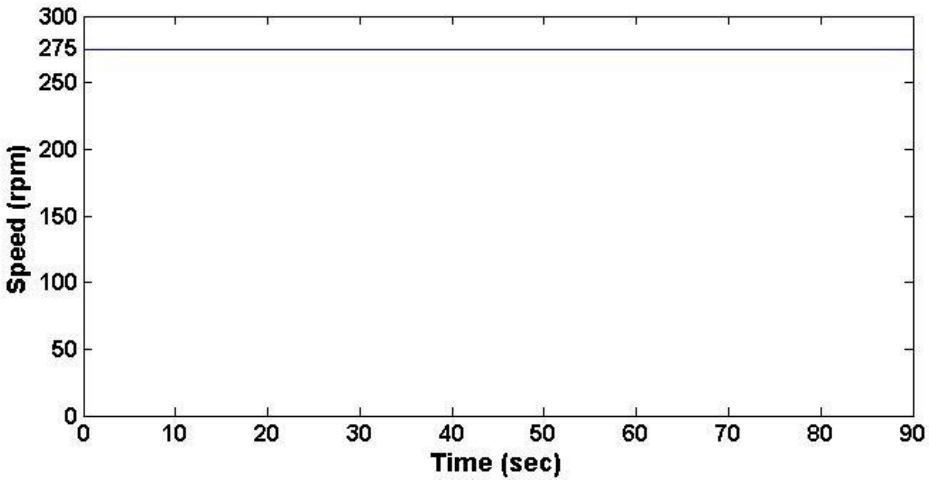


Fig. 3.53 Set-point of speed in the experiment of type 2 and 4b

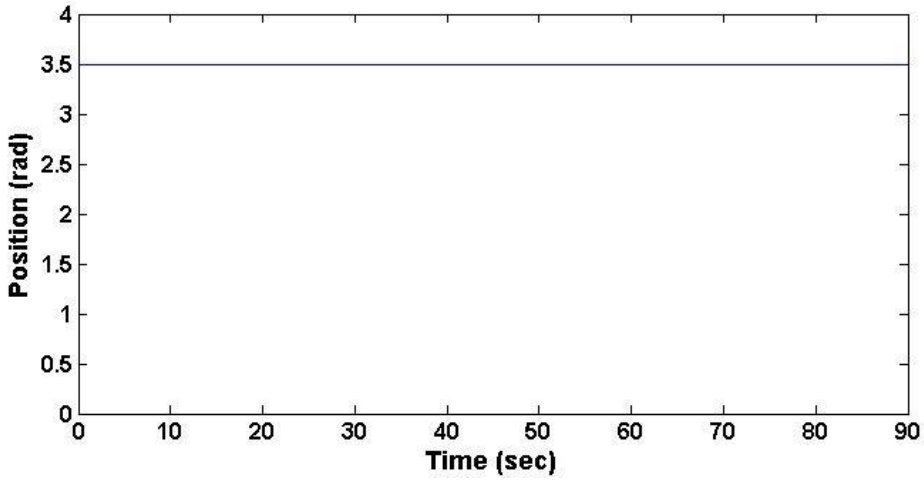


Fig. 3.54 Set-point of position in the experiment of type 2

The performance items used as a performance index for Fig. 3.54 are:

i. % steady state error for position (S_p)

ii. $ITAE$ for overall 90 seconds with the formula as in Eq. (3 - 25).

The fitness function for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) and for position based on $ITAE_p$ (f_p) are formulated as in Eq. (3 - 26) and Eq. (3 - 27) respectively. The fitness function based on $\%O_s$ (f_{os}), based on t_s (f_{ts}), and based on $\%S_p$ (f_{sp}) are formulated as in Eq. (3 - 28), Eq. (3 - 29), and Eq. (3 - 30) respectively.

The total fitness function for speed control (fit_v) and the total fitness function for position control (fit_p) are formulated as in Eq. (3 - 31) and Eq. (3 - 32) respectively. If fit_2 is the total fitness function for speed and position control in the experiment 2, then it can be formulated as follows:

$$fit_2 = \frac{2fit_v + fit_p}{3} \qquad (3 - 35)$$

The set-points in the experiment of speed and position controller with set-point of 250 rpm, 5 rad, unloaded, full-loaded after 15 sec (type 3a) are shown in Fig. 3.55 and Fig. 3.56. The performance items used as a performance index for Fig. 3.55 are:

i. % overshoot ($\%O_s$)

ii. Settling time (t_s)

iii. % undershoot ($\%U_s$)

iv. $ITAE$ for the first 8 sec with the formula as in Eq. (3 - 24).

v. $ITAE$ for the 9-sec start loading speed (from 15 to 24 sec) with the formula as follows:

$$ITAE_{vpl} = t \int_{15}^{24} |SP(t) - PV(t)|dt \qquad (3 - 36)$$

The performance items used as a performance index for Fig. 3.56 are:

i. % steady state error for position (S_p)

ii. $ITAE$ for overall 90 seconds with the formula as in Eq. (3 - 25).

The fitness function for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) and for position based on $ITAE_p$ (f_p) are formulated as in Eq. (3 - 26) and Eq. (3 - 27) respectively. The fitness function for the 9-sec start loading speed (from 14 to 23 sec) based on $ITAE_{vpl}$ for N types of controller is formulated as follows:

$$f_{vpl}(i) = \frac{\left(ITAE_{vpl,max} - ITAE_{vpl}(i)\right)}{\sum_{i=1}^{N}\left(ITAE_{vpl,max} - ITAE_{vpl}(i)\right)} \qquad (3 - 37)$$

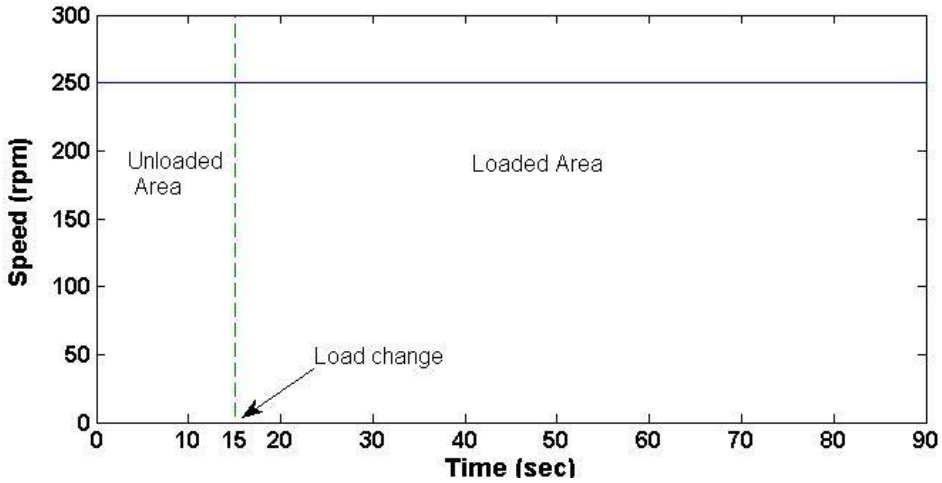


Fig. 3.55 Set-point of speed in the experiment of type 3a

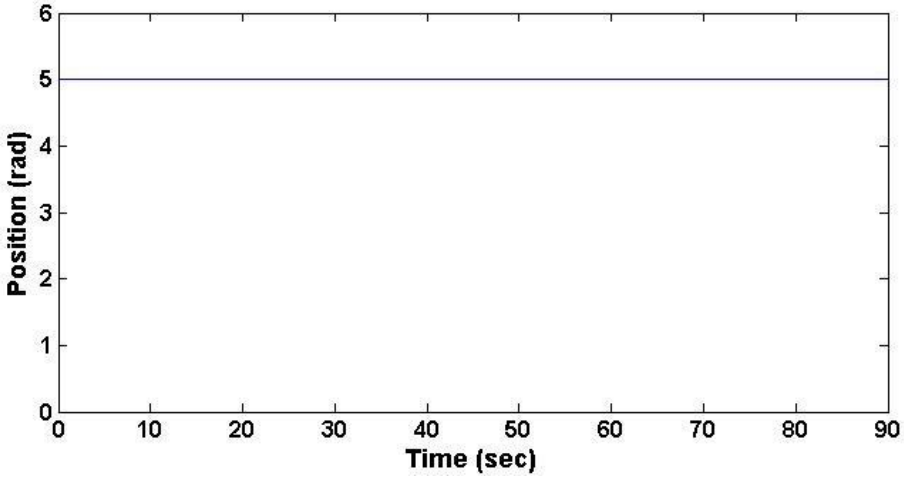


Fig. 3.56 Set-point of position in the experiment of type 3a and 3b

The total fitness function for speed control (fit_{v3}) is formulated as follows:

$$fit_{v3} = \frac{f_{vp} + f_{os} + f_{us} + f_{vpl} + f_{ts}}{5} \quad (3 - 38)$$

and the total fitness function for position control (fit_p) is formulated as in Eq. (3 - 32). If fit_{3a} is the total fitness function for speed and position control in the experiment 3a, then it can be formulated as follows:

$$fit_{3a} = \frac{2fit_{v3} + fit_p}{3} \quad (3 - 39)$$

The set-points in the experiment of speed and position controller with set-point of 250 rpm, 5 rad, full-loaded, unloaded after 15 sec (type 3b) are shown in Fig. 3.57 and Fig. 3.56. The performance items used as a performance index for Fig. 3.57 are:

i. % overshoot ($\%O_s$)
ii. Settling time (t_s)
iii. % overshoot when start unloading ($\%O_{s2}$)
iv. $ITAE$ for the first 8 sec with the formula as in Eq. (3 - 24).
v. $ITAE$ for the 9-sec start unloading speed (from 15 to 24 sec) with the formula as in Eq. (3 - 36).

The performance items used as a performance index for Fig. 3.56 are:

i. % steady state error for position (S_p)
ii. $ITAE$ for overall 90 seconds with the formula as in Eq. (3 - 25).

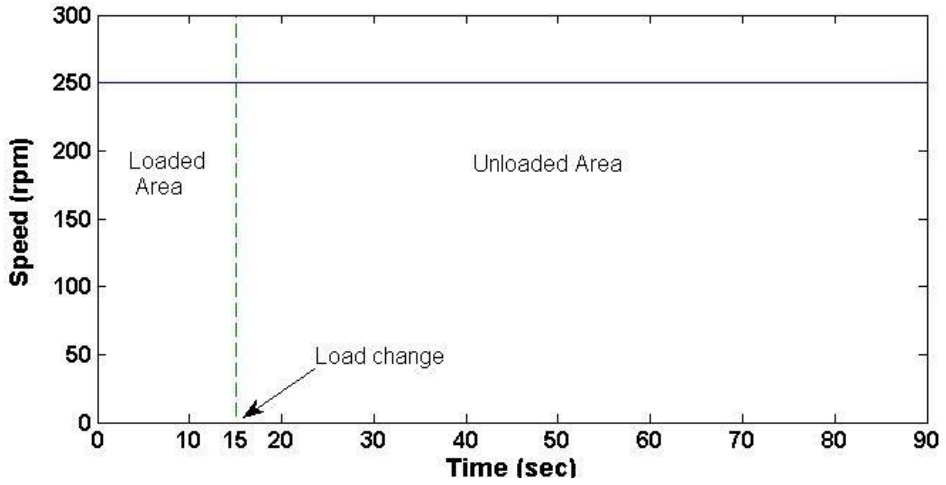


Fig. 3.57 Set-point of speed in the experiment of type 3b

The fitness function for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) and for position based on $ITAE_p$ (f_p) are formulated as in Eq. (3 - 26) and Eq. (3 - 27) respectively. The fitness function for the 9-sec start loading speed (from 14 to 23 sec) based on $ITAE_{vpl}$ is formulated as in Eq. (3 - 37).

The total fitness function for speed control (fit_{v3}) and the total fitness function for position control (fit_p) is formulated as in Eq. (3 - 38) and Eq. (3 - 32) respectively. If fit_{3b} is the total fitness function for speed and position control in the experiment 3b, then it can be formulated as follows:

$$f_{3b} = \frac{2fit_{v3} + fit_p}{3} \qquad (3 - 40)$$

The set-points in the experiment of speed and position controller with variations of speed, unloaded (type 4a) is shown in Fig. 3.58. The performance items used as a performance index is IAE with the formula as follows:

$$IAE_v = \int_0^{90} |SP(t) - PV(t)|dt \qquad (3 - 41)$$

The fitness function for speed (f_v) is formulated as follows:

$$f_v(i) = \frac{\left(IAE_{v,max} - IAE_v(i)\right)}{\sum_{i=1}^{9}\left(IAE_{v,max} - IAE_v(i)\right)} \qquad (3 - 42)$$

and the overall fitness function for the experiment of type 4a is as follows:

$$fit_{4a} = f_v \qquad (3 - 43)$$

The set-points in the experiment of speed and position controller with set-point of 275 rpm, 2 rad, 5 rad after 45 sec, unloaded (type 4b) are shown in Fig. 3.53 and Fig. 3.59. The performance items used as a performance index for Fig. 3.53 are:

i. % overshoot ($\%O_s$)
ii. Settling time (t_s)
iii. $ITAE$ for the first 8 sec with the formula as in Eq. (3 - 24).

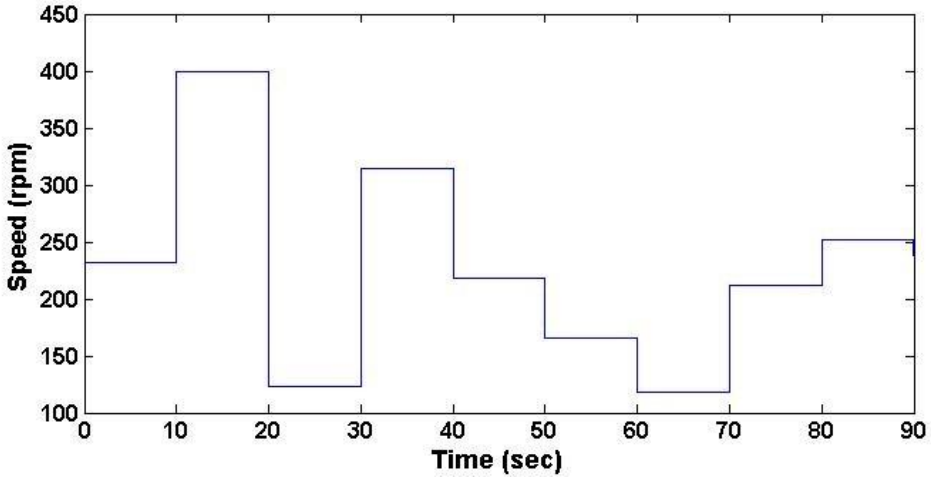


Fig. 3.58 Set-point of speed in the experiment of type 4a

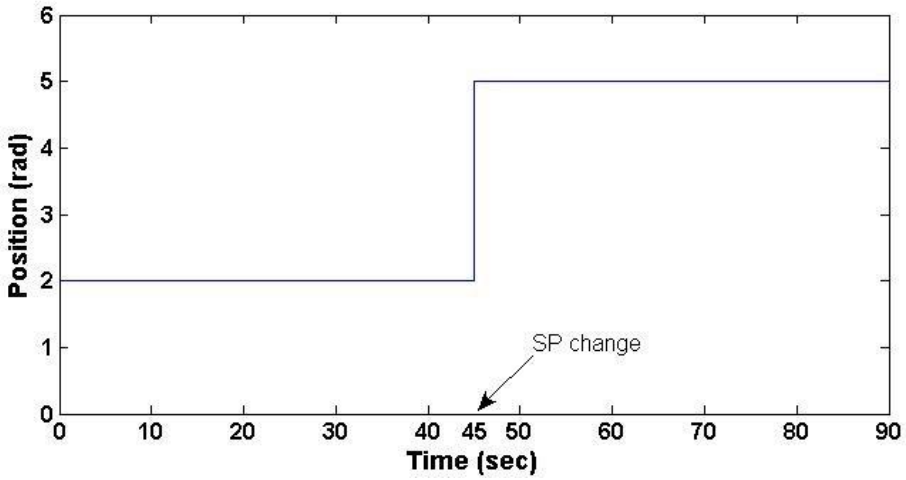


Fig. 3.59 Set-point of position in the experiment of type 4b

The performance items used as a performance index for Fig. 3.59 are:

i. % steady state error for position (S_p)

ii. $ITAE$ for overall 90 seconds with the formula as in Eq. (3 - 25).

The fitness function for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) and for position based on $ITAE_p$ (f_p) are formulated as in Eq. (3 - 26) and Eq. (3 - 27) respectively. The fitness function based on $\%O_s$ (f_{os}), based on t_s (f_{ts}), and based on $\%S_p$ (f_{sp}) are formulated as in Eq. (3 - 28), Eq. (3 - 29), and Eq. (3 - 30) respectively.

The total fitness function for speed control (fit_v) and the total fitness function for position control (fit_p) are formulated as in Eq. (3 - 31) and Eq. (3 - 32) respectively. If fit_{4b} is the total fitness function for speed and position control in the experiment 4b, then it can be formulated as follows:

$$fit_{4b} = \frac{2fit_v + fit_p}{3} \quad (3 - 44)$$

The overall fitness function for all experiments can be formulated as:

$$fit = fit_{1a} + fit_{1b} + fit_2 + fit_{3a} + fit_{3b} + fit_{4a} + fit_{4b} \quad (3 - 45)$$

The maximum value of *fit* is the best performance.

If $fit_{vx,g}$ is the total fitness function of SPOGA-optimized hybrid controller for speed control in the experiment x and $fit_{vx,h}$ is the total fitness function of non-SPOGA-optimized hybrid controller for speed control in the experiment x where $x = \{1a, 1b, 2, 3a, 3b, 4a, 4b\}$, then the improvement value for speed control in the experiment x, I_{pvx}, is formulated as follows:

$$I_{pvx} = fit_{vx,g} - fit_{vx,h} \quad (3 - 46)$$

The overall improvement value for speed for all experiments can be formulated as:

$$I_{pv} = I_{pv1a} + I_{pv1b} + I_{pv2} + I_{pv3a} + I_{pv3b} + I_{pv4a} + I_{pv4b} \quad (3 - 47)$$

3.8 Summary

The detailed methodology of experiments has been presented in this chapter. There were two types of experiments: simulation and hardware experiments. The simulation experiments were conducted to predict the characteristic and the performance of controllers applied to the DC servomotor. Running the GA/SPOGA in the optimization process is much better on the simulation experiment mode than on the

hardware experiment mode. Identification of DC servomotor was conducted to run on the simulation mode by obtaining the transfer function in *s*-domain.

The results of simulation experiment, gray box *s*-modeling of hardware, GA/SPOGA simulation, and SPOGA optimization will be presented in Chapter 4. The results of hardware experiment and experiment of hardware specification will be presented in Chapter 5. The performance comparisons will be presented in Chapter 4 for simulation results and Chapter 5 for hardware experiment results.

CHAPTER 4

SIMULATION RESULTS AND DISCUSSIONS

4.1 Introduction

The design of controllers and algorithms which are then verified via simulations and real-time implementations have been presented in Chapter 3. The results of experiments on *s*-modelling, GA and SPOGA, and the simulations of speed and position control will be presented in this chapter, and the real-time implementation will be presented in Chapter 5.

Performing simulations of speed and position controllers need a transfer function that is to be obtained from *s*-modelling of a servomotor system. The simulation results of conventional and fuzzy logic controllers are then obtained, followed by the simulation results of hybrid-fuzzy controllers. The best conventional and fuzzy logic controllers are compared graphically with the best hybrid-fuzzy controllers.

Simulation of GA and SPOGA are done to compare the performance and efficiency. Either one is then selected based on the performance and efficiency, and then applied to optimize the hybrid-fuzzy controllers using the specifications based on the result of simulation.

The optimized controllers are evaluated via simulation and the performance indices were recorded. Note that at this point, the controllers are optimized using GA or SPOGA and their performances are going to be evaluated. The notation of GA/SPOGA is being used here to indicate the exercise conducted. The performance of GA/SPOGA optimized controllers are then compared graphically with non-GA/SPOGA optimized controllers to the improvement value of the GA/SPOGA over the non-GA/SPOGA optimized controllers. One of the GA/SPOGA optimized

hybrid-fuzzy controllers is selected as the best controller based on the improvement values attained.

4.2 Input-Output Modeling of A DC Servomotor

The process of *s*-modeling is shown in Fig. 3.8. There are five types of input sequences, namely Type 1, Type 2, Type 3, Type 4, and Type 5. The result of the best model of each type of input sequence is shown in Table 4.1.

Table 4.1 The best model of each type of input sequence

NO.	INPUT SEQUENCE	BEST MODEL	FIT
1	Type 1	$\frac{456.3713e^{-0.1682s}}{s^3+9.504s^2+80.7s+204.5}$	86.59
2	Type 2	$\frac{9.7174e^{-0.2077s}}{s+4.4293}$	79.47
3	Type 3	$\frac{151.9520e^{-0.1503s}}{s^2+18.8651s+67.8296}$	80.47
4	Type 4	$\frac{465.5417e^{-0.0302s}}{s^3+8.981s^2+85.05s+214.2}$	78.99
5	Type 5	$\frac{210.9704}{s^3+6.639s^2+49.64s+94.91}$	85.64

Based on Table 4.1, the best model is indicated by the largest value of FIT which is obtained from Eq. (2 - 22), as follows:

$$G(s)=\frac{456.3713e^{-0.1682s}}{s^3+9.504s^2+80.7s+204.5} \qquad (4-1)$$

The graphical comparison of actual (real time) and estimation (model) based on eq. (4 - 1) is shown in Fig. 4.1. Based on this figure, the average deviation between actual and estimation is 0.78 % for **IN1**, 2.00 % for **IN2**, and 4.03 % for overall. These values are good enough as a verification result. Eq (4 - 1) is then used as the plant transfer function in all of the simulation experiments.

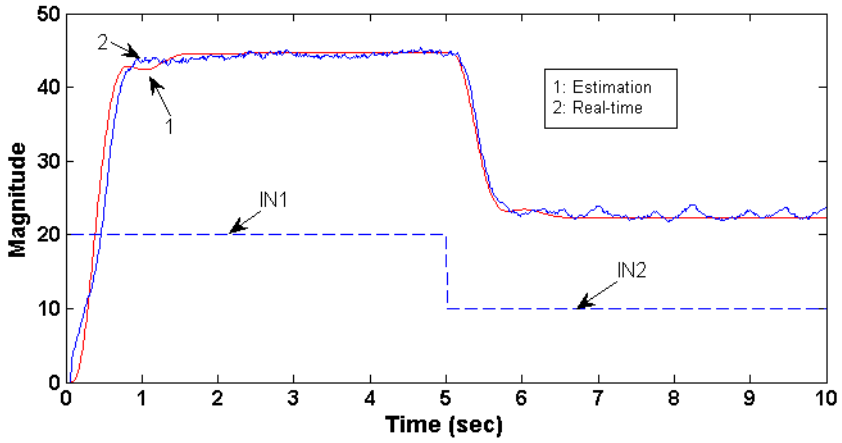


Fig. 4.1 Graphical verification of input-output modelling of a DC servomotor

4.3 Simulation of Conventional and Fuzzy Controllers

There are three types of controllers in this simulation: PI controller, PID controller, and fuzzy logic controller (FLC). The simulation is done as a basis of controllers in GA application (especially for FLC) and in measuring the performance improvement of GA optimized controllers (especially for PI and PID).

Based on Eq. (2 - 31) and Eq. (2 - 32), the experiment of ultimate cycle in tuning PI and PID gives the result as follows:

$K_{P\text{-}u}$ = 23.5, T_u = 10 sec

Using Eq. (2 - 31), the parameters of PI after fine-tuning are as shown below:

K_P = 10.58, K_I = 14.7

Using Eq. (2 - 32), the parameters of PID after fine-tuning are as follow:

K_P = 14.1, K_D = 0.1, K_I = 24.4

Experimentally, the proportional constant for position controller (K_{Pp}) is 0.61.

The membership functions of FLC in the simulation of speed controller experiment are shown in Fig. 3.20 and Fig. 3.21, the rules are shown in Table 3.3, and the output is integrated using Eq. (3 - 16).

The membership functions of FLC in the simulation of position controller experiment are shown in Fig. 3.18 and Fig. 3.19, the rules are shown in Table 3.2, and

the output is directly applied to the next stage. The output is multiplied by an adjustable constant (K_{Pp}) and experimentally, the value is 0.35.

4.3.1 Description on Types of Simulations

There are seven types of simulation experiment as explained in Section 3.7 which are classified into four condition, i.e.:

i. Extreme condition, namely Simulation 1a and 1b
ii. Moderate condition, namely Simulation 2
iii. Variable load condition, namely Simulation 3a and 3b
iv. Variable set point condition, namely Simulation 4a and 4b

The types of simulation are summarized in Table 4.2.

The number of simulation types are larger than that in Table 4.2, however most would fall into the listed conditions, hence the simulation and analysis are limited only to these condition.

For the different controllers that will be applied in the system, their performance will be taken for analysis and comparison based on the condition as specified in Table 4.2.

Table 4.2 Types of simulation

Type	Simulation Condition	Specifications	
1a	Extreme a	Set point of speed (rpm) Set point of position (rad) Loaded	150.00 6.00 No
1b	Extreme b	Set point of speed (rpm) Set point of position (rad) Loaded	400.00 0.50 No
2	Moderate	Set point of speed (rpm) Set point of position (rad) Loaded	275.00 3.50 No
3a	Variable load a	Set point of speed (rpm) Set point of position (rad) Loaded	250.00 5.00 After 15 sec
3b	Variable load b	Set point of speed (rpm) Set point of position (rad) Loaded	250.00 5.00 Up to 15 sec
4a	Variable set point a	Set point of speed (rpm) Set point of position (rad) Loaded	variable [0.00, 400.00] - No
4b	Variable set point b	Set point of speed (rpm) Set point of position (rad) Loaded	275.00 2 (up to 45 sec), 5 (after 45 sec) No

4.3.2 Performance Comparisons of Conventional and Fuzzy Controllers

The comparison on the effectiveness of implementing conventional and fuzzy logic controllers for simulation experiment based on the second order underdamped response analysis is presented in Table 4.3, the comparison based on error analysis is presented in Table 4.4, and the comparison based on fitness value analysis is presented in Table 4.5.

It is shown in Table 4.3, Table 4.4, and Table 4.5 in the simulation 1a that the best overshoot and settling time for speed control is PI controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is found to be the PI controller.

In the position control, the best SSEP is PI controller, but the best fitness value based on $ITAE_p$ (f_p) is FLC. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PI controller.

Table 4.3 Simulation results of conventional and fuzzy logic controllers based on second order underdamped response analysis

Type	PERFORMANCE ITEM	Controller		
		PI	PID	FLC
1a	Overshoot ($\%O_S$, %)	0.00	18.07	24.53
	Settling time (t_s, sec)	4.10	4.85	9.91
	SSEP ($\%S_p$, %)	0.13	0.19	0.32
1b	SSEP ($\%S_{p,\%}$)	24.50	13.46	138.67
2	Overshoot ($\%O_S$, %)	0.00	18.06	13.22
	Settling time (t_s, sec)	4.10	4.85	5.93
	SSEP ($\%S_p$, %)	0.35	0.04	0.10
3a	Overshoot ($\%O_S$, %)	0.00	18.07	15.45
	Settling time (t_s, sec)	4.10	4.85	7.69
	Undershoot ($\%U_S$, %)	39.60	39.96	41.69
	SSEP ($\%S_p$, %)	0.55	0.04	0.02
3b	Overshoot ($\%O_S$, %)	0.00	0.00	2.23
	Settling time (t_s, sec)	5.31	3.52	4.26
	Overshoot 2 ($\%O_{S2}$, %)	66.32	66.76	62.66
	SSEP ($\%S_p$, %)	0.02	0.08	0.00
4b	Overshoot ($\%O_S$, %)	0.00	18.06	13.22
	Settling time (t_s, sec)	4.10	4.85	5.93
	SSEP ($\%S_p$, %)	0.02	0.08	0.04

SSEP: Steady State Error of Position

Table 4.4 Simulation results of conventional and fuzzy logic controllers based on error analysis

Type	PERFORMANCE ITEM	Controller		
		PI	PID	FLC
1a	$ITAE_{vp}$	1.14E+02	1.09E+02	2.44E+02
	$ITAE_p$	6.04E+03	5.97E+03	5.66E+03
1b	$ITAE_p$	4.96E+02	2.73E+02	2.80E+03
2	$ITAE_{vp}$	2.10E+02	2.00E+02	4.51E+02
	$ITAE_p$	4.26E+02	3.77E+02	4.40E+02
3a	$ITAE_{vp}$	1.91E+02	1.81E+02	3.80E+02
	$ITAE_{vl}$	2.06E+03	1.26E+03	2.06E+03
	$ITAE_p$	1.43E+03	1.30E+03	1.42E+03
3b	$ITAE_{vp}$	4.15E+02	1.72E+02	5.19E+02
	$ITAE_{vl}$	2.09E+03	1.88E+03	5.23E+03
	$ITAE_p$	1.32E+03	1.29E+03	1.36E+03
4a	IAE_v	9.44E+02	8.31E+02	2.62E+03
4b	$ITAE_{vp}$	2.10E+02	2.00E+02	4.51E+02
	$ITAE_p$	1.84E+03	1.81E+03	1.96E+03

$ITAE_{vp}$: Integral of time absolute value of error for the first 8 sec; $ITAE_p$: ITAE for position, $ITAE_{vl}$: ITAE for the 9-sec start loading speed; IAE_v : integral of absolute value of error for overall 90 sec.

The best total fitness value for speed and position control in the simulation 1a (f_{1a}) is PI controller. Therefore, PI is the best conventional controller in simulation 1a which is shown to be better than FLC.

Table 4.5 Simulation results of conventional and fuzzy logic controllers based on fitness value analysis

Type	PERFORMANCE ITEM	Controller		
		PI	PID	FLC
1a	f_{vp}	0.49	0.51	0.00
	f_p	0.00	0.15	0.85
	fit_v	0.61	0.39	0.00
	fit_p	0.29	0.28	0.43
	fit_{1a}	0.50	0.36	0.14
1b	f_p	0.48	0.52	0.00
	fit_p	0.48	0.52	0.00
	fit_{1b}	0.48	0.52	0.00
2	f_{vp}	0.49	0.51	0.00
	f_p	0.18	0.82	0.00
	fit_v	0.63	0.29	0.07
	fit_p	0.09	0.69	0.22
	fit_2	0.45	0.43	0.12
3a	f_{vp}	0.49	0.51	0.00
	f_{vpl}	0.00	0.10	0.00
	f_p	0.00	0.97	0.03
	fit_v	0.48	0.49	0.03
	fit_p	0.00	0.73	0.27
	fit_{3a}	0.32	0.57	0.11
3b	f_{vp}	0.23	0.77	0.00
	f_{vpl}	0.00	0.00	0.00
	f_p	0.41	0.59	0.00
	fit_v	0.18	0.47	0.09
	fit_p	0.42	0.30	0.28
	fit_{3b}	0.26	0.41	0.16
4a	f_v	0.48	0.52	0.00
	fit_v	0.48	0.52	0.00
	fit_{4a}	0.48	0.52	0.00
4b	f_{vp}	0.49	0.51	0.00
	f_p	0.44	0.56	0.00
	fit_v	0.63	0.29	0.07
	fit_p	0.52	0.28	0.20
	fit_{4b}	0.60	0.29	0.11

f_{vp} : Fitness value for the first 8-sec starting speed based on $ITAE_{vp}$; f_p : Fitness value for position based on $ITAE_p$; f_{vpl} : Fitness value for the 9-sec start loading speed (from 14 to 23 sec) based on $ITAE_{vpl}$; f_v: Fitness value for overall 90 sec of speed based on IAE_v; fit_v : Total fitness value for speed control; fit_p : Total fitness value for position control; fit_x : Total fitness value for speed and position control in the experiment of Type x

The simulation 1b presents the position performance since it is in the extreme condition with maximum speed and minimum position. The best fitness value based on $ITAE_p$ (f_p) is PID. The best total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PID controller.

The best total fitness value for speed and position control in the simulation 1b (f_{1b}) is PID controller. Therefore, PID is the best conventional controller in the simulation 1b which is observed to be better than FLC.

The best overshoot and settling time in the simulation 2 for speed control is PI controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best one is the PI controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is PID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best one is PID controller.

The best total fitness value for speed and position control in the simulation 2 (f_2) is PI controller. Therefore, PI is the best conventional controller in the simulation 2 which is better than FLC in term of position control. In term of speed control, PI is the best for simulation 2.

The best overshoot and settling time in the simulation 3a for speed control is PI controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, undershoot, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is the PID controller. When start loading, the best undershoot is the PI controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is PID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PID controller.

The best total fitness value for speed and position control in the simulation 3a (f_{3a}) is PID controller. Therefore, PID is the best conventional controller in the simulation 3a which is better than FLC.

In the simulation 3b, the best overshoot and settling time, and the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) for speed control is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, overshoot 2, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is the PID controller. All of the overshoot 2 when start unloading are more than 50%. Consequently, the fitness value for 9-sec start unloading speed based on $ITAE_{vpl}$ (f_{vpl}) are zero.

In the position control, the best SSEP is FLC but the best fitness value based on $ITAE_p$ (f_p) is PI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PI controller.

The best total fitness value for speed and position control in the simulation 3b (f_{3b}) is PID controller. Therefore, PID is the best conventional controller in the simulation 3b which is shown to be better than FLC.

The simulation 4a presents the speed performance without overshoot and settling time since it is in the variations of speed set point. The best fitness value based on IAE_v (f_v) is PID controller. The best total fitness value for speed control (fit_v) is obtained based on the IAE_v only, and the best is the PID controller.

The best total fitness value for speed and position control in the simulation 4a (f_{4a}) is PID controller. Therefore, PID is the best conventional controller in the simulation 4a which is better than FLC.

The best overshoot and settling time in the simulation 4b for speed control is PI controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the PI controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is PI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PI controller.

The best total fitness value for speed and position control in the simulation 4b (f_{4b}) is PI controller. Therefore, PI is the best conventional controller in the simulation 4b which is better than FLC.

4.3.3 Simulation Results Summary of Conventional and Fuzzy Controllers

The speed and position control simulation of the conventional and fuzzy logic controller have been presented. PI controller is the best speed controller as compared to PID controller and FLC for the simulation of Type 1a, 2, and 4b. PID controller is

the best speed controller as compared to PI controller and FLC for the simulation of Type 3a, 3b, and 4a. Using Eq.(3 - 45), the best overall speed controller is PID as compared to PI and fuzzy logic. In the application of speed control, FLC is not as good as the conventional controllers.

In both speed and position control, PI is the best controller as compared to PID controller and FLC for the simulation of Type 1a, 2 and 4b. PID controller is the best controller as compared to PI controller and FLC for the simulation of Type 1b, 3a, 3b, and 4a. Using Eq. (3 - 45), the best overall speed and position controller is PI as compared to PID and fuzzy logic. In the application of speed and position control, FLC is not as good as conventional controller because in the standalone condition, the output of FLC has to be in integration mode to make the zero steady state error. This makes the time response slow and tends to be unstable.

4.4 Simulation of Hybrid-Fuzzy Controllers

There are three types of controllers in the simulation: FLBPI controller, FLBPID controller, and FLIC. The simulation was conducted as an improvement of PID and FLC based on the results that the performance of standard FLC is not as good as PID or PI controller.

The membership functions of FLBPI and FLBPID in the simulation of speed controller experiment are shown in Fig. 3.23, the rules are shown in Table 3.4, and the output is connected to PI for FLBPI and PID for FLBPID using Eq.(3 - 17) for FLBPI and Eq.(3 - 18) for FLBPID. Experimentally, the parameters of FLBPI and FLBPID are as follow:

K_{Pm} = 6 and K_{Im} = 79 for FLBPI

K_{Pm} = 10.4, K_{Dm} = 0.1, and K_{Im} = 66 for FLBPID

The K_{Dm} in FLBPID is taken from the result of PID tuning using Ziegler-Nichols (ultimate cycle).

The membership functions of FLIC in the simulation of speed controller experiment are shown in Fig. 3.20 and Fig. 3.25, the rules are shown in Table 3.3, and

the output is paralleled with integral controller. Experimentally, the parameters of FLIC are as follow:

K_e = 1.03, K_{ce} = 1, K_u = 0.25, K_I = 1.

The variable proportional controller is used as a position controller for hybrid speed controller where the value of K_{Pv} is as in Table 3.1. The values of K_{Pp} are experimentally as follow:

K_{Pp} = 0.98 for FLBPI and FLBPID controllers, and K_{Pp} = 0.75 for FLIC.

There are seven types of simulation as explained in Section 3.7 which are summarized in Table 4.2.

4.4.1 Performance Comparisons of Hybrid-Fuzzy Controllers

The comparison on the effectiveness of implementing hybrid-fuzzy controllers for simulation experiment based on the second order underdamped response analysis is presented in Table 4.6, the comparison based on error analysis is presented in Table 4.7, and the comparison based on fitness value analysis is presented in Table 4.8.

It is shown in Table 4.6, Table 4.7, and Table 4.8 in the simulation 1a that the best overshoot, settling time and fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is FLBPID controller for speed control. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the FLBPID controller.

In the position control, the best SSEP is FLBPID controller, but the best fitness value based on $ITAE_p$ (f_p) is FLIC. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLBPI controller.

The best total fitness value for speed and position control in the simulation 1a (f_{1a}) is FLBPID controller. Therefore, FLBPID is the best hybrid controller in the simulation 1a (extreme condition).

Table 4.6 Simulation results of hybrid-fuzzy controllers based on second order underdamped response analysis

Type	PERFORMANCE ITEM	Controller		
		FLBPI	FLBPID	FLIC
1a	Overshoot ($\%O_S$, %)	20.21	3.74	22.07
	Settling time (t_s, sec)	2.37	1.81	2.23
	SSEP ($\%S_p$, %)	0.08	0.08	0.11
1b	SSEP ($\%S_p$)	0.47	0.53	0.00
2	Overshoot ($\%O_S$, %)	8.99	3.55	2.27
	Settling time (t_s, sec)	2.57	1.74	1.44
	SSEP ($\%S_p$, %)	0.04	0.04	0.03
3a	Overshoot ($\%O_S$, %)	8.48	3.41	4.78
	Settling time (t_s, sec)	2.52	1.73	1.62
	Undershoot ($\%U_S$, %)	40.00	40.00	39.99
	SSEP ($\%S_p$, %)	0.05	0.01	0.01
3b	Overshoot ($\%O_S$, %)	1.54	0.06	0.00
	Settling time (t_s, sec)	1.45	2.23	3.09
	Overshoot 2 ($\%O_{S2}$, %)	66.67	66.67	66.67
	SSEP ($\%S_p$, %)	0.07	0.07	0.06
4b	Overshoot ($\%O_S$, %)	8.99	3.55	2.27
	Settling time (t_s, sec)	2.57	1.74	1.44
	SSEP ($\%S_p$, %)	0.07	0.07	0.06

SSEP: Steady State Error of Position

Table 4.7 Simulation results of hybrid-fuzzy controllers based on error analysis

Type	PERFORMANCE ITEM	Controller		
		FLBPI	FLBPID	FLIC
1a	$ITAE_{vp}$	4.96E+01	3.63E+01	4.79E+01
	$ITAE_p$	5.93E+03	5.97E+03	5.93E+03
1b	$ITAE_p$	4.42E+00	4.36E+00	4.88E+00
2	$ITAE_{vp}$	9.03E+01	8.01E+01	6.97E+01
	$ITAE_p$	3.77E+02	3.80E+02	3.78E+02
3a	$ITAE_{vp}$	7.74E+01	7.05E+01	6.28E+01
	$ITAE_{vl}$	1.03E+03	1.11E+03	1.50E+03
	$ITAE_p$	1.30E+03	1.30E+03	1.31E+03
3b	$ITAE_{vp}$	8.56E+01	1.29E+02	1.73E+02
	$ITAE_{vl}$	1.47E+03	1.38E+03	2.25E+03
	$ITAE_p$	1.27E+03	1.28E+03	1.29E+03
4a	IAE_v	7.13E+02	6.62E+02	9.17E+02
4b	$ITAE_{vp}$	9.03E+01	8.01E+01	6.97E+01
	$ITAE_p$	1.80E+03	1.81E+03	1.82E+03

$ITAE_{vp}$: Integral of time absolute value of error for the first 8 sec; $ITAE_p$: ITAE for position, $ITAE_{vl}$: ITAE for the 9-sec start loading speed; IAE_v : Integral of absolute value of error for overall 90 sec.

The simulation 1b presents the position performance since it is in the extreme condition with maximum speed and minimum position. The best SSEP is FLBPI but the best fitness value based on $ITAE_p$ (f_p) is FLIC. The best total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLIC.

Table 4.8 Simulation results of hybrid-fuzzy controllers based on fitness value analysis

Type	PERFORMANCE ITEM	Controller		
		FLBPI	FLBPID	FLIC
1a	f_{vp}	0.00	0.89	0.11
	f_p	0.47	0.00	0.53
	fit_v	0.03	0.87	0.10
	fit_p	0.48	0.25	0.27
	fit_{1a}	0.18	0.66	0.16
1b	f_p	0.00	0.06	0.27
	fit_p	0.24	0.26	0.49
	fit_{1b}	0.24	0.26	0.49
2	f_{vp}	0.00	0.33	0.67
	f_p	0.63	0.00	0.37
	fit_v	0.00	0.40	0.60
	fit_p	0.32	0.03	0.65
	fit_2	0.11	0.28	0.62
3a	f_{vp}	0.00	0.32	0.68
	f_{vpl}	0.55	0.45	0.00
	f_p	0.62	0.38	0.00
	fit_v	0.14	0.45	0.41
	fit_p	0.31	0.46	0.23
	fit_{3a}	0.19	0.46	0.35
3b	f_{vp}	0.67	0.33	0.00
	f_{vpl}	0.00	0.00	0.00
	f_p	0.70	0.30	0.00
	fit_v	0.45	0.42	0.13
	fit_p	0.35	0.25	0.40
	fit_{3b}	0.42	0.37	0.22
4a	f_v	0.45	0.56	0.00
	fit_v	0.45	0.56	0.00
	fit_{4a}	0.45	0.56	0.00
4b	f_{vp}	0.00	0.33	0.67
	f_p	0.67	0.33	0.00
	fit_v	0.00	0.40	0.60
	fit_p	0.33	0.26	0.41
	fit_{4b}	0.11	0.35	0.53

f_{vp} : Fitness value for the first 8-sec starting speed based on $ITAE_{vp}$; f_p : Fitness value for position based on $ITAE_p$; f_{vpl} : Fitness value for the 9-sec start loading speed (from 14 to 23 sec) based on $ITAE_{vpl}$; f_v: Fitness value for overall 90 sec of speed based on IAE_v; fit_v : Total fitness value for speed control; fit_p : Total fitness value for position control; fit_x : Total fitness value for speed and position control in the experiment of Type x

The best total fitness value for speed and position control in the simulation 1b (f_{1b}) is FLIC. Therefore, FLIC is the best hybrid controller in the simulation 1b (extreme condition).

The best overshoot in the simulation 2 for speed control is FLBPID but the best settling time and the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is FLIC. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the FLIC.

In the position control, the best SSEP is FLIC but the best fitness value based on $ITAE_p$ (f_p) is FLBPI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLIC.

The best total fitness value for speed and position control in the simulation 2 (f_2) is FLIC. Therefore, FLIC is the best hybrid controller in the simulation 2 (moderate condition).

The best overshoot and settling time in the simulation 3a for speed control is FLBPID controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is FLIC. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, undershoot, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is PID controller. When start loading, the best undershoot is FLBPID controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is FLBPI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLBPID controller.

The best total fitness value for speed and position control in the simulation 3a (f_{3a}) is FLBPID controller. Therefore, FLPID is the best hybrid controller in the simulation 3a (variable load condition).

The best overshoot in the simulation 3b for speed control is FLIC but the settling time and the fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is FLBPI controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, overshoot 2, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is FLBPI controller. All of the overshoot 2 when start unloading are more than 50%. Consequently, the fitness value for 9-sec start unloading speed based on $ITAE_{vpl}$ (f_{vpl}) are zero.

In the position control, the best SSEP is FLIC but the best fitness value based on $ITAE_p$ (f_p) is FLBPI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLIC.

The best total fitness value for speed and position control in the simulation 3b (f_{3b}) is FLBPI controller. Therefore, FLBPI is the best hybrid controller in the simulation 3b (variable load condition).

The simulation 4a presents the speed performance without overshoot and settling time since it is in the variations of speed set point. The best fitness value based on IAE_v (f_v) is FLBPID controller. The best total fitness value for speed control (fit_v) is obtained based on the IAE_v only, and the best is the FLBPID controller.

The best total fitness value for speed and position control in the simulation 4a (f_{4a}) is FLBPID controller. Therefore, FLBPID is the best hybrid controller in the simulation 4a (variable set point).

In the simulation 4b, the best overshoot, settling time, and fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) for speed control is FLIC. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the FLIC.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is FLIC. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLIC.

The best total fitness value for speed and position control in the simulation 4b (f_{4b}) is FLIC. Therefore, FLIC is the best hybrid controller in the simulation 4b (variable set point).

4.4.2 Simulation Results Summary of Hybrid-Fuzzy Controllers

The speed and position control simulation of hybrid-fuzzy controller has been presented. FLBPI controller is the best speed controller as compared to FLBPID controller and FLIC for simulation of Type 3b. FLBPID controller is the best speed controller as compared to FLBPI controller and FLIC for simulation of Type 1a, 3a, and 4a. FLIC is the best speed controller as compared to FLBPI controller and FLBPID controller for simulation of Type 2 and 4b. Using Eq.(3 - 45), the best overall speed controller is FLBPID as compared to FLBPI and FLIC.

In both speed and position control, FLBPI is the best controller as compared to FLBPID controller and FLIC for simulation of Type 3b. FLBPID controller is the best controller as compared to FLBPI controller and FLIC for simulation of Type 1a, 1b, 3a, and 4a. FLIC is the best controller as compared to FLBPI controller and FLBPID controller for simulation of Type 2 and 4b. Using Eq. (3 - 45), the best overall speed and position controller is FLBPID as compared to FLBPI and FLIC.

4.5 Performance Comparisons of Conventional, Fuzzy, and Hybrid-Fuzzy Controllers

This section presents the performance comparisons of conventional (PI and PID controllers), FLC, and hybrid-fuzzy controllers based on the fitness values where the performance items are based on Table 4.3 to Table 4.8.

There are seven types of simulation as explained in Section 3.7 which are summarized in Table 4.2.

4.5.1 Results on Performance Comparisons of Conventional, Fuzzy, and Hybrid-Fuzzy Controllers

The graphs of input-output characteristic of speed error, position error, speed, and position between the best conventional and fuzzy controllers and the best hybrid controllers are presented in the section for each simulation type.

The comparison on the effectiveness of implementing conventional, fuzzy, and hybrid controllers based on the performance metrics for simulation experiment is presented in Table 4.9 where the performance items are based on Table 4.5 and Table 4.8.

It is shown in the Table 4.9 that the best speed controller for simulation 1a is PI for conventional controller and FLBPID for hybrid-fuzzy controller, and the best position controller is FLC for conventional controller and FLBPI for hybrid-fuzzy controller. For speed and position control, PI is the best conventional controller and

FLBPID is the best hybrid controller. Comparison on the best conventional to the best hybrid are shown in the table that hybrid-fuzzy controllers are better then conventional controllers in the simulation 1a. The graphical comparisons are shown in the Fig. 4.2 to Fig. 4.5.

Table 4.9 Performance comparisons of conventional, fuzzy, and hybrid-fuzzy controllers

Type	**CONTROLLERS**	**Fitness**		
		fit_v	**fit_p**	**fit_x**
1a	PI	0.25	0.09	0.19
	PID	0.14	0.11	0.13
	FLC	0.00	0.26	0.09
	FLBPI	0.17	0.20	0.18
	FLBPID	0.28	0.17	0.24
	FLIC	0.16	0.18	0.17
1b	PI	-	0.17	0.17
	PID	-	0.19	0.19
	FLC	-	0.00	0.00
	FLBPI	-	0.21	0.21
	FLBPID	-	0.21	0.21
	FLIC	-	0.21	0.21
2	PI	0.19	0.03	0.13
	PID	0.08	0.22	0.13
	FLC	0.03	0.08	0.04
	FLBPI	0.20	0.22	0.21
	FLBPID	0.25	0.22	0.24
	FLIC	0.26	0.23	0.25
3a	PI	0.15	0.00	0.10
	PID	0.13	0.22	0.16
	FLC	0.01	0.10	0.04
	FLBPI	0.23	0.22	0.23
	FLBPID	0.26	0.23	0.25
	FLIC	0.22	0.22	0.22
3b	PI	0.07	0.24	0.13
	PID	0.15	0.10	0.13
	FLC	0.02	0.23	0.09
	FLBPI	0.17	0.15	0.16
	FLBPID	0.18	0.14	0.17
	FLIC	0.16	0.14	0.15
4a	PI	0.19	-	0.19
	PID	0.20	-	0.20
	FLC	0.00	-	0.00
	FLBPI	0.21	-	0.21
	FLBPID	0.22	-	0.22
	FLIC	0.19	-	0.19
4b	PI	0.20	0.30	0.23
	PID	0.08	0.10	0.09
	FLC	0.03	0.15	0.07
	FLBPI	0.20	0.15	0.18
	FLBPID	0.25	0.14	0.21
	FLIC	0.26	0.15	0.23

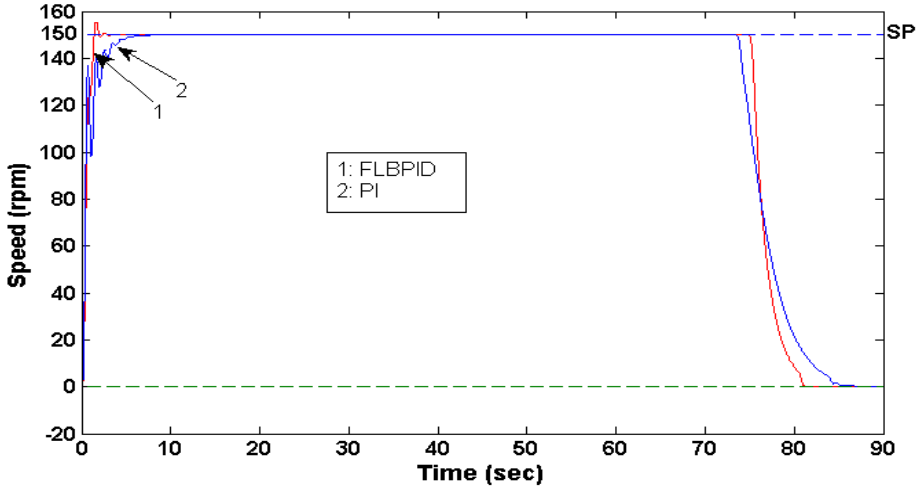


Fig. 4.2 Speed control of DC servomotor using FLBPID vs. PI for simulation 1a

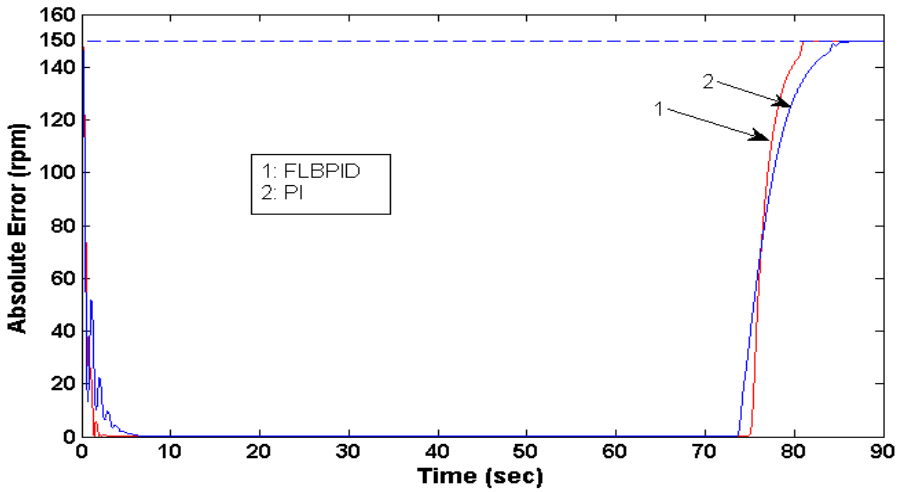


Fig. 4.3 Absolute error of speed control of DC servomotor using FLBPID vs. PI for simulation 1a

It is shown in Fig. 4.2 and Fig. 4.3 that FLBPID has the faster settling time than PI eventhough there is a very small overshoot. Consequently, the absolute error of FLBPID is smaller than PI.

It is shown in Fig. 4.4 and Fig. 4.5 that FLBPI has the faster rise time although the settling time is similar to the FLC. This makes the absolute error of FLBPI smaller than FLC.

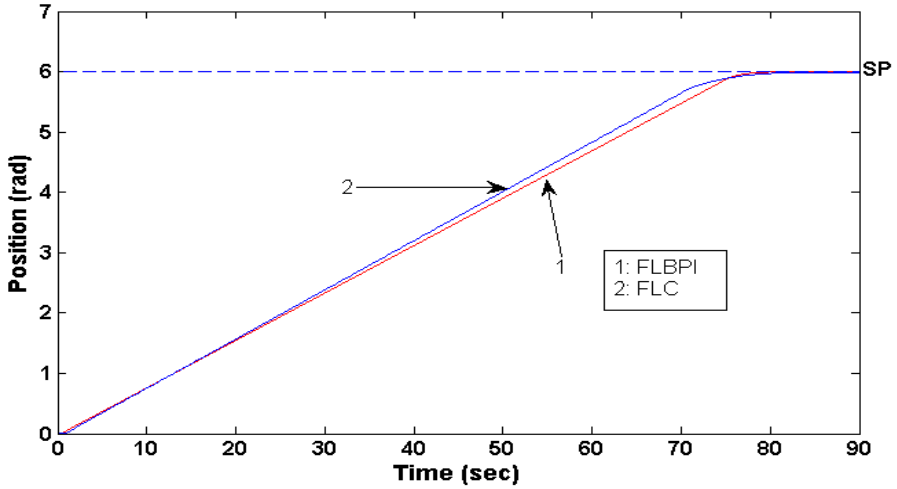


Fig. 4.4 Position control of DC servomotor using FLBPI vs. FLC for simulation 1a

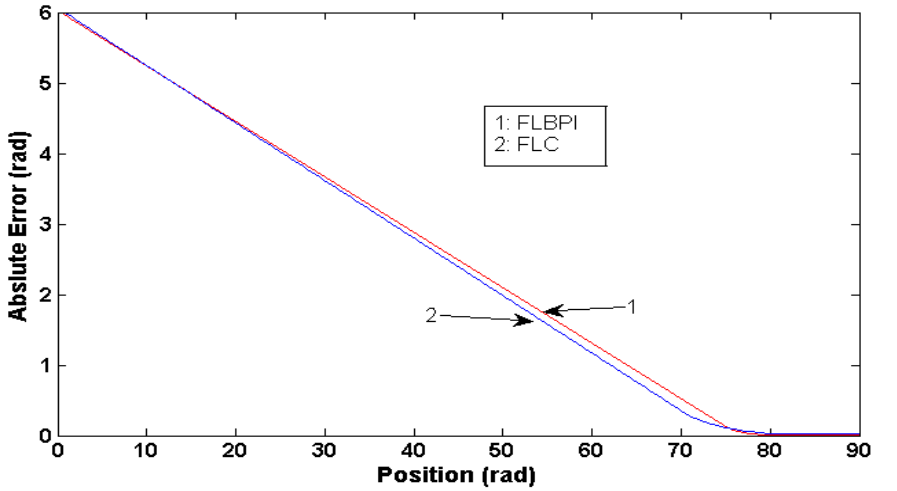


Fig. 4.5 Absolute error of position control of DC servomotor using FLBPI vs. FLC for simulation 1a

The best position controller for simulation 1b is PID for conventional controller and FLBPID for hybrid-fuzzy controller. As a speed and position controller, PID is the best conventional controller and FLBPID is the best hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that hybrid-fuzzy controllers are better then conventional controllers in the simulation 1b. The graphical comparisons are shown in the Fig. 4.6 and Fig. 4.7.

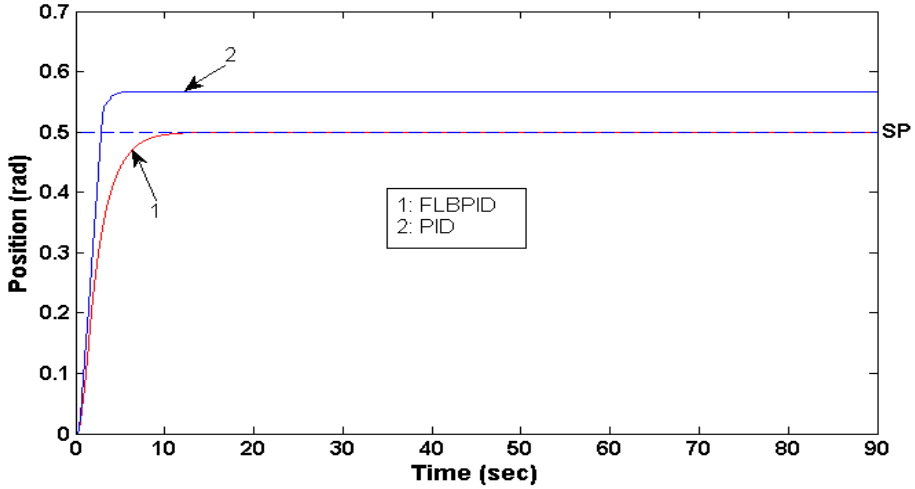


Fig. 4.6 Position control of DC servomotor using FLBPID vs. PID for simulation 1b

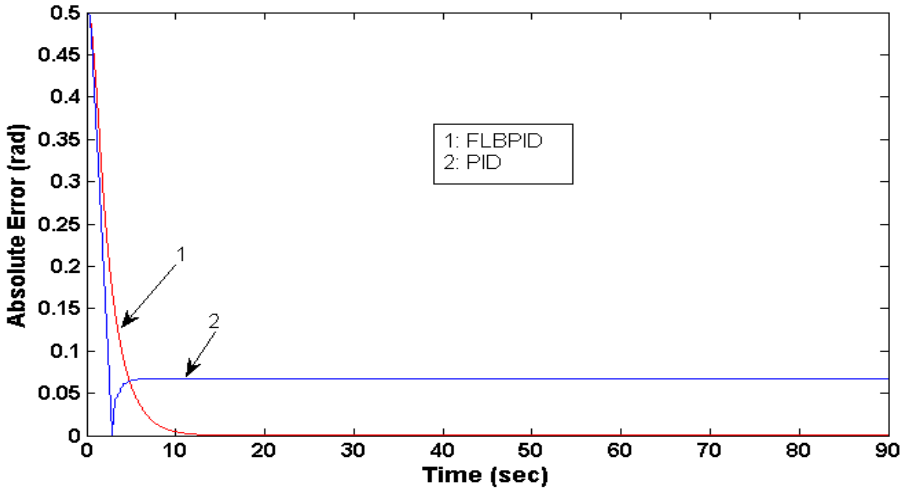


Fig. 4.7 Absolute error of position control of DC servomotor using FLBPID vs. PID for simulation 1b

It is shown in Fig. 4.6 and Fig. 4.7 that FLBPID can reach the setpoint while PID has steady state error. This makes the absolute error of PID larger than FLBPID.

The best speed controller in the simulation 2 is PI for conventional controller and FLIC for hybrid-fuzzy controller, and the best position controller is PID for conventional controller and FLIC for hybrid-fuzzy controller. As a speed and position controller, PI is the best conventional controller and FLIC is the best hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that hybrid-fuzzy controllers are better then conventional controllers in the simulation 2. The graphical comparisons are shown in the Fig. 4.8 to Fig. 4.11.

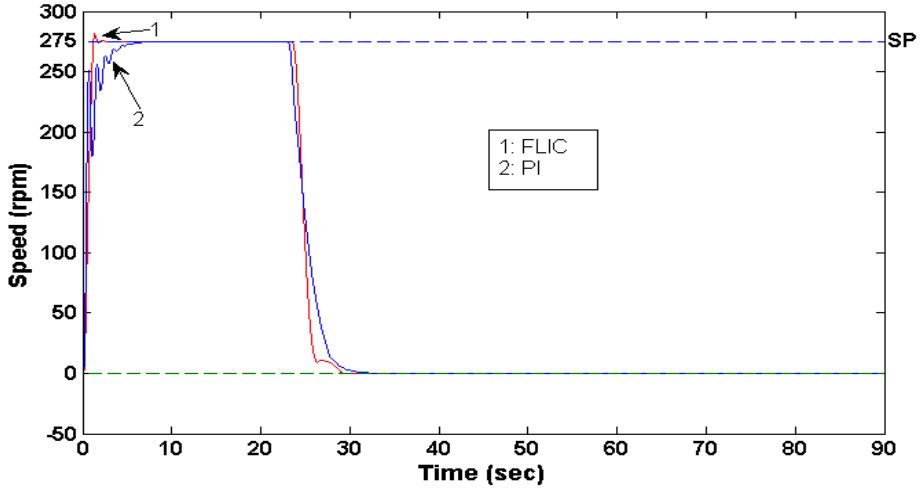


Fig. 4.8 Speed control of DC servomotor using FLIC vs. PI for simulation 2

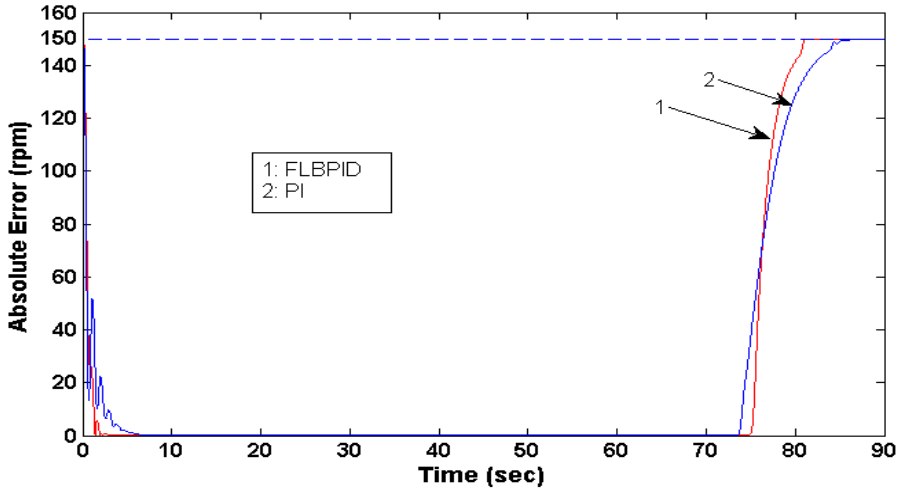


Fig. 4.9 Absolute error of speed control of DC servomotor using FLIC vs. PI for simulation 2

It is shown in Fig. 4.8 and Fig. 4.9 that FLIC has the faster settling time than PI eventhough there is a very small overshoot. Consequently, the absolute error of FLIC is smaller than PI.

It is shown in Fig. 4.10 and Fig. 4.11 that FLIC has the faster settling time than PI. This makes the absolute error of FLIC smaller than PI.

The best speed controller in the simulation 3a is PI for conventional controller and FLBPID for hybrid-fuzzy controller, and the best position controller is PID for conventional controller and FLBPID for hybrid-fuzzy controller. As a speed and position controller, PID is the best conventional controller and FLBPID is the best

hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that hybrid-fuzzy controllers are better then conventional controllers in the simulation 3a. The graphical comparisons are shown in the Fig. 4.12 to Fig. 4.15.

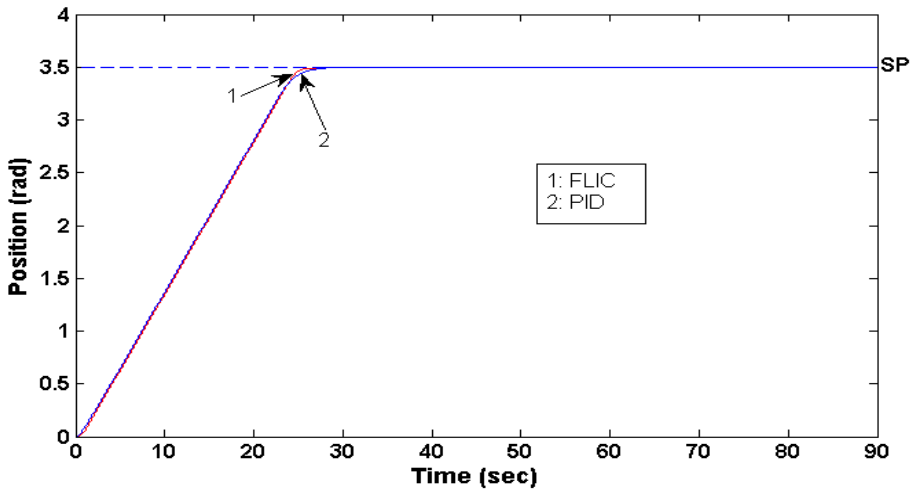


Fig. 4.10 Position control of DC servomotor using FLIC vs. PID for simulation 2

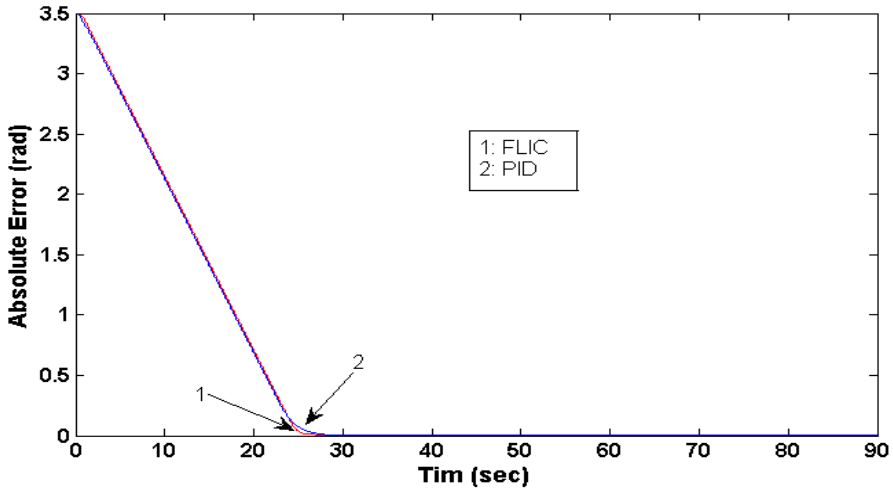


Fig. 4.11 Absolute error of position control of DC servomotor using FLIC vs. PID for simulation 2

Based on Fig. 4.12 and Fig. 4.13, FLBPID has the faster settling time than PI either unloaded or loaded eventhough there is a very small of overshoot. Consequently, the absolute error of FLBPID is smaller than PI.

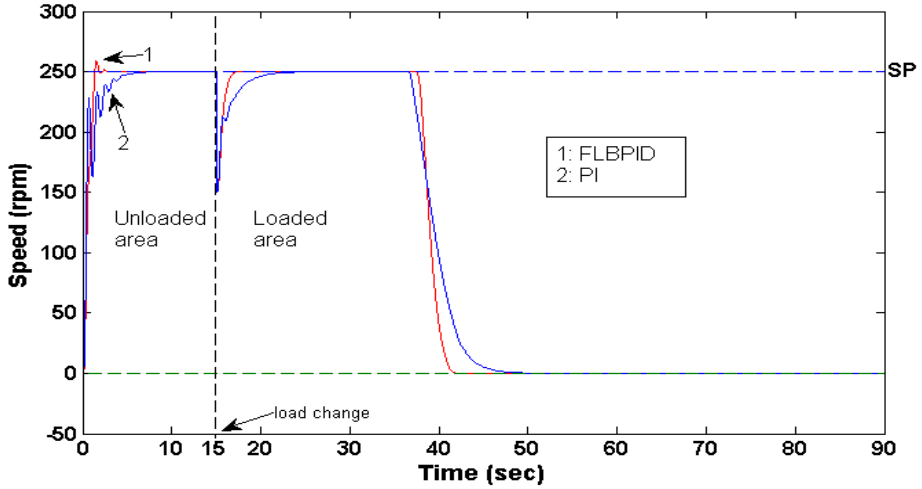


Fig. 4.12 Speed control of DC servomotor using FLBPID vs. PI for simulation 3a

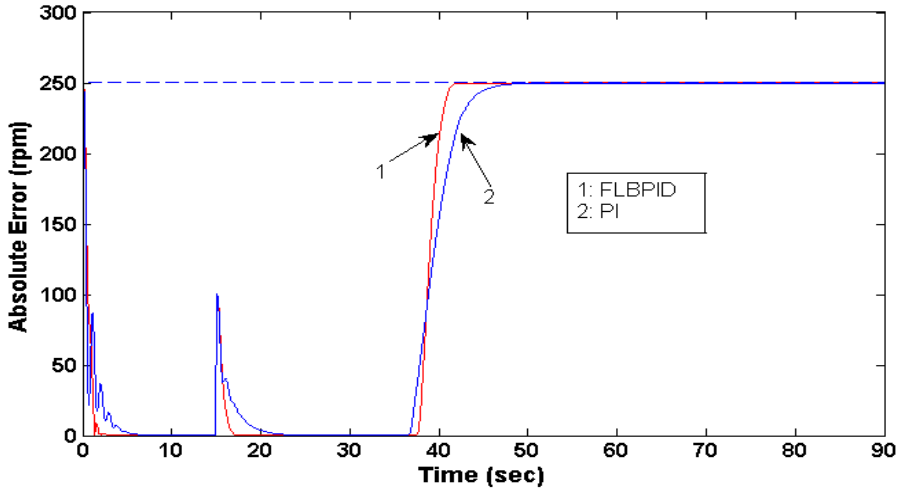


Fig. 4.13 Absolute error of speed control of DC servomotor using FLBPID vs. PI for simulation 3a

Based on Fig. 4.14 and Fig. 4.15, FLBPID has the faster settling time than PID. This makes the absolute error of FLBPID is smaller than PID.

The best speed controller in the simulation 3b is PID for conventional controller and FLBPID for hybrid-fuzzy controller, and the best position controller is PI for conventional controller and FLBPID for hybrid-fuzzy controller. As a speed and position controller, PID is the best conventional controller and FLBPID is the best hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that hybrid-fuzzy controllers are better then conventional

controllers in the simulation 3b. The graphical comparisons are shown in the Fig. 4.16 to Fig. 4.19.

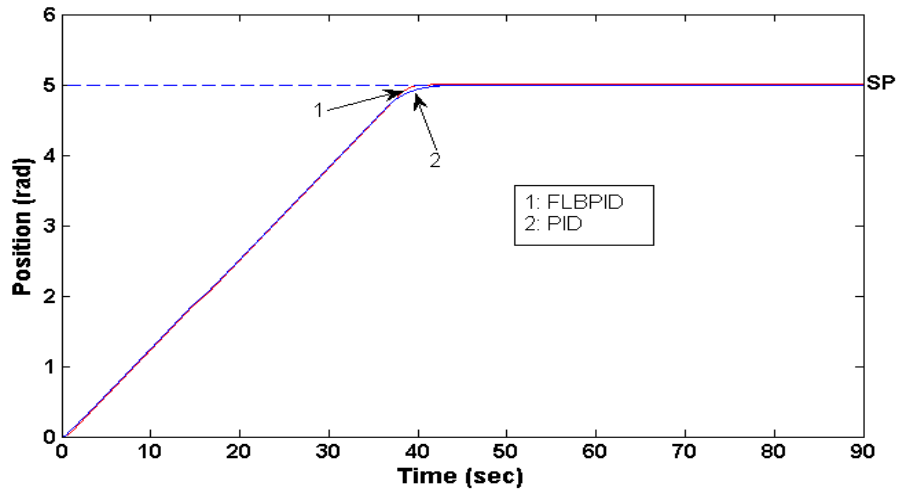


Fig. 4.14 Position control of DC servomotor using FLBPID vs. PID for simulation 3a

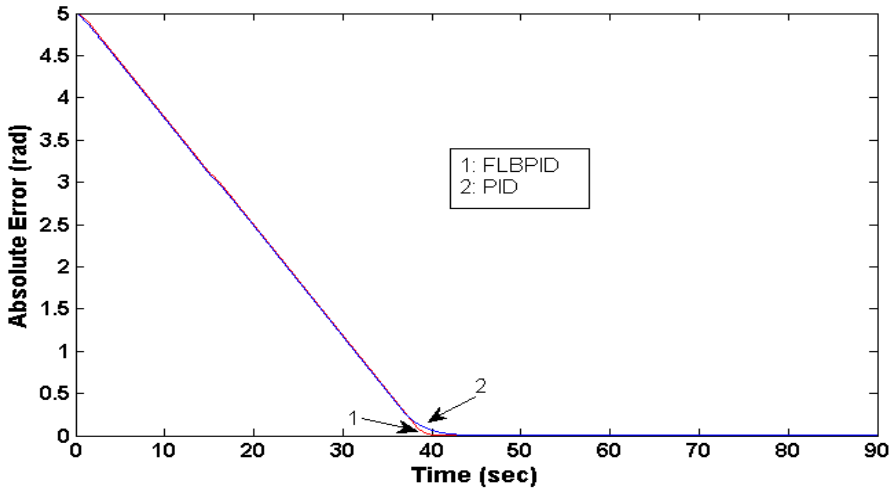


Fig. 4.15 Absolute error of position control of DC servomotor using FLBPID vs. PID for simulation 3a

Based on Fig. 4.16 and Fig. 4.17, FLBPID has the faster settling time than PI eventhough the overshoot 2 of both FBPID and PID are more than 50%.. Consequently, the absolute error of FLBPID is smaller than PI.

Based on Fig. 4.18 and Fig. 4.19, FLBPID has the faster settling time than PID. This makes the absolute error of FLBPID is smaller than PID.

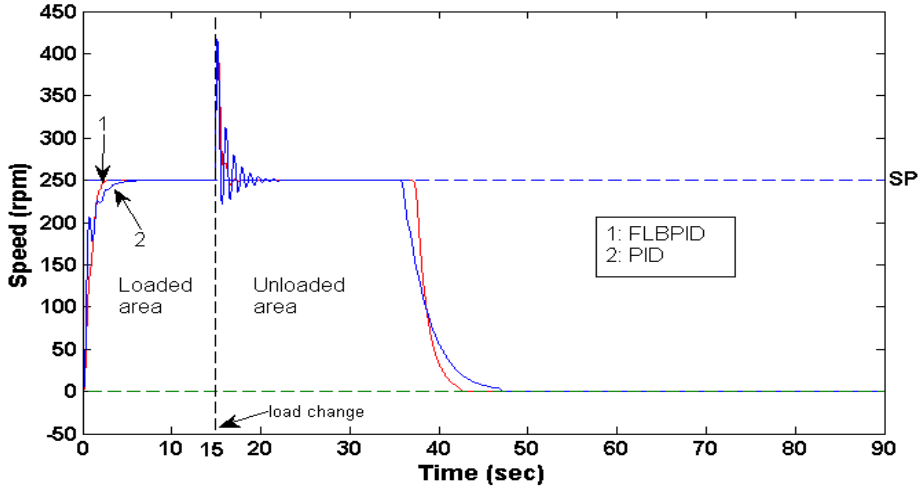


Fig. 4.16 Speed control of DC servomotor using FLBPID vs. PID for simulation 3b

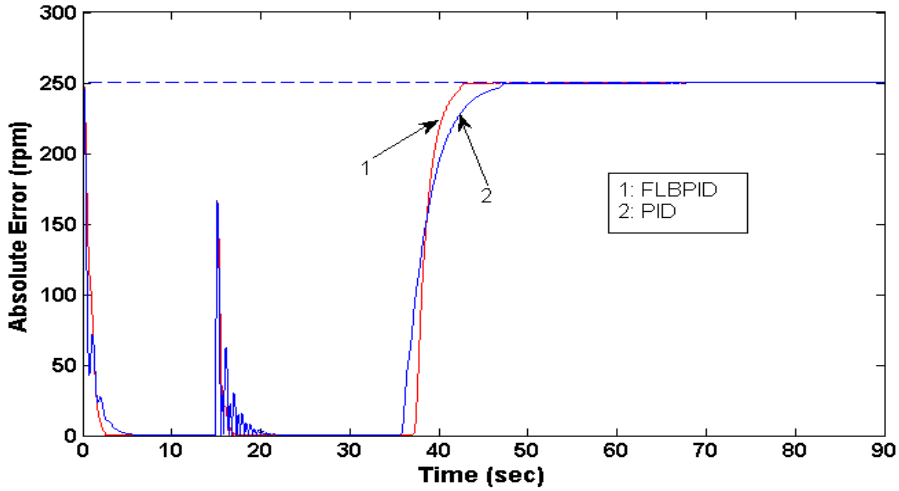


Fig. 4.17 Absolute error of speed control of DC servomotor using FLBPID vs. PID for simulation 3b

The best speed controller in the simulation 4a is PID for conventional controller and FLBPID for hybrid-fuzzy controller. As a speed and position controller, PID is the best conventional controller and FLBPID is the best hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that hybrid controllers are better than conventional controllers in the simulation 4a. The graphical comparisons are shown in the Fig. 4.20 and Fig. 4.21.

Based on Fig. 4.20 and Fig. 4.21, PID has more oscillation than FLBPID. This makes the absolute error of PID is larger than FLBPID.

The best speed controller in the simulation 4b is PI for conventional controller and FLIC for hybrid-fuzzy controller, and the best position controller is PI for conventional controller and FLIC for hybrid-fuzzy controller. As a speed and position controller, PI is the best conventional controller and FLIC is the best hybrid controller. Comparison on the best conventional to the best hybrid are shown in the table that conventional controllers are better than hybrid-fuzzy controllers in the simulation 4b since the position controller of PI is much better then FLIC. The graphical comparisons are shown in the Fig. 4.22 to Fig. 4.25.

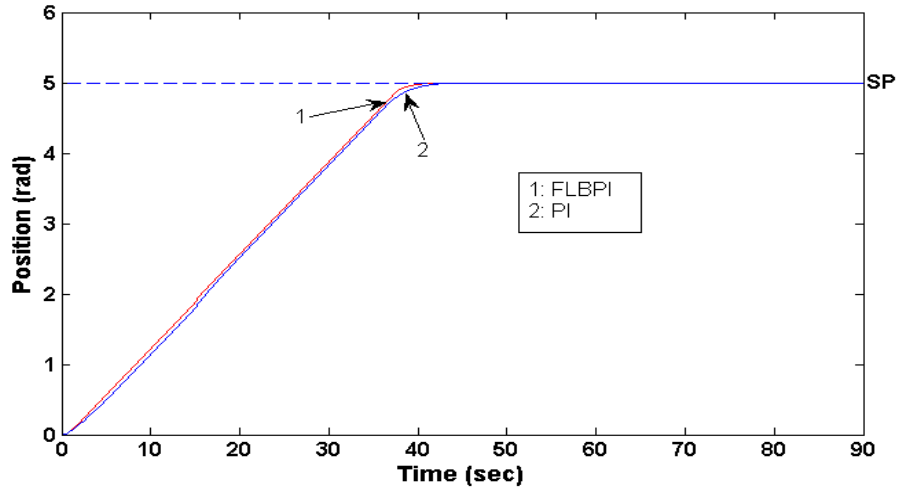


Fig. 4.18 Position control of DC servomotor using FLBPI vs. PI for simulation 3b

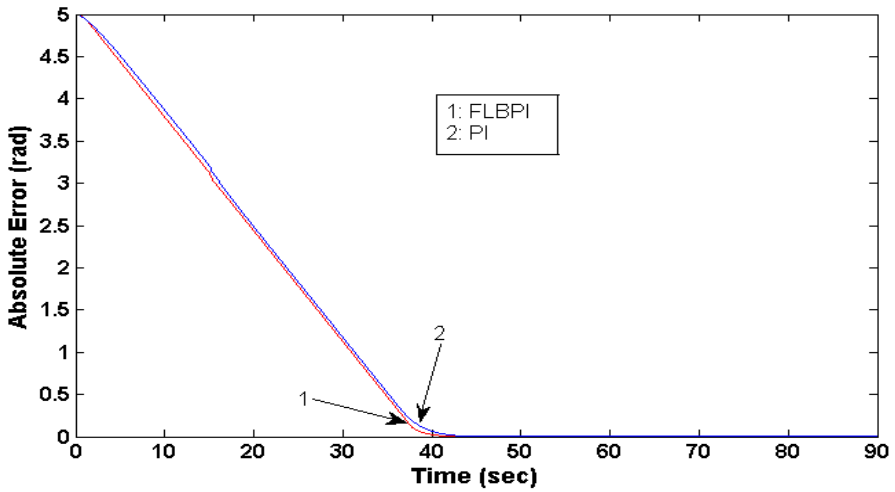


Fig. 4.19 Absolute error of position control of DC servomotor using FLBPI vs. PI for simulation 3b

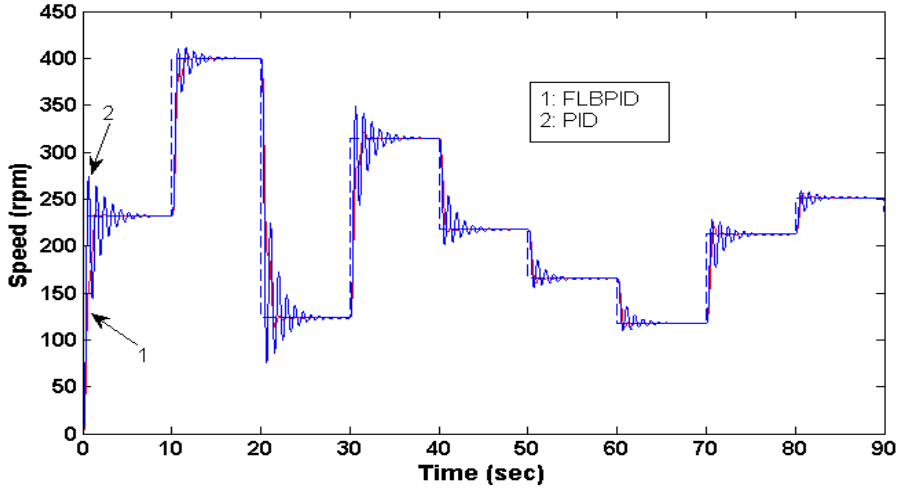


Fig. 4.20 Speed control of DC servomotor using FLBPID vs. PID for simulation 4a

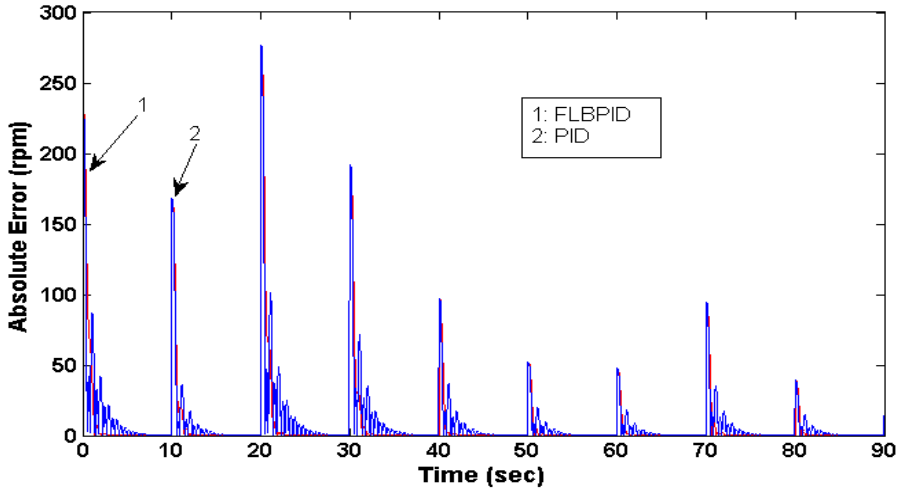


Fig. 4.21 Absolute error of speed control of DC servomotor using FLBPID vs. PID for simulation 4a

Based on in Fig. 4.22 and Fig. 4.23, FLIC has the faster settling time than PI eventhough there is a very small of overshoot. Consequently, the absolute error of FLIC is smaller than PI.

Based on Fig. 4.24 and Fig. 4.25, FLIC has steady state error. This makes the absolute error of FLIC larger than PI.

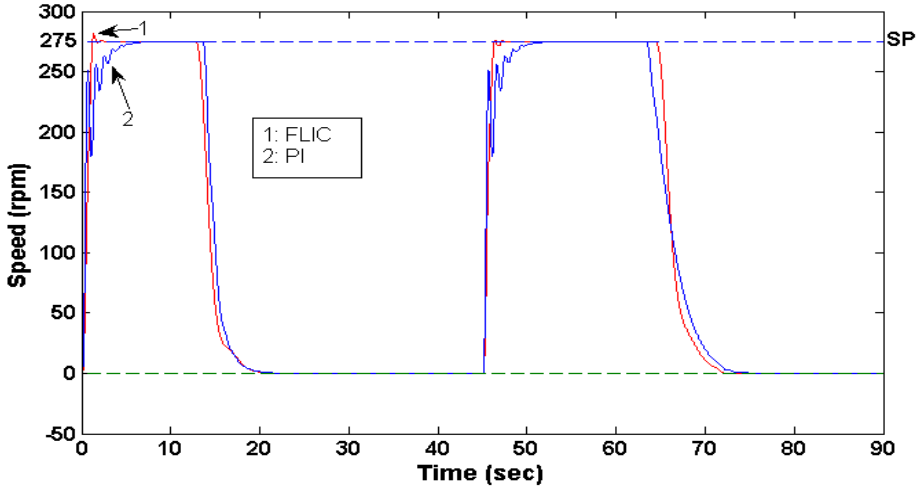


Fig. 4.22 Speed control of DC servomotor using FLIC vs. PI for simulation 4b

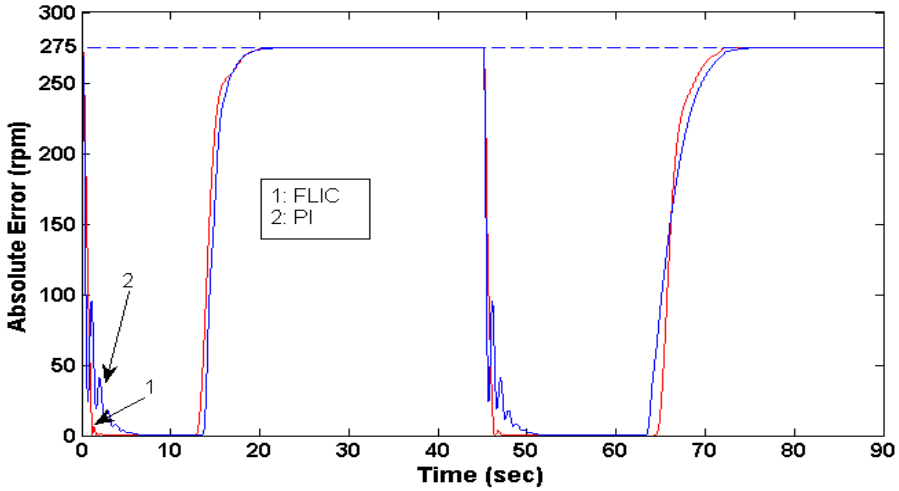


Fig. 4.23 Absolute error of speed control of DC servomotor using FLIC vs. PI for simulation 4b

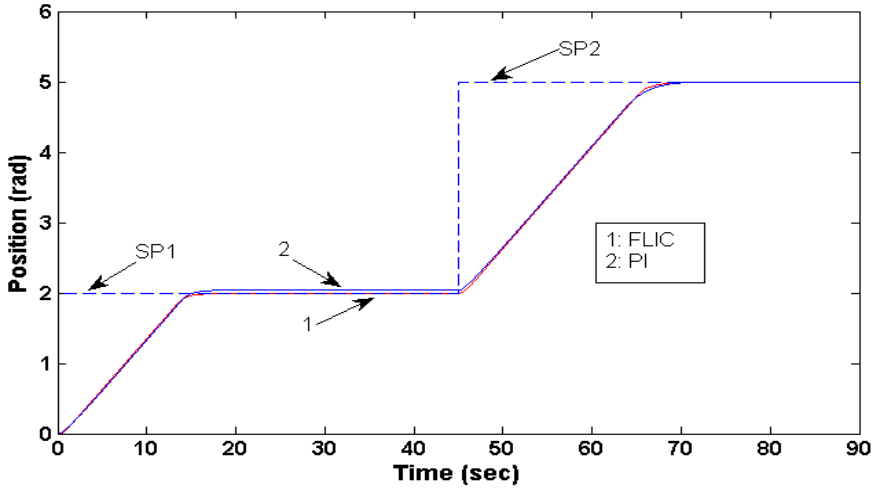


Fig. 4.24 Position control of DC servomotor using FLIC vs. PI for simulation 4b

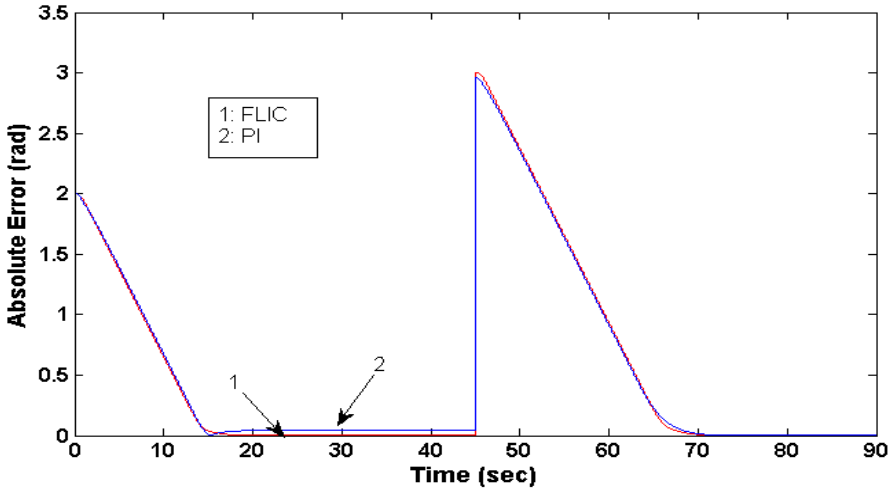


Fig. 4.25 Absolute error of position control of DC servomotor using FLIC vs. PI for simulation 4b

4.5.2 Simulation Results Summary of Conventional, Fuzzy, and Hybrid-Fuzzy Controllers

The speed and position control simulation of hybrid-fuzzy controller compared to conventional and fuzzy controller has been presented. FLBPID controller is the best speed controller as compared to FLBPI controller, FLIC, conventional controllers and fuzzy controller for the simulation of Type 1a, 1b, 3a, 3b, and 4a. FLIC is the best speed controller as compared to FLBPI controller, FLBPID controller, conventional

controllers and fuzzy controller for the simulation of Type 2 and 4b. Using Eq.(3 - 45), the best overall speed controller is FLBPID.

In both speed and position control, FLBPID controller is the best controller as compared to FLBPI controller, FLIC, conventional controllers and fuzzy controller for the simulation of Type 1a, 1b, 3a, 3b, and 4a. FLIC is the best controller as compared to FLBPI controller, FLBPID controller, conventional controllers and fuzzy controller for the simulation of Type 2. PI controller is the best controller as compared to hybrid-fuzzy controllers, PID controller, and fuzzy controller for the simulation of Type 4b. Using Eq.(3 - 45), the best overall speed and position controller is FLBPID.

It is concluded that hybrid-fuzzy controllers have the better performance than conventional and fuzzy controllers. Therefore, the genetic algorithm will be applied to optimize the parameters of hybrid-fuzzy controllers.

4.6 Simulation results of GA and SPOGA

The simulation was done using the method as in Section 3.6.1. There are three results for both GA and SPOGA: the result with minimum specifications, the result with the specifications in which fulfill the minimum criteria, and the result with the specifications in which fulfill the good criteria.

The minimum specifications for both GA and SPOGA are as follow:
Bit length : 40
Population size : 30
Number of generation : 20
Probability of crossover : 0.9
Probability of mutation : 0.01 and 0.1
The result is shown in Table 4.10.

To fulfil the minimum criteria, the number of generation is maintained to be 20 but the population size is continuously increased by 10 until fulfil the requirements. The result for GA is shown in Table 4.11 and for SPOGA is shown in Table 4.12.

Table 4.10 Performance of GA and SPOGA with minimum specification

NO.	ITEMS	p_m = 0.01		p_m = 0.1	
		GA	SPOGA	GA	SPOGA
1	Average error (%)	2.99	2.42	2.62	2.44
2	Accuracy (%)	82.00	96.00	95.00	98.00

The complete results of simulation are shown in Appendix B.

Table 4.11 Results of GA simulation for minimum criteria

NO.	POPULATION SIZE	p_m = 0.01		p_m = 0.1	
		AVERAGE ERROR (%)	ACCURACY (%)	AVERAGE ERROR (%)	ACCURACY (%)
1	30	2.99	82.00	2.62	95.00
2	40	2.39	94.00	2.54	99.00
3	50	2.00	97.00	2.45	99.00
4	60	1.72	98.00	2.16	100.00
5	70	1.49	99.00	2.05	100.00
6	80	1.45	100.00	2.03	100.00
7	90	-	-	1.80	100.00

Table 4.12 Results of SPOGA simulation for minimum criteria

NO.	POPULATION SIZE	p_m = 0.01		p_m = 0.1	
		AVERAGE ERROR (%)	ACCURACY (%)	AVERAGE ERROR (%)	ACCURACY (%)
1	30	2.42	96.00	2.44	98.00
2	40	1.63	100.00	2.19	100.00
3	50	-	-	1.61	100.00

To fulfil good criteria, the number of generation is increased until it fulfils the requirements using the population size in which fulfils the minimum criteria. The result of good criteria for GA is shown in Table 4.13 and for SPOGA is shown in Table 4.14.

It is shown in Table 4.10 that in the minimum specification, SPOGA has the better performance than GA in which the average error is smaller and the accuracy is larger. To fulfil the minimum criteria, GA needs 80 population size for p_m=0.01 and 90 population size for p_m=0.1 in 20 generations as shown in Table 4.11. This means that GA needs (80x21) or 1,680 test runs for p_m=0.01 and (90x21) or 1,890 test runs for p_m=0.1 to get the fitness values for minimum criteria. Meanwhile, SPOGA needs 40 population size for p_m=0.01 and 50 population size for p_m=0.1 in 20 generations as shown in Table 4.12. This means that SPOGA needs (40x21) or 840 test runs for p_m=0.01 and (50x21) or 1,050 test runs for p_m=0.1 to get the fitness values for

minimum criteria. Therefore, SPOGA can make the reduction of (1,680-840) or 840 test runs for p_m=0.01 and (1,890-1,050) or 840 test runs for p_m=0.1 for minimum criteria. In other words, SPOGA can reduce 50 % test runs for p_m=0.01 and 44.44 % test runs for p_m=0.1 for minimum criteria.

Table 4.13 Results of GA simulation for good criteria

NO.	GENERATION	p_m = 0.01 (population=80)		p_m = 0.1 (population=90)	
		AVERAGE ERROR (%)	ACCURACY (%)	AVERAGE ERROR (%)	ACCURACY (%)
1	30	0.92	92.00	1.29	82.00
2	40	0.70	95.00	0.92	94.00
3	50	0.66	93.00	0.92	99.00
4	100	0.52	97.00	0.39	99.00
5	110	0.40	100.00	-	-
6	160	-	-	0.20	100.00

Table 4.14 Results of SPOGA simulation for good criteria

NO.	GENERATION	p_m = 0.01 (population=40)		p_m = 0.1 (population=50)	
		AVERAGE ERROR (%)	ACCURACY (%)	AVERAGE ERROR (%)	ACCURACY (%)
1	30	1.59	76.00	1.29	82.00
2	40	1.43	80.00	0.92	94.00
3	50	1.14	85.00	0.92	99.00
4	100	1.11	94.00	0.39	99.00
5	200	0.62	100.00	-	-
6	210	-	-	0.17	100.00

It is shown in Table 4.13 that to fulfil the good criteria, GA needs 110 generations with 80 populations for p_m=0.01 and 160 generations with 90 populations for p_m=0.1. This means that GA needs (80x111) or 8,880 test runs for p_m=0.01 and (90x161) or 14,490 test runs for p_m=0.1 to get the fitness values for good criteria.

Based on Table 4.14, to fulfil the good criteria, SPOGA needs 200 generations with 40 populations for p_m=0.01 and 210 generations with 50 populations for p_m=0.1. This means that SPOGA needs (40x201) or 8,040 test runs for pm=0.01 and (50x211) or 10,550 test runs to get the fitness values for good criteria. To say it in another way, SPOGA can reduce 9.46 % test runs for p_m=0.01, and 27.19 % test runs for p_m=0.1 for good criteria.

The SPOGA is then applied to optimize the hybrid-fuzzy controllers using minimum criteria. Regarding the simulation with minimum criteria and based on the

fact that GAs work based on natural evolutions or random processes, it is noted that the population size is better to be set at about 4/3 x bit length.

4.7 Process Results of SPOGA in Optimizing Controllers

There are three hybrid controllers to be optimized by SPOGA: FLBPI, FLBPID, and FLIC. The performance of three controllers are to be compared for seven types of experiments.

4.7.1 FLBPI

The SPOGA optimizes the K_{Pm} in the range of [0.00, 23.50] and K_{Im} in the range of [0.00, 130.00] with 12 bit length and 14 bit length respectively using the method as discussed in Section 3.6.1. The probability of crossover and mutation are set to be 0.9 and 0.01 respectively.

In the process of SPOGA, there are three conditions of generation: maximum duplicate chromosomes generation, maximum homogeneous chromosomes generation, and maximum fit chromosome generation. The maximum duplicate chromosomes happened in the second generation as in Table C.1. The maximum homogeneous chromosomes happened in the 16^{th} generation as in Table C.3. The maximum fit chromosome generation happened in the 17^{th} generation as in Table C.7. The maximum fit is the 16^{th} chromosome in the 17^{th} generation as in Table 4.15.

The maximum duplicate chromosomes are not solution chromosomes since there is a possibility a local maxima condition. The generation was still going on with the probability of mutation 0.01 until the maximum generation, i.e. 20. After 20^{th} generation, the maximum fit chromosome was searched and found in the 17^{th} generation as a 16^{th} chromosome as in Table 4.15 in the generation as in Table C.7. Table 4.15 is a solution chromosome where by using Fig. 3.35 gives the parameters of FLBPI as follows:

K_{Pm} = 9.5492, K_{Im} = 73.1295, and experimentally, the K_{Pp} for position controller is 0.97.

Table 4.15 Maximum fit chromosome for FLBPI parameters

0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

4.7.2 FLBPID

The SPOGA optimizes the K_{Pm} in the range of [0.00, 23.50], K_{Dm} in the range of [0.00, 0.20] and K_{Im} in the range of [0.00, 110.00] with 12 bit length, 5 bit length, and 14 bit length respectively using the method as in Section 3.6.2. The probability of crossover and mutation are set to be 0.9 and 0.01 respectively.

In the process of SPOGA, there are three conditions of generation: maximum duplicate chromosomes generation, maximum homogeneous chromosomes generation, and maximum fit chromosome generation. The maximum duplicate chromosomes happened in the second generation as in Table C.2. The maximum homogeneous chromosomes happened in the 19^{th} generation as in Table C.4. The maximum fit chromosome generation happened in the 17^{th} generation as in Table C.8. The maximum fit is the 15^{th} chromosome in the 17^{th} generation as in Table 4.16.

The maximum duplicate chromosomes are not solution chromosomes since there is a possibility of a local maxima condition. The generation was still going on with the probability of mutation 0.01 until getting the maximum generation, i.e. 20. After 20^{th} generation, the maximum fit chromosome was searched and it was found in the 17^{th} generation as a 15^{th} chromosome as in Table 4.16 in the generation as in Table C.8. Table 4.16 is a solution chromosome where by using Fig. 3.35 gives the parameters of FLBPID as follow:

K_{Pm} = 11.7529, K_{Dm} = 0.2, and K_{Im} = 106.5757, and experimentally, the K_{Pp} for position controller is 0.99.

Table 4.16 Maximum fit chromosome for FLBPID parameters

1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1

4.7.3 FLIC

There are two steps in optimizing FLIC and these are outlined as follow:

i. Optimizing the membership functions and rules of FLC part

ii. Optimizing the I/O scales of FLC part and constant of integral controller part

4.7.3.1 Optimizing Membership Function and Rules

The SPOGA optimizes the error membership functions, change of error membership functions and output membership functions which are initially 7 membership functions each with 7 bit length each using the method as in Section 3.6.3.1. The probability of crossover and mutation are set to be 0.9 and 0.01 respectively.

In the process of SPOGA, there are two conditions of generation: maximum homogeneous chromosomes generation and maximum fit chromosome generation. The maximum homogeneous chromosomes occured in the 10^{th} generation as in Table C.5 since the average fit is the largest among the 20-generation. The maximum fit chromosome generation occured in the 11^{th} generation as in Table C.9. The maximum fit is the 26^{th} chromosome in the 11^{th} generation as in Table 4.17.

Table 4.17 Maximum fit chromosome for FLC parameters in FLIC

1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1

The maximum duplicate chromosomes are not solution chromosomes since there is a possibility of a local maxima condition. The generation was still going on with the probability of mutation 0.01 until the maximum generation, i.e. 20. The maximum fit chromosome was searched and it was found in the 11^{th} generation as a 26^{th} chromosome as in Table 4.17 in the generation as in Table C.9. Table 4.17 is a solution chromosome only for FLC part of FLIC in which the results are as in Fig. 4.26, Fig. 4.27 and Table 4.18. This is not a final solution since the process is still

continued for optimization of FLC I/O scales and integral constant of integral controller.

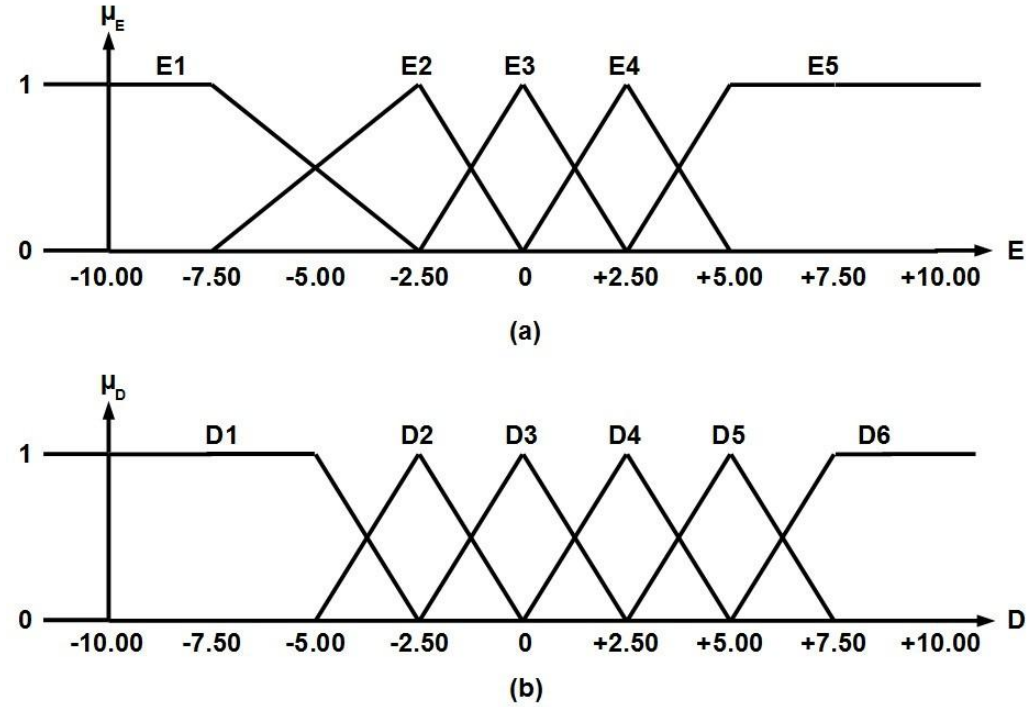


Fig. 4.26 Input membership functions of SPOGA-optimized FLIC: (a) Error membership function, (b) Change of error membership functions

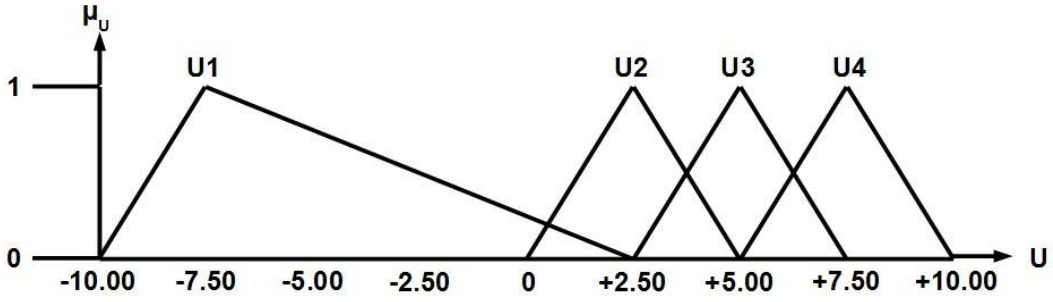


Fig. 4.27 Output membership functions SPOGA-optimized FLIC

Table 4.18 Rules of SPOGA-optimized FLIC

E, Error	D, Change of Error					
	D1	D2	D3	D4	D5	D6
E1	U1	U2	U2	U2	U2	U2
E2	U2	U2	U2	U2	U2	U2
E3	U3	U3	U3	U4	U4	U4
E4	U3	U4	U4	U4	U4	U4
E5	U3	U4	U4	U4	U4	U4

4.7.3.2 Optimizing I/O Scales and Integral Constant

The SPOGA optimizes the K_e in the range of [0.00, 1.50], K_u in the range of [0.00, 0.50], and K_I in the range of [0.00, 1.00] with 8 bit length, 6 bit length, and 7 bit length respectively using the method as described in Section 3.6.3.2. The probability of crossover and mutation are set to be 0.90 and 0.01 respectively.

In the process of SPOGA, there are two conditions of generation: maximum average fit generation and maximum fit chromosome generation. The maximum homogeneous chromosomes occured in the 3rd generation as in Table C.6 since the average fit is the largest among the 20-generation. The maximum fit chromosome generation occured in the 10th generation as in Table C.10. The maximum fit is the 26th chromosome in the 10th generation as in Table 4.19.

The maximum homogenous chromosomes are not solution chromosomes since there is a possibility of a local maxima condition. The generation keeps going with the probability of mutation 0.01 until the maximum generation, i.e. 20. The maximum fit chromosome was searched and it was found in the 10th generation as a 26th chromosome as in Table 4.19 in the generation as in Table C.10. Table 4.19 is a solution chromosome only for I/O scales of FLC and integral constant of integral controller of FLIC after the membership functions and rules are optimized.

Using Fig. 3.35 the I/O scales of FLC and integral constant of integral controllers are as follow:

K_e = 1.50, K_u = 0.00; K_I = 0.54, and experimentally, the K_{Pp} for position controller is 0.69.

This means that the result is an integral controller only. As shown in Table C.5 and Table C.6, comparing with Table C.1 and Table C.2, the homogenous chromosomes in FLIC are not as good as in FLBPI and FLBPID. It is predicted that the result of SPOGA-optimized FLIC will not be as good as the result of SPOGA-optimized FLBPI and FLBPID.

Table 4.19 Maximum fit chromosome for I/O scales and integral constant in FLIC

1	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0

4.8 Simulation of SPOGA Optimized Controllers

There are three types of controllers in the simulation: SPOGA-FLBPI controller, SPOGA-FLPID controller, and SPOGA-FLIC. This simulation was done with the controllers having their parameters optimized using SPOGA.

There are seven types of simulation as explained in Section 3.7 which are summarized in Table 4.2.

4.8.1 Results and Discussions on SPOGA Optimized Controllers

The comparison on the effectiveness of implementing SPOGA optimized hybrid-fuzzy controllers for simulation experiment based on the second order underdamped response analysis is presented in Table 4.20, the comparison based on error analysis is presented in Table 4.21, and the comparison based on fitness value analysis is presented in Table 4.22.

It is shown in Table 4.20, Table 4.21, and Table 4.22 that the best overshoot, settling time and fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is SPOGA-FLBPI controller for speed control in the simulation 1a. The total

fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the SPOGA-FLBPI controller.

Table 4.20 Simulation results of SPOGA optimized hybrid-fuzzy controllers based on second order underdamped response analysis

Type	PERFORMANCE ITEM	Controllers		
		SPOGA-FLBPI	SPOGA-FLBPID	SPOGA-FLIC
1a	Overshoot ($\%O_S$, %)	1.88	3.41	27.31
	Settling time (t_s, sec)	0.80	1.80	4.31
	SSEP ($\%S_p$, %)	0.08	0.08	0.12
1b	SSEP ($\%S_p$)	0.08	0.08	0.04
2	Overshoot ($\%O_S$, %)	6.07	8.26	6.03
	Settling time (t_s, sec)	1.88	2.35	3.31
	SSEP ($\%S_p$, %)	0.06	0.06	0.00
3a	Overshoot ($\%O_S$, %)	5.64	8.16	8.32
	Settling time (t_s, sec)	1.87	2.34	3.14
	Undershoot ($\%U_S$, %)	40.00	40.00	40.00
	SSEP ($\%S_p$, %)	0.05	0.07	0.34
3b	Overshoot ($\%O_S$, %)	0.67	1.30	0.28
	Settling time (t_s, sec)	1.70	1.39	3.18
	Overshoot 2 ($\%O_{S2}$, %)	66.67	66.67	66.67
	SSEP ($\%S_p$, %)	0.07	0.07	0.06
4b	Overshoot ($\%O_S$, %)	6.07	8.26	6.03
	Settling time (t_s, sec)	1.88	2.35	3.31
	SSEP ($\%S_p$, %)	0.07	0.07	0.06

SSEP: Steady State Error of Position

Table 4.21 Simulation results of SPOGA optimized hybrid-fuzzy controllers based on error analysis

Type	PERFORMANCE ITEM	Controller		
		SPOGA-FLBPI	SPOGA-FLBPID	SPOGA-FLIC
1a	$ITAE_{vp}$	2.20E+01	2.25E+01	1.36E+02
	$ITAE_p$	5.94E+03	5.93E+03	5.99E+03
1b	$ITAE_p$	4.45E+00	4.46E+00	4.82E+00
2	$ITAE_{vp}$	7.76E+01	7.01E+01	2.30E+02
	$ITAE_p$	3.78E+02	3.74E+02	3.90E+02
3a	$ITAE_{vp}$	6.60E+01	6.29E+01	2.00E+02
	$ITAE_{vl}$	9.77E+02	8.55E+02	1.94E+03
	$ITAE_p$	1.30E+03	1.29E+03	1.40E+03
3b	$ITAE_{vp}$	9.61E+01	7.49E+01	3.32E+02
	$ITAE_{vl}$	1.25E+03	1.20E+03	2.46E+03
	$ITAE_p$	1.28E+03	1.27E+03	1.33E+03
4a	IAE_v	6.59E+02	6.14E+02	1.30E+03
4b	$ITAE_{vp}$	7.76E+01	7.01E+01	2.30E+02
	$ITAE_p$	1.80E+03	1.78E+03	1.89E+03

$ITAE_{vp}$: Integral of time absolute value of error for the first 8 sec; $ITAE_p$: ITAE for position, $ITAE_{vl}$: ITAE for the 9-sec start loading speed; IAE_v : Integral of absolute value of error for overall 90 sec.

Table 4.22 Simulation results of SPOGA optimized hybrid-fuzzy controllers based on fitness value analysis

Type	PERFORMANCE ITEM	Controller		
		SPOGA-FLBPI	SPOGA-FLBPID	SPOGA-FLIC
1a	f_{vp}	0.50	0.50	0.00
	f_p	0.46	0.54	0.00
	fit_v	0.53	0.47	0.00
	fit_p	0.49	0.51	0.00
	fit_{1a}	0.52	0.48	0.00
1b	f_p	0.51	0.49	0.00
	fit_p	0.28	0.25	0.48
	fit_{1b}	0.28	0.25	0.48
2	f_{vp}	0.49	0.51	0.00
	f_p	0.43	0.57	0.00
	fit_v	0.53	0.30	0.17
	fit_p	0.25	0.29	0.46
	fit_2	0.43	0.30	0.27
3a	f_{vp}	0.49	0.51	0.00
	f_{vpl}	0.47	0.53	0.00
	f_p	0.48	0.52	0.00
	fit_v	0.63	0.37	0.00
	fit_p	0.50	0.50	0.00
	fit_{3a}	0.59	0.41	0.00
3b	f_{vp}	0.48	0.52	0.00
	f_{vpl}	0.00	0.00	0.00
	f_p	0.48	0.52	0.00
	fit_v	0.33	0.27	0.15
	fit_p	0.36	0.26	0.39
	fit_{3b}	0.34	0.26	0.23
4a	f_v	0.48	0.52	0.00
	fit_v	0.48	0.52	0.00
	fit_{4a}	0.48	0.52	0.00
4b	f_{vp}	0.49	0.51	0.00
	f_p	0.45	0.55	0.00
	fit_v	0.53	0.30	0.17
	fit_p	0.32	0.27	0.41
	fit_{4b}	0.46	0.29	0.25

f_{vp} : Fitness value for the first 8-sec starting speed based on $ITAE_{vp}$; f_p : Fitness value for position based on $ITAE_p$; f_{vpl} : Fitness value for the 9-sec start loading speed (from 14 to 23 sec) based on $ITAE_{vpl}$; f_v: Fitness value for overall 90 sec of speed based on IAE_v; fit_v : Total fitness value for speed control; fit_p : Total fitness value for position control; fit_x : Total fitness value for speed and position control in the experiment of Type x

In the position control, the best SSEP is SPOGA-FLBPI controller, but the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPID. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPID controller.

The best total fitness value for speed and position control in the simulation 1a (f_{1a}) is SPOGA-FLBPI controller. Therefore, SPOGA-FLBPI is the best hybrid-fuzzy controller in the simulation 1a (extreme condition).

The simulation 1b presents the position performance since it is in the extreme condition with maximum speed and minimum position. The best SSEP and fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPI. The best total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPI.

The best total fitness value for speed and position control in the simulation 1b (f_{1b}) is SPOGA-FLBPI. Therefore, SPOGA-FLBPI is the best hybrid-fuzzy controller in the simulation 1b (extreme condition).

The best overshoot and settling time for speed control in the simulation 2 is SPOGA-FLBPI but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is SPOGA-FLBPID. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the SPOGA-FLBPI.

In the position control, the best SSEP is SPOGA-FLIC but the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPID.

The best total fitness value for speed and position control in the simulation 2 (f_2) is SPOGA-FLBPI. Therefore, SPOGA-FLBPI is the best hybrid-fuzzy controller in the simulation 2 (moderate condition).

The best overshoot and settling time for speed control in the simulation 3a is SPOGA-FLBPI controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is SPOGA-FLBPID. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, undershoot, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is the SPOGA-FLBPI controller. When start loading, the undershoot is the same for all controllers.

In the position control, the best SSEP is SPOGA-FLBPI controller and the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPID controller.

The best total fitness value for speed and position control in the simulation 3a (f_{3a}) is SPOGA-FLBPI controller. Therefore, SPOGA-FLBPI is the best hybrid-fuzzy controller in the simulation 3a (variable load condition)

The best overshoot for speed control in the simulation 3b is SPOGA-FLBPI but the settling time and the fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is SPOGA-FLBPID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, overshoot 2, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is SPOGA-FLBPI controller. All of the overshoot 2 when start unloading are more than 50%. Consequently, the fitness value for 9-sec start unloading speed based on $ITAE_{vpl}$ (f_{vpl}) are zero.

In the position control, the best SSEP is SPOGA-FLIC but the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLIC.

The best total fitness value for speed and position control in the simulation 3b (f_{3b}) is SPOGA-FLBPI controller. Therefore, SPOGA-FLBPI is the best hybrid-fuzzy controller in the simulation 3b (variable load condition)

The simulation 4a presents the speed performance without overshoot and settling time since it is in the variations of speed set point. The best fitness value based on IAE_v (f_v) is SPOGA-FLBPID controller. The best total fitness value for speed control (fit_v) is obtained based on the IAE_v only, and the best is the SPOGA-FLBPID controller.

The best total fitness value for speed and position control in the simulation 4a (f_{4a}) is SPOGA-FLBPID controller. Therefore, SPOGA-FLBPID is the best hybrid controller in the simulation 4a (variable set point condition).

The best overshoot for speed control in the simulation 4b is SPOGA-FLIC. The best settling time is SPOGA-FLBPI controller, and the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is SPOGA-FLBPID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the SPOGA-FLBPI controller.

In the position control, the best SSEP is SPOGA-FLIC but the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPID. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPI.

The best total fitness value for speed and position control in the simulation 4b (f_{4b}) is SPOGA-FLBPI. Therefore, SPOGA-FLBPI is the best hybrid controller in the simulation 4b (variable set point condition).

4.8.2 Simulation Results Summary of SPOGA-Hybrid-Fuzzy Controllers

The speed and position control simulation of SPOGA optimized hybrid-fuzzy controller has been presented. SPOGA-FLBPI controller is the best speed controller as compared to SPOGA-FLBPID controller and SPOGA-FLIC for simulation of Type 1a, 2, 3a, 3b, and 4b. SPOGA-FLBPID controller is the best speed controller as compared to SPOGA-FLBPI controller and SPOGA-FLIC for simulation of Type 4a. Using Eq. (3 - 45), the best overall speed controller is SPOGA-FLBPI as compared to SPOGA-FLBPID and SPOGA-FLIC.

In both speed and position control, SPOGA-FLBPI controller is the best speed controller as compared to SPOGA-FLBPID controller and SPOGA-FLIC for simulation of Type 1a, 2, 3a, 3b, and 4b. SPOGA-FLBPID controller is the best speed controller as compared to SPOGA-FLBPI controller and SPOGA-FLIC for simulation of Type 4a. SPOGA-FLIC is the best controller as compared to SPOGA-FLBPI controller and SPOGA-FLBPID controller for simulation of Type 1b. Using Eq. (3 - 45), the best overall speed and position controller is SPOGA-FLBPI as compared to SPOGA-FLBPID and SPOGA-FLIC.

4.9 Performance Comparisons of SPOGA to non-SPOGA Controllers

This section presents the performance improvements of non-SPOGA hybrid-fuzzy controllers and SPOGA optimized hybrid-fuzzy controllers based on the fitness values where the performance items are based on Table 4.6 to Table 4.8 and Table 4.20 to Table 4.22.

There are seven types of simulation as explained in Section 3.7 which are summarized in Table 4.2.

The graphs of input-output characteristic of speed error, position error, speed, and position between the best improvement of SPOGA optimized hybrid controllers and the corresponding hybrid controllers are presented in the section for each simulation type.

4.9.1 Comparison of SPOGA Optimized and Non-SPOGA Optimized Controllers

The comparison on the improvement of SPOGA optimized and non-SPOGA hybrid controllers based on the performance metrics for simulation experiment is presented in Table 4.23 where the performance items are based on Table 4.8 and Table 4.22.

It is shown in Table 4.23 that SPOGA-FLBPI makes the best fit improvement in the simulation 1a for speed control. This means that SPOGA-FLBPI makes the best fit improvement in the simulation 1a. The best position controller is SPOGA-FLBPID but it is not optimized by SPOGA. The improvement of SPOGA-FLIC is less than zero, this means that the SPOGA-FLIC cannot make improvement in the simulation 1a. The graphical comparisons of SPOGA-FLBPI to FLBPI are shown in Fig. 4.28 and Fig. 4.29.

It is shown in Fig. 4.28 and Fig. 4.29 that SPOGA-FLBPI has no overshoot and the settling time is faster than FLBPI. This makes the absolute error of SPOGA-FLBPI is smaller than FLBPI.

In the simulation 1b, SPOGA-FLIC has the best fit for position control but the SPOGA is not optimize the position controller. This means that SPOGA-FLIC does not make improvement in the simulation 1b.

Table 4.23 Performance improvement comparison of SPOGA optimized and non-SPOGA hybrid-fuzzy controllers for simulation experiment

Type	PERFORMANCE ITEMS	Controller					
		FLBPI	SPOGA-FLBPI	FLBPID	SPOGA-FLBPID	FLIC	SPOGA-FLIC
1a	fit_v	0.14	0.27	0.22	0.24	0.13	0.00
	fit_p	0.24	0.21	0.17	0.22	0.17	0.00
	fit	0.17	0.25	0.21	0.23	0.15	0.00
	I_{pv1a}		0.13		0.01		-0.13
1b	fit_v	-	-	-	-	-	-
	fit_p	0.15	0.13	0.16	0.11	0.15	0.31
	fit	0.15	0.13	0.16	0.11	0.15	0.31
	I_{pv1b}		-		-		-
2	fit_v	0.10	0.19	0.24	0.13	0.28	0.05
	fit_p	0.18	0.11	0.16	0.12	0.22	0.20
	fit	0.12	0.16	0.22	0.13	0.26	0.10
	I_{pv2}		0.09		-0.11		-0.23
3a	fit_v	0.13	0.22	0.27	0.16	0.22	0.00
	fit_p	0.20	0.20	0.21	0.20	0.20	0.00
	fit	0.15	0.21	0.25	0.17	0.22	0.00
	I_{pv3a}		0.10		-0.11		-0.22
3b	fit_v	0.11	0.12	0.12	0.12	0.10	0.06
	fit_p	0.14	0.13	0.12	0.13	0.11	0.04
	fit	0.12	0.13	0.12	0.12	0.11	0.05
	I_{pv3b}		0.01		-0.00		-0.04
4a	fit_v	0.20	0.22	0.22	0.23	0.13	0.00
	fit_p	-	-	-	-	-	-
	fit	0.20	0.22	0.22	0.23	0.13	0.00
	I_{pv4a}		0.02		0.02		-0.13
4b	fit_v	0.10	0.19	0.24	0.13	0.28	0.05
	fit_p	0.15	0.15	0.16	0.12	0.20	0.22
	fit	0.12	0.18	0.21	0.13	0.26	0.11
	I_{pv4b}		0.09		-0.11		-0.23

SPOGA-FLBPI makes the best fit improvement in the simulation 2. The improvement of SPOGA-FLBPID and SPOGA-FLIC are less than zero, this means that both SPOGA-FLBPID and SPOGA-FLIC cannot make improvement in the simulation 2. The graphical comparisons of SPOGA-FLBPI to FLBPI are shown in Fig. 4.30 and Fig. 4.31.

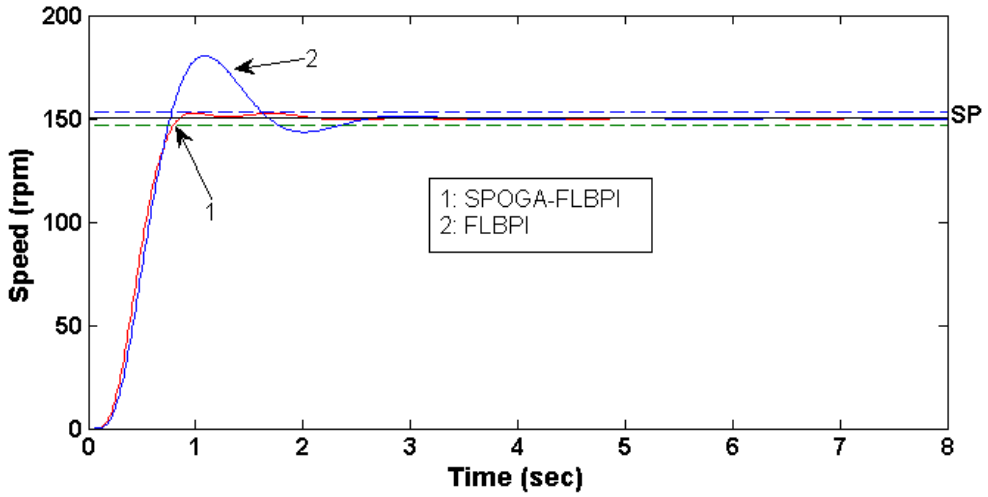


Fig. 4.28 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 1a (see Fig. D.1)

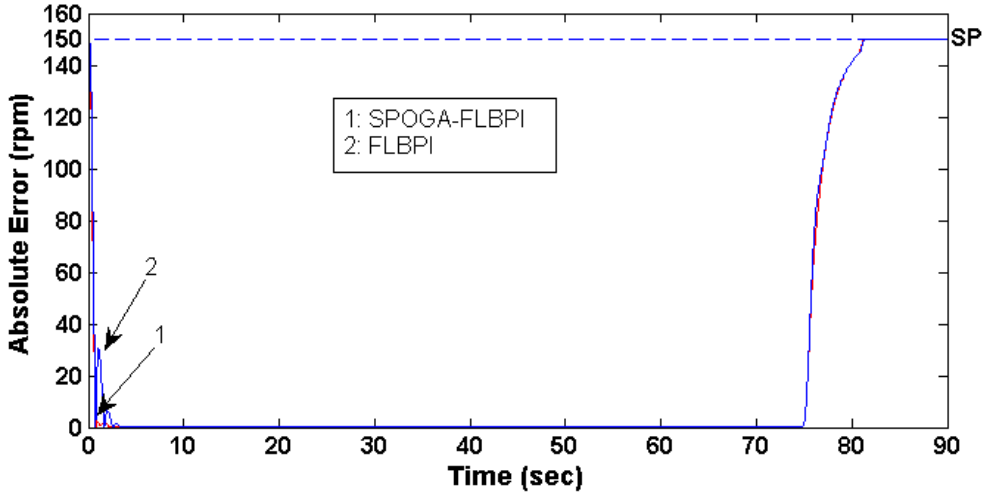


Fig. 4.29 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 1a

It shown in the Fig. 4.30 and Fig. 4.31 that SPOGA-FLBPI has smaller overshoot than FLBPI. This makes the absolute error of SPOGA-FLBPI smaller than FLBPI.

SPOGA-FLBPI makes the best fit improvement in the simulation 3a. The improvement of SPOGA-FLBPID and SPOGA-FLIC are less than zero, this means that both SPOGA-FLBPID and SPOGA-FLIC cannot make improvement in the simulation 3a. The graphical comparisons of SPOGA-FLBPI to FLBPI are shown in Fig. 4.32 and Fig. 4.33.

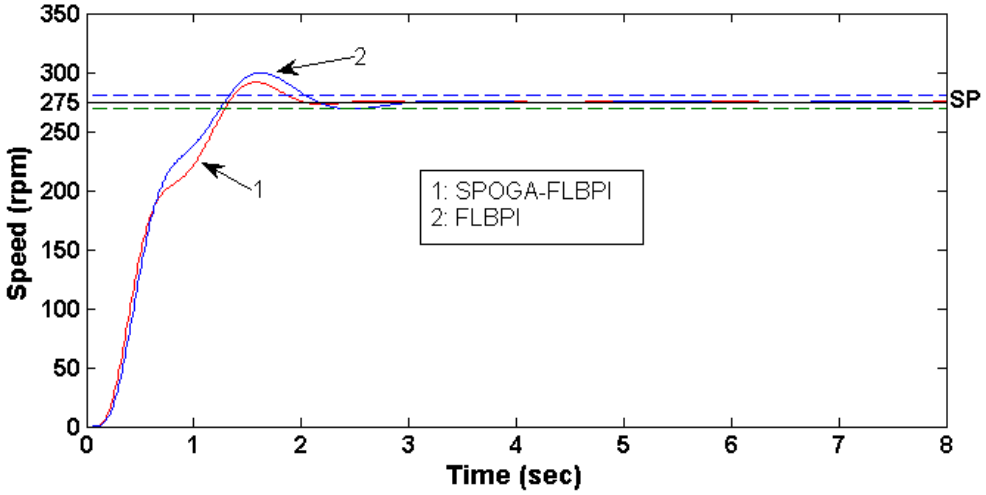


Fig. 4.30 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 2 (see Fig. D.2)

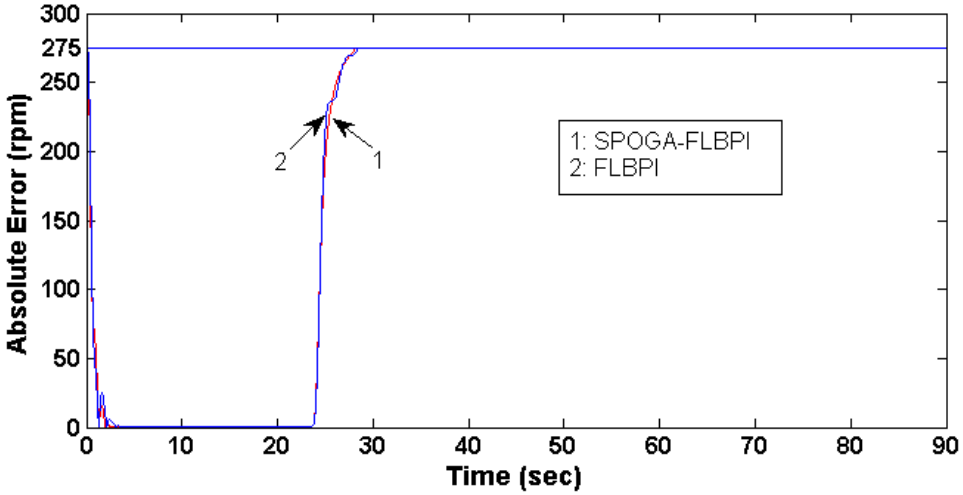


Fig. 4.31 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 2

It shown in the Fig. 4.32 and Fig. 4.33 that SPOGA-FLBPI has smaller overshoot than FLBPI. This makes the absolute error of SPOGA-FLBPI smaller than FLBPI.

SPOGA-FLBPI makes the best fit improv`ement in the simulation 3b. The improvement of SPOGA-FLBPID and SPOGA-FLIC are less than zero, this means that SPOGA-FLBPID and SPOGA-FLIC cannot make improvement in the simulation 3b. The graphical comparisons of SPOGA-FLBPI to FLBPI are shown in Fig. 4.34 and Fig. 4.35.

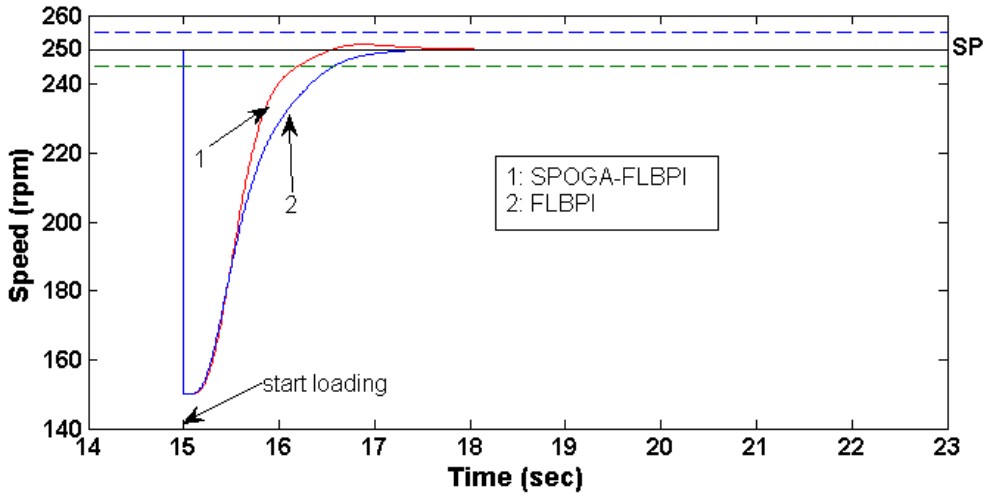


Fig. 4.32 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 3a (see Fig. D.3)

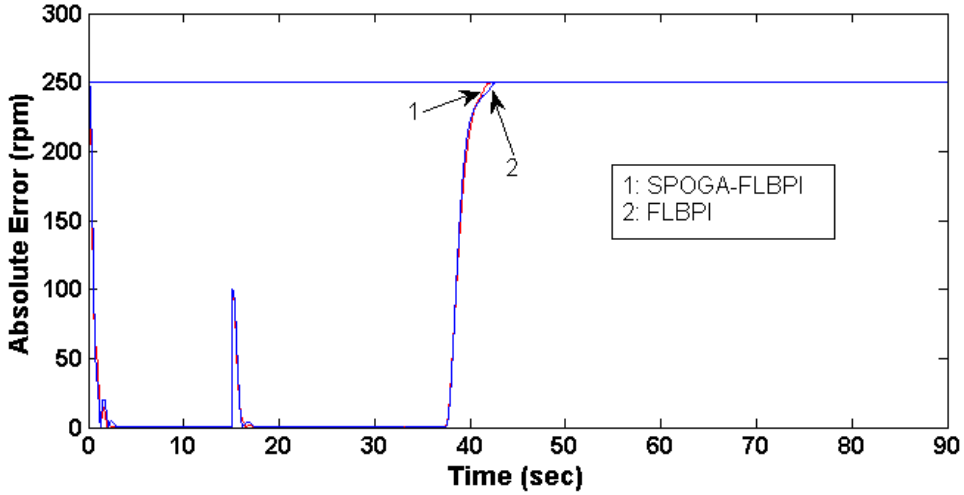


Fig. 4.33 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 3a

It shown in the Fig. 4.34 and Fig. 4.35 that SPOGA-FLBPI has smaller overshoot than FLBPI. On the breaking process SPOGA-FLBPI stops faster than FLBPI. This makes the absolute error of SPOGA-FLBPI smaller than FLBPI eventhough on the start unloading both SPOGA-FLBPI and FLBPI have overshoot more than 50 %.

SPOGA-FLBPI makes the best fit improvement in the simulation 4a. The improvement of SPOGA-FLIC is less than zero, this means that SPOGA-FLIC is

cannot make improvement in the simulation 4a. The graphical comparisons of SPOGA-FLBPI to FLBPI are shown in Fig. 4.36 and Fig. 4.37.

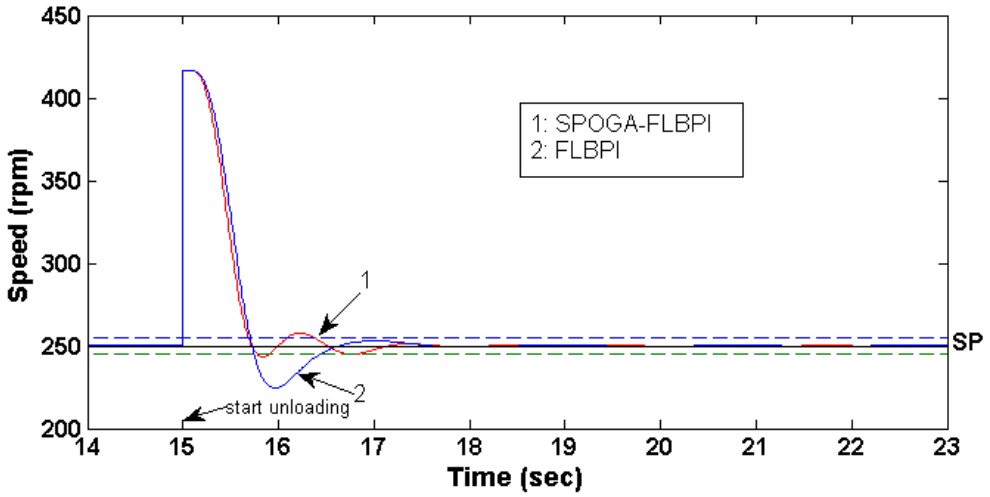


Fig. 4.34 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 3b (see Fig. D.4)

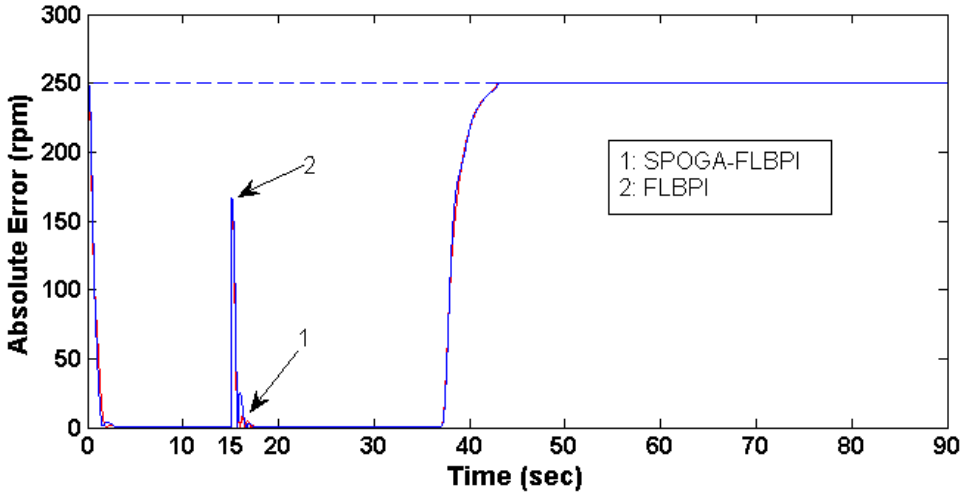


Fig. 4.35 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 3b

It shown in the Fig. 4.36 and Fig. 4.37 that SPOGA-FLBPI has smaller overshoot than FLBPI. This makes the absolute error of SPOGA-FLBPI smaller than FLBPI.

SPOGA-FLBPI makes the best fit improvement in the simulation 4b. The improvement of SPOGA-FLBPID and SPOGA-FLIC are less than zero, this means that both SPOGA-FLBPID and SPOGA-FLIC cannot any improvement in the

simulation 4b. The graphical comparisons of SPOGA-FLBPI to FLBPI are shown in Fig. 4.38 to Fig. 4.40.

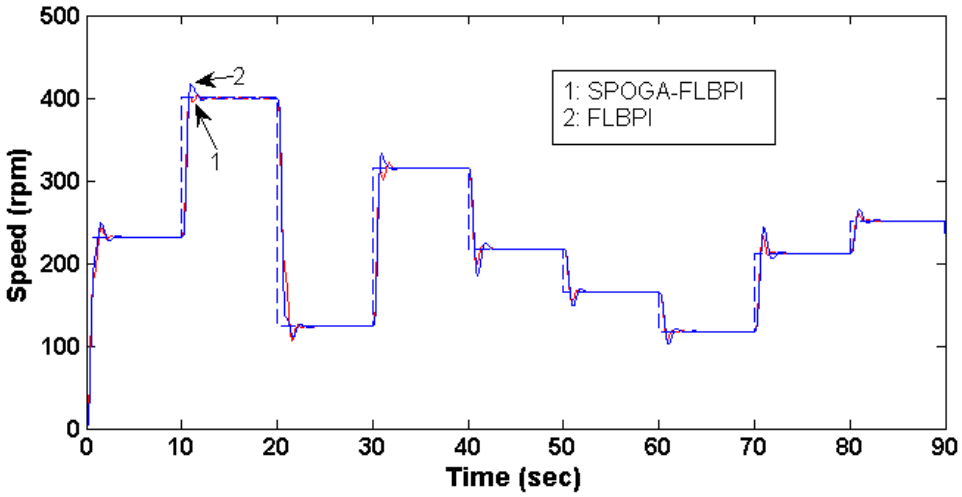


Fig. 4.36 Speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 4a

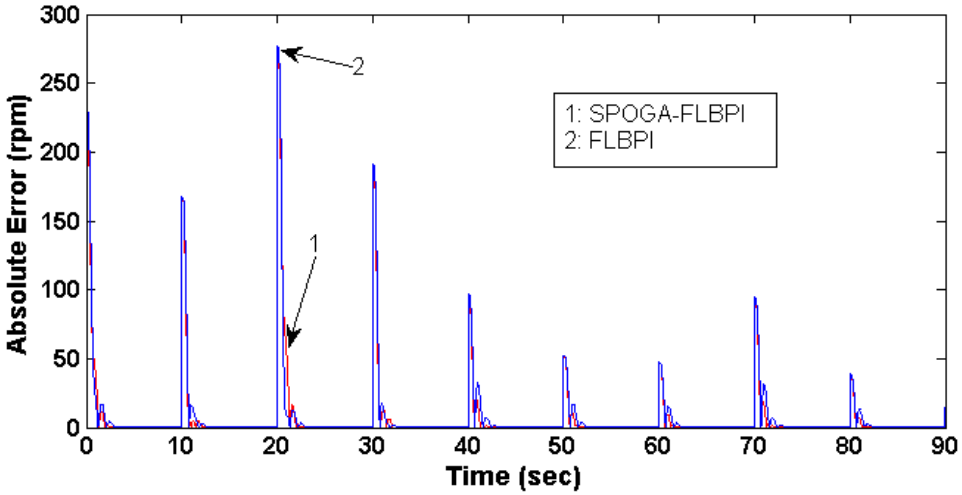


Fig. 4.37 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 4a

It shown in the Fig. 4.38 and Fig. 4.39 that SPOGA-FLBPI has smaller overshoot than FLBPI. This makes the absolute error of SPOGA-FLBPI smaller than FLBPI.

It is shown in Fig. 4.40 that SPOGA-FLBPI is similar to FLBPI. This means that SPOGA-FLBPI does not make any improvement in position control, but the SPOGA is for optimizing the speed controller only.

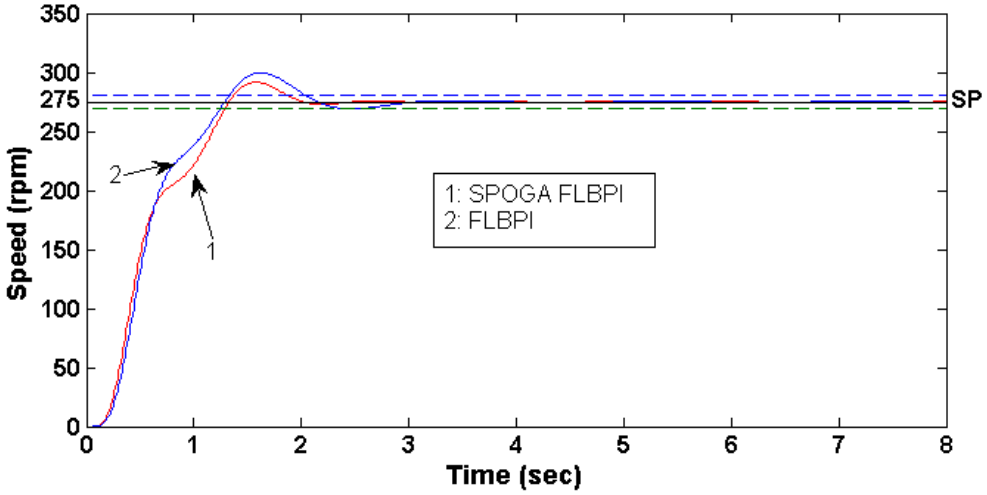


Fig. 4.38 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 4b (see Fig. D.5)

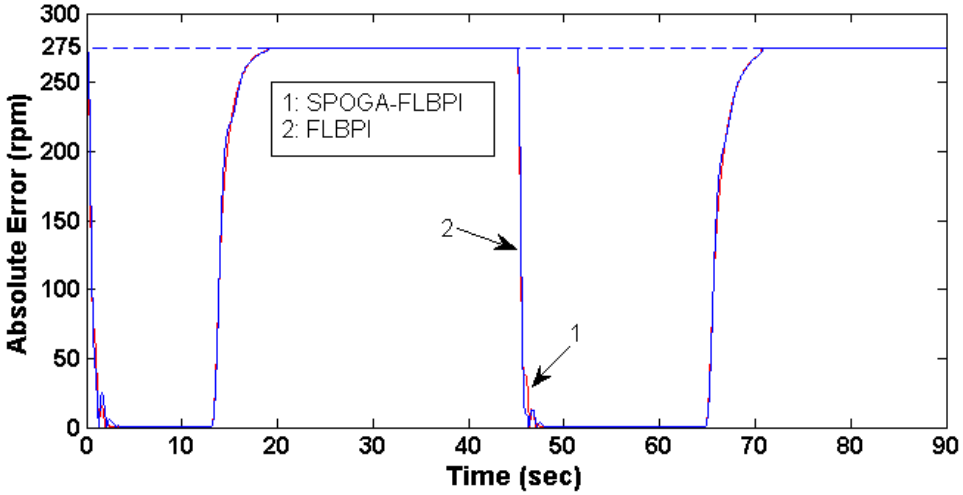


Fig. 4.39 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for simulation 4b

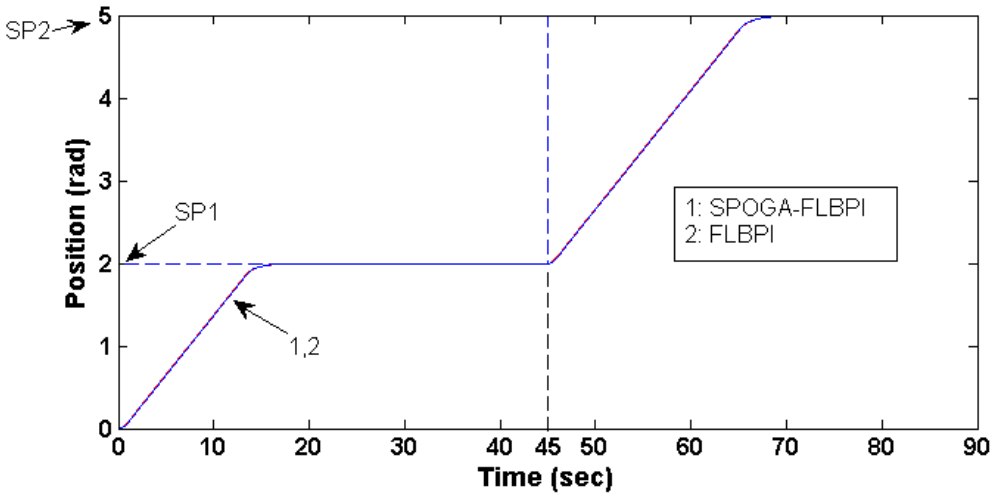


Fig. 4.40 Position control of DC servomotor using SPOGA-FLBPI vs. FLPI for simulation 4b

4.9.2 Simulation Results Summary of SPOGA Optimized and non-SPOGA Optimized Hybrid-Fuzzy Controllers

The improvement tests of SPOGA optimized hybrid-fuzzy to non-SPOGA optimized hybrid-fuzzy controllers have been presented. It was shown that the SPOGA-FLBPI can make fit improvement in all of the simulation types. SPOGA-FLBPID can make improvement in the simulation of Type 1a, and 4a, and cannot make improvement in the simulation of Type 2, 3a, 3b and 4b. SPOGA-FLIC cannot make improvement in all of the simulation types.

The simulation results show that the best total improvement of speed control is SPOGA-FLBPI based on Eq. (3 - 47). This means that the SPOGA can optimize the parameters of FLBPI successfully.

4.10 Summary

This Chapter has presented the simulation of speed and position controllers using conventional controllers (PI and PID controllers), FLC, and hybrid-fuzzy controllers (FLBPI, FLBPID, and FLIC) based on the result of *s*-modeling the DC servomotor.

The simulation results show that hybrid controllers have the better performance than conventional controllers and FLC alone has not been as good performance as conventional and hybrid controllers. Therefore, the GA/SPOGA is applied to optimize the hybrid-fuzzy controllers to make a significant improvement to the conventional controllers.

The simulation results of GA and SPOGA show that to fulfil the minimum criteria, SPOGA can reduce 50% test runs for p_m=0.01 and 44.44 % test runs for p_m=0.1. To fulfil the good criteria, SPOGA can reduce 9.46 % test runs for p_m=0.01 and 27.19 % test runs for p_m=0.1. Therefore, SPOGA is selected to optimize the hybrid-fuzzy controllers using minimum criteria.

Simulation results of SPOGA optimized hybrid-fuzzy controllers show that SPOGA can optimize the parameters of FLBPI successfully, and the comparison to the conventional controllers will be presented in Chapter 5..

CHAPTER 5

REAL-TIME IMPLEMENTATION RESULTS AND DISCUSSIONS

5.1 Introduction

The simulation design of controllers and algorithms have been presented in Chapter 3. The results of *s*-modelling, GA and SPOGA experiments, and simulations of speed and position control have been presented in Chapter 4. The results of real-time implementation on hardware experimental rig will be presented in this chapter.

The simulation results show that hybrid-fuzzy controllers have the better performance than conventional controllers and FLC has not as good performance as conventional and hybrid-fuzzy controllers. The simulation results of SPOGA optimized hybrid-fuzzy controllers show that SPOGA can optimize the parameters of FLBPI successfully, and the comparison to the conventional controllers will be presented in this chapter.

Experiment of sampling period and FIR is conducted to find the appropriate sampling period and the number of points in FIR based on the computer system and the open loop characteristic of the plant. The experiment determines whether the computer based controller can be applied to the hardware or not and determines the appropriate number of points in FIR.

Based on the experiment of sampling period and FIR, the hardware experiment is continued to the experiment of conventional, fuzzy, and hybrid PID-fuzzy controllers using the parameters as in the Chapter 4. The performance evaluations are used as in the Chapter 4 with the 5 % criteria for settling time and steady state error. The experiment results are then compared with the simulation results to test the result of s-modeling of the DC servomotor.

The parameters of SPOGA optimized hybrid-fuzzy controllers in the Chapter 4 are applied to the hardware experiment rig and the results are compared with the simulation results. The performance of SPOGA optimized hybrid-fuzzy controllers are then compared graphically with non-SPOGA optimized hybrid-fuzzy controllers to get the improvement value. Finally, the SPOGA optimized hybrid-fuzzy controllers are compared graphically with the conventional and fuzzy controller to determine how much the SPOGA optimized hybrid-fuzzy controllers can improve the performance of a controller.

5.2 Experiment on Sampling Period and FIR

The hardware design was explained in Section 3.2 of Chapter 3. This section shows the experiment result of determining the sampling period and the appropriate number of points of FIR.

Experiment of sampling period is conducted to get the minimum sampling period that the computer system can handle the hardware and to calibrate between the SIMULINK time and the real time. The experiment result is shown in Table 5. 1.

Table 5. 1 Experiment result of sampling period

SAMPLING PERIOD (sec)	SIMULINK TIME (sec)	REAL TIME (sec)	PLANT TIME CONSTANT (sec)	REMARKS
0.100	100.00	100.89	0.50	Too long
0.010	100.00	101.60		Fulfil the requirement
0.001	100.000	-		Error/Stop responding

Based on Table 5. 1, the sampling period of 0.01 sec can fulfil the requirement regarding to the open loop time constant of DC servomotor which is 0.5 sec, where the maximum sampling period is $\frac{0.5}{10} = 0.05$ sec. Using this sampling period, the difference between real time and SIMULINK time is 1.6 sec, or the real time is delayed 1.6 sec.

Experiment of FIR is conducted to get the appropriate number of points (*N*) of FIR in which the output has the maximum deviation with respect to noise less than 2 % and the time constant of the process value is not too long. Comparison between 25-point FIR and 30-point FIR gives the result as in Table 5.2

Table 5.2 Comparison between 25-point FIR and 30-point FIR

Item	25-point FIR	30-point FIR
Maximum deviation	2.20	1.87
Time constant of process value	0.62	0.66

Based on Table 5.2, the 30-point FIR has the maximum deviation less than 2 %. Increasing the number of point of FIR will make the time constant of process value longer.

5.3 Experiments of Conventional and Fuzzy Logic Controllers

There are three types of controllers in this hardware experiment: (1) PI controller, (2) PID controller, and (3) Fuzzy logic controller. The experiment was done as a comparison with the simulation results for the same controllers.

The parameters of PI and PID controller and the related position controller are the same as in the simulaton experiment of PI and PID controller as described in Section 4.3. The membership functions and rules of FLC for both speed and position controller and the position constant are the same as in the simulation experiment of FLC as described in Section 4.3.

The seven types of hardware experiment similar to as explained in Section 3.7 are going to be conducted. These are classified into four conditions, i.e.:

i. Extreme condition, namely Experiment 1a and 1b
ii. Moderate condition, namely Experiment 2
iii. Variable load condition, namely Experiment 3a and 3b
iv. Variable set point condition, namely Experiment 4a and 4b

The types of experiment are summarized in Table 5.3.

For convenience, Table 4.2 is adopted here and is presented as Table 5.3 that suumarize the type of experiments to be conducted.

Table 5.3 Types of experiment

Type	Experiment Condition	Specifications	
1a	Extreme a	Set point of speed (rpm) Set point of position (rad) Loaded	150.00 6.00 No
1b	Extreme b	Set point of speed (rpm) Set point of position (rad) Loaded	400.00 0.50 No
2	Moderate	Set point of speed (rpm) Set point of position (rad) Loaded	275.00 3.50 No
3a	Variable load a	Set point of speed (rpm) Set point of position (rad) Loaded	250.00 5.00 After 15 sec
3b	Variable load b	Set point of speed (rpm) Set point of position (rad) Loaded	250.00 5.00 Up to 15 sec
4a	Variable set point a	Set point of speed (rpm) Set point of position (rad) Loaded	variable [0.00, 400.00] - No
4b	Variable set point b	Set point of speed (rpm) Set point of position (rad) Loaded	275.00 2 (up to 45 sec), 5 (after 45 sec) No

5.3.1 Results for Conventional and Fuzzy Logic Controllers in Real-time Implementation

The comparison on the effectiveness of implementing conventional and fuzzy logic controllers for hardware experiment based on the second order underdamped response analysis is presented in Table 5.4, the comparison based on error analysis is presented in Table 5.5, and the comparison based on fitness value analysis is presented in Table 5.6.

It is shown in Table 5.4, Table 5.5, and Table 5.6 that the best overshoot and settling time in the experiment 1a for speed control is PI controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the PI controller.

In the position control, the best SSEP is PI controller, but the best fitness value based on $ITAE_p$ (f_p) is PID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PID controller.

Table 5.4 Experiment results of conventional and fuzzy logic controllers based on second order underdamped response analysis

Type	PERFORMANCE ITEM	Controller		
		PI	PID	FLC
1a	Overshoot ($\%O_S$, %)	0.50	23.04	18.84
	Settling time (t_s, sec)	3.53	3.82	9.91
	SSEP ($\%S_p$, %)	0.17	0.20	0.42
1b	SSEP ($\%S_p$)	23.05	13.15	144.59
2	Overshoot ($\%O_S$, %)	1.44	2.20	14.57
	Settling time (t_s, sec)	4.06	2.70	6.02
	SSEP ($\%S_p$, %)	0.09	0.08	0.90
3a	Overshoot ($\%O_S$, %)	0.84	4.42	12.85
	Settling time (t_s, sec)	4.03	6.96	7.70
	Undershoot ($\%U_S$, %)	35.04	33.42	41.83
	SSEP ($\%S_p$, %)	0.46	0.03	0.72
3b	Overshoot ($\%O_S$, %)	0.00	1.74	4.12
	Settling time (t_s, sec)	5.42	3.47	6.14
	Overshoot 2 ($\%O_{S2}$, %)	46.66	51.63	63.59
	SSEP ($\%S_p$, %)	0.00	0.03	0.59
4b	Overshoot ($\%O_S$, %)	0.00	2.49	11.57
	Settling time (t_s, sec)	3.52	2.40	5.33
	SSEP ($\%S_p$, %)	0.07	0.09	0.15

SSEP: Steady State Error of Position

Table 5.5 Experiment results of conventional and fuzzy logic controllers based on error analysis

Type	PERFORMANCE ITEM	Controller		
		PI	PID	FLC
1a	$ITAE_{vp}$	1.22E+02	1.03E+02	2.21E+02
	$ITAE_p$	6.02E+03	5.96E+03	6.05E+03
1b	$ITAE_p$	4.67E+02	2.67E+02	2.91E+03
2	$ITAE_{vp}$	2.27E+02	1.22E+02	4.55E+02
	$ITAE_p$	3.92E+02	3.79E+02	5.31E+02
3a	$ITAE_{vp}$	2.38E+02	1.45E+02	3.69E+02
	$ITAE_{vl}$	2.04E+03	1.60E+03	2.42E+03
	$ITAE_p$	1.41E+03	1.34E+03	1.55E+03
3b	$ITAE_{vp}$	1.51E+03	1.63E+03	2.18E+03
	$ITAE_{vl}$	2.17E+03	1.55E+03	4.24E+03
	$ITAE_p$	1.32E+03	1.30E+03	1.47E+03
4a	IAE_v	1.14E+03	8.86E+02	2.30E+03
4b	$ITAE_{vp}$	2.52E+02	1.27E+02	4.42E+02
	$ITAE_p$	1.84E+03	1.81E+03	1.93E+03

$ITAE_{vp}$: Integral of time absolute value of error for the first 8 sec; $ITAE_p$: ITAE for position, $ITAE_{vl}$: ITAE for the 9-sec start loading speed; IAE_v : integral of absolute value of error for overall 90 sec.

The best total fitness value for speed and position control in the experiment 1a (f_{1a}) is PI controller. Therefore, PI is the best conventional controller in the experiment 1a which is better than FLC.

Comparing with the simulation results, the performance is not exactly the same but the best controller with the best total fitness value for speed and position control is the same.

Table 5.6 Experiment results of conventional and fuzzy logic controllers based on fitness value analysis

Type	PERFORMANCE ITEM	Controller		
		PI	PID	FLC
1a	f_{vp}	0.46	0.54	0.00
	f_p	0.27	0.73	0.00
	fit_v	0.60	0.34	0.05
	fit_p	0.40	0.60	0.00
	fit_{1a}	0.54	0.43	0.03
1b	f_p	0.48	0.52	0.00
	fit_p	0.48	0.52	0.00
	fit_{1b}	0.48	0.52	0.00
2	f_{vp}	0.41	0.59	0.00
	f_p	0.48	0.52	0.00
	fit_v	0.43	0.57	0.00
	fit_p	0.49	0.51	0.00
	fit_2	0.45	0.55	0.00
3a	f_{vp}	0.37	0.63	0.00
	f_{vpl}	0.32	0.68	0.00
	f_p	0.39	0.61	0.00
	fit_v	0.37	0.63	0.00
	fit_p	0.32	0.68	0.00
	fit_{3a}	0.39	0.61	0.00
3b	f_{vp}	0.31	0.69	0.00
	f_{vpl}	0.43	0.00	0.00
	f_p	0.46	0.54	0.00
	fit_v	0.40	0.46	0.00
	fit_p	0.49	0.51	0.00
	fit_{3b}	0.43	0.48	0.00
4a	f_v	0.45	0.55	0.00
	fit_v	0.45	0.55	0.00
	fit_{4a}	0.45	0.55	0.00
4b	f_{vp}	0.38	0.62	0.00
	f_p	0.43	0.57	0.00
	fit_v	0.44	0.56	0.00
	fit_p	0.51	0.49	0.00
	fit_{4b}	0.46	0.54	0.00

f_{vp} : Fitness value for the first 8-sec starting speed based on $ITAE_{vp}$; f_p : Fitness value for position based on $ITAE_p$; f_{vpl} : Fitness value for the 9-sec start loading speed (from 14 to 23 sec) based on $ITAE_{vpl}$; f_v: Fitness value for overall 90 sec of speed based on IAE_v; fit_v : Total fitness value for speed control; fit_p : Total fitness value for position control; fit_x : Total fitness value for speed and position control in the experiment of Type x

The experiment 1b presents the position performance since it is in the extreme condition with maximum speed and minimum position. The best SSEP and the best fitness value based on $ITAE_p$ (f_p) is PID controller. The best total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PID controller.

The best total fitness value for speed and position control in the experiment 1b (f_{1b}) is PID controller. Therefore, PID is the best conventional controller in the experiment 1b which is better than FLC, as in the simulation experiment.

The best overshoot in the experiment 2 for speed control is PI controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) and the best settling time is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the PID controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is PID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PID controller.

The best total fitness value for speed and position control in the experiment 2 (f_2) is PID controller. Therefore, PID is the best conventional controller in the experiment 2 which is better than FLC.

Comparing with the simulation results, the performance is not exactly the same but the best controller with the best total fitness value for speed and position control is the same.

The best overshoot and settling time in the experiment 3a for speed control is PI controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, undershoot, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is PI controller. When start loading, the best undershoot is PID controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is PID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PID controller.

The best total fitness value for speed and position control in the experiment 3a (f_{3a}) is PID controller. Therefore, PID is the best conventional controller in the experiment 3a which is better than FLC.

Comparing with the simulation results, the performance is not exactly the same but the best controller with the best total fitness value for speed and position control is the same.

The best overshoot and overshoot 2 (when start unloading) in the experiment 3b is PI controller but the best settling time and the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) for speed control is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, overshoot 2, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is the PID controller.

In the position control, the best SSEP is PI controller but the best fitness value based on $ITAE_p$ (f_p) is PID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PID controller.

The best total fitness value for speed and position control in the experiment 3b (f_{3b}) is PID controller. Therefore, PID is the best conventional controller in the experiment 3b which is better than FLC.

Comparing with the simulation results, the performance is not exactly the same but the best controller with the best total fitness value for speed and position control is the same.

The experiment 4a presents the speed performance without overshoot and settling time since it is in the variations of speed set point. The best fitness value based on IAE_v (f_v) is PID controller. The best total fitness value for speed control (fit_v) is obtained based on the IAE_v only, and the best is the PID controller.

The best total fitness value for speed and position control in the experiment 4a (f_{4a}) is PID controller. Therefore, PID is the best conventional controller in the experiment 4a which is better than FLC, as in the simulation experiment.

The best overshoot in the experiment 4b for speed control is PI controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) and the best settling time is PID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best one is PID controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is PI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the PID controller.

The best total fitness value for speed and position control in the experiment 4b (f_{4b}) is PID controller. Therefore, PID is the best conventional controller in the experiment 4b which is better than FLC.

Comparing with the simulation results, the performance is not exactly the same because the tuning process was done on the transfer function obtained via system identification of s-modeling of the system.

5.3.2 Experiment Results Summary of Conventional and Fuzzy Controllers

The speed and position control hardware of conventional and fuzzy logic controller has been presented. PI controller is the best speed controller as compared to PID controller and FLC for experiment of Type 1a, and 3a. PID controller is the best speed controller as compared to PI controller and FLC for experiment of Type 2, 3b, 4a, and 4b. Using Eq. (3 - 45), the best overall speed controller is PID as compared to PI and fuzzy logic. In the application of speed control, FLC is not as good as conventional controller.

In both speed and position control, PI is the best controller as compared to PID controller and FLC for experiment of Type 1a. PID controller is the best controller as

compared to PI controller and FLC for experiment of Type 1b, 2a,3a, 3b, 4a and 4b. Using Eq. (3 - 45), the best overall speed and position controller is PID as compared to PI and fuzzy logic. In the application of speed and position control, FLC is not as good as conventional controller as in the simulation results, but in the real-time implementation results, the best is PID controller while in the simulation results the best is PI. Noise and filtering in the real-time implementation make the effect of derivative part in the PID more significant.

Comparing with the simulation results, the real-time implementation results tend to be similar to the simulation results.

5.4 Experiment of Hybrid-Fuzzy Controllers

There are three types of controllers in this experiment: FLBPI controller, FLPID controller, and FLIC. The experiment was done as an improvement of PID and FLC based on the results that the performance of standard FLC is not as good as PID or PI controller.

The parameters of FLBPI and FLBPID controller and the related position controller are the same as in the simulaton experiment of FLBPI and FLBPID controller as described in Section 4.4. The membership functions and rules of FLIC for both speed controller and the position constant are the same as in the simulation experiment of FLIC as described in Section 4.4.

There are seven types of hardware experiment as explained in Section 3.7 which are summarized in Table 5.3.

5.4.1 Results of Hybrid-Fuzzy Controller in Real-time Implementation

The comparison on the effectiveness of implementing hybrid-fuzzy controllers for hardware experiment based on the second order underdamped response analysis is presented in Table 5.7, the comparison based on error analysis is presented in

Table 5.8, and the comparison based on fitness value analysis is presented in Table 5.9.

It is shown in Table 5.7, Table 5.8, and Table 5.9 that the best overshoot and settling time in the experiment 1a is FLBPID controller but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is FLIC for speed control. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the FLBPID controller.

In the position control, the best SSEP is FLBPI controller, but the best fitness value based on $ITAE_p$ (f_p) is FLIC. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLBPI controller.

The best total fitness value for speed and position control in the experiment 1a (f_{1a}) is FLBPID controller. Therefore, FLBPID is the best hybrid controller in the experiment 1a (extreme condition).

Table 5.7 Experiment results of hybrid-fuzzy controllers based on second order underdamped response analysis

Type	PERFORMANCE ITEM	Controller		
		FLBPI	FLBPID	FLIC
1a	Overshoot ($\%O_S$, %)	27.45	17.26	26.56
	Settling time (t_s, sec)	2.38	1.96	2.36
	SSEP ($\%S_p$, %)	0.03	0.03	0.06
1b	SSEP ($\%S_p$)	0.79	0.79	1.00
2	Overshoot ($\%O_S$, %)	3.08	2.49	6.31
	Settling time (t_s, sec)	1.37	1.87	5.10
	SSEP ($\%S_p$, %)	0.06	0.08	0.03
3a	Overshoot ($\%O_S$, %)	4.23	3.97	4.60
	Settling time (t_s, sec)	4.45	1.91	6.93
	Undershoot ($\%U_S$, %)	37.95	36.42	37.72
	SSEP ($\%S_p$, %)	0.03	0.09	0.69
3b	Overshoot ($\%O_S$, %)	3.51	3.52	3.25
	Settling time (t_s, sec)	1.29	1.98	3.07
	Overshoot 2 ($\%O_{S2}$, %)	58.24	45.89	61.61
	SSEP ($\%S_p$, %)	0.04	0.02	0.08
4b	Overshoot ($\%O_S$, %)	7.37	1.71	4.07
	Settling time (t_s, sec)	1.61	1.70	2.98
	SSEP ($\%S_p$, %)	0.06	0.08	0.09

SSEP: Steady State Error of Position

Comparing with the simulation results, the performance is not exactly the same but the best controller with the best total fitness value for speed and position control is the same.

The experiment 1b presents the position performance since it is in the extreme condition with maximum speed and minimum position. The best SSEP and the best fitness value based on $ITAE_p$ (f_p) is both FLBPI and FLBPID. The best total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is both FLBPI and FLBPID.

Table 5.8 Experiment results of hybrid-fuzzy controllers based on error analysis

Type	PERFORMANCE ITEM	Controller		
		FLBPI	FLBPID	FLIC
1a	$ITAE_{vp}$	7.36E+01	6.70E+01	5.62E+01
	$ITAE_p$	5.92E+03	5.95E+03	5.92E+03
1b	$ITAE_p$	1.85E+01	1.85E+01	2.37E+01
2	$ITAE_{vp}$	1.19E+02	1.29E+02	1.80E+02
	$ITAE_p$	3.77E+02	3.84E+02	3.78E+02
3a	$ITAE_{vp}$	1.15E+02	1.27E+02	1.49E+02
	$ITAE_{vl}$	1.15E+03	1.93E+03	1.63E+03
	$ITAE_p$	1.29E+03	1.38E+03	1.42E+03
3b	$ITAE_{vp}$	1.14E+02	2.58E+02	2.26E+02
	$ITAE_{vl}$	1.50E+03	1.57E+03	1.91E+03
	$ITAE_p$	1.26E+03	1.34E+03	1.29E+03
4a	IAE_v	8.86E+02	8.91E+02	1.14E+03
4b	$ITAE_{vp}$	1.26E+02	1.15E+02	1.74E+02
	$ITAE_p$	1.79E+03	1.81E+03	1.82E+03

$ITAE_{vp}$: Integral of time absolute value of error for the first 8 sec; $ITAE_p$: ITAE for position, $ITAE_{vl}$: ITAE for the 9-sec start loading speed; IAE_v : integral of absolute value of error for overall 90 sec.

The best total fitness value for speed and position control in the experiment 1b (f_{1b}) is both FLBPI and FLBPID. Therefore, both FLBPI and FLBPID are the best hybrid controllers in the experiment 1b (extreme condition).

Comparing with the simulation results, the best performance is very different because the tuning process was done on the transfer function as a result of s-modeling of the system, but the experiment results are more valid than the simulation results.

The best overshoot in the experiment 2 for speed control is FLBPID controller but the best settling time is FLBPI controller and the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is FLIC. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the FLBPI controller.

Table 5.9 Experiment results of hybrid-fuzzy controllers based on fitness value analysis

Type	PERFORMANCE ITEM	Controller		
		FLBPI	FLBPID	FLIC
1a	f_{vp}	0.00	0.28	0.72
	f_p	0.45	0.00	0.55
	fit_v	0.00	0.72	0.28
	fit_p	0.49	0.24	0.28
	fit_{1a}	0.16	0.56	0.28
1b	f_p	0.50	0.50	0.00
	fit_p	0.50	0.50	0.00
	fit_{1b}	0.50	0.50	0.00
2	f_{vp}	0.54	0.46	0.00
	f_p	0.52	0.00	0.48
	fit_v	0.51	0.49	0.00
	fit_p	0.42	0.00	0.58
	fit_2	0.54	0.46	0.00
3a	f_{vp}	0.61	0.39	0.00
	f_{vpl}	0.72	0.00	0.28
	f_p	0.76	0.24	0.00
	fit_v	0.51	0.42	0.07
	fit_p	0.64	0.36	0.00
	fit_{3a}	0.55	0.40	0.05
3b	f_{vp}	0.82	0.00	0.18
	f_{vpl}	0.00	0.46	0.00
	f_p	0.62	0.00	0.38
	fit_v	0.37	0.21	0.27
	fit_p	0.52	0.29	0.19
	fit_{3b}	0.42	0.24	0.24
4a	f_v	0.50	0.50	0.00
	fit_v	0.50	0.50	0.00
	fit_{4a}	0.50	0.50	0.00
4b	f_{vp}	0.45	0.55	0.00
	f_p	0.71	0.29	0.00
	fit_v	0.32	0.56	0.12
	fit_p	0.71	0.29	0.00
	fit_{4b}	0.45	0.47	0.08

f_{vp} : Fitness value for the first 8-sec starting speed based on $ITAE_{vp}$; f_p : Fitness value for position based on $ITAE_p$; f_{vpl} : Fitness value for the 9-sec start loading speed (from 14 to 23 sec) based on $ITAE_{vpl}$; f_v: Fitness value for overall 90 sec of speed based on IAE_v; fit_v : Total fitness value for speed control; fit_p : Total fitness value for position control; fit_x : Total fitness value for speed and position control in the experiment of Type x

In the position control, the best SSEP is FLIC but the best fitness value based on $ITAE_p$ (f_p) is FLBPI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLIC.

The best total fitness value for speed and position control in the experiment 2 (f_2) is FLBPI controller. Therefore, FLBPI is the best hybrid controller in the experiment 2 (moderate condition).

Comparing with the simulation results, the performance is not exactly the same and the best performance is different because the tuning process was done on the transfer function as a result of s-modeling of the system, but the experiment results are more valid than the simulation results.

The best overshoot, settling time and undershoot in the experiment 3a for speed control is FLBPID controller, but the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is FLBPI controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, undershoot, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is FLBPID controller. When start loading, the best undershoot is FLBPI controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is FLBPI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLBPI controller.

The best total fitness value for speed and position control in the experiment 3a (f_{3a}) is FLBPI controller. Therefore, FLBPI is the best hybrid controller in the experiment 3a (variable load condition).

Comparing with the simulation results, the performance is not exactly the same and the best performance is different because the tuning process was done on the transfer function as a result of s-modeling of the system, but the experiment results are more valid than the simulation results.

The best overshoot in the experiment 3b for speed control is FLIC but the settling time and the fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is FLBPI controller, and the best overshoot 2 (when start unloading) is FLBPID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, overshoot 2, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is the FLBPI controller.

In the position control, the best SSEP is FLBPID controller but the best fitness value based on $ITAE_p$ (f_p) is FLBPI controller. The total fitness value for position

control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLBPI controller.

The best total fitness value for speed and position control in the experiment 3b (f_{3b}) is FLBPI controller. Therefore, FLBPI is the best hybrid controller in the experiment 3b (variable load condition).

Comparing with the simulation results, the performance is not exactly the same but the best controller with the best total fitness value for speed and position control is the same.

The experiment 4a presents the speed performance without overshoot and settling time since it is in the variations of speed set point. The best fitness value based on IAE_v (f_v) is FLBPI controller. The best total fitness value for speed control (fit_v) is obtained based on the IAE_v only, and the best is the FLBPI controller.

The best total fitness value for speed and position control in the experiment 4a (f_{4a}) is FLBPI controller. Therefore, FLBPI is the best hybrid controller in the experiment 4a (variable set point condition).

Comparing with the simulation results, the best performance is different because the tuning process was done on the transfer function as a result of s-modeling of the system, but the experiment results are more valid than the simulation results.

In the experiment 4b, the best overshoot and the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) for speed control is FBPID controller, but the best settling time is FBPI controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the FLBPID controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is FLBPI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the FLBPI controller.

The best total fitness value for speed and position control in the experiment 4b (f_{4b}) is FLBPID controller. Therefore, FLBPID is the best hybrid controller in the experiment 4b (variable set point condition).

Comparing with the simulation results, the best performance is very different because the tuning process was done on the transfer function as a result of s-modeling of the system, but the experiment results are more valid than the simulation results.

5.4.2 Experiment Results Summary of Hybrid-Fuzzy Controllers

The speed and position control experiment of hybrid-fuzzy controller has been presented. FLBPI controller is the best speed controller as compared to FLBPID controller and FLIC for experiment of Type 2, 3a, 3b, and 4a. FLBPID controller is the best speed controller as compared to FLBPI controller and FLIC for experiment of Type 1a, and 4b. FLIC is not as good as FLBPI controller and FLBPID controller as a speed controller. Using Eq. (3 - 45), the best overall speed controller is FLBPID as compared to FLBPI and FLIC.

In both speed and position control, FLBPI is the best controller as compared to FLBPID controller and FLIC for experiment of Type 2, 3a, 3b and 4b. FLBPID controller is the best controller as compared to FLBPI controller and FLIC for simulation of Type 1a, 1b, and 4b. FLIC is not as good as FLBPI controller and FLBPID controller as both speed and position controller. Using Eq. (3 - 45), the best overall speed and position controller is FLBPI as compared to FLBPID and FLIC.

Comparing with the simulation results, the real-time implementation results tend to be similar to the simulation results.

5.5 Experiment of SPOGA Optimized Controllers

There are three types of controllers in this simulation: SPOGA-FLBPI controller, SPOGA-FLPID controller, and SPOGA-FLIC. The experiment was done as a result of optimization using SPOGA.

There are seven types of hardware experiment as explained in Section 3.7 which are summarized in Table 5.3.

5.5.1 Results of SPOGA Optimized Controllers in Real-time Implementation

The comparison on the effectiveness of implementing SPOGA optimized hybrid-fuzzy controllers for hardware experiment based on the second order underdamped response analysis is presented in Table 5.10, the comparison based on error analysis is presented in Table 5.11, and the comparison based on fitness value analysis is presented in Table 5.12.

It is shown in Table 5.10, Table 5.11, and Table 5.12 that the best overshoot in the experiment 1a is SPOGA-FLBPI controller, but the best settling time and fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is SPOGA-FLBPID controller for speed control. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the SPOGA-FLBPID controller.

Table 5.10 Experiment results of SPOGA optimized hybrid-fuzzy controllers based on second order underdamped response analysis

Type	PERFORMANCE ITEM	Controller		
		SPOGA-FLBPI	SPOGA-FLBPID	SPOGA-FLIC
1a	Overshoot ($\%O_S$, %)	14.55	16.83	44.52
	Settling time (t_s, sec)	1.79	1.74	2.06
	SSEP ($\%S_p$, %)	0.09	0.07	0.08
1b	SSEP ($\%S_p$)	0.48	0.39	0.39
2	Overshoot ($\%O_S$, %)	3.72	4.29	2.27
	Settling time (t_s, sec)	1.43	1.05	2.05
	SSEP ($\%S_p$, %)	0.07	0.05	0.09
3a	Overshoot ($\%O_S$, %)	2.72	3.59	3.10
	Settling time (t_s, sec)	1.48	1.01	2.05
	Undershoot ($\%U_S$, %)	38.39	32.38	41.35
	SSEP ($\%S_p$, %)	0.05	0.02	0.10
3b	Overshoot ($\%O_S$, %)	4.85	2.20	3.79
	Settling time (t_s, sec)	1.36	1.08	3.12
	Overshoot 2 ($\%O_{S2}$, %)	48.72	49.62	59.43
	SSEP ($\%S_p$, %)	0.01	0.08	0.015
4b	Overshoot ($\%O_S$, %)	3.67	6.75	3.10
	Settling time (t_s, sec)	1.38	1.21	2.21
	SSEP ($\%S_p$, %)	0.08	0.07	0.08

SSEP: Steady State Error of Position

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPID controller.

Table 5.11 Experiment results of SPOGA optimized hybrid-fuzzy controllers based on error analysis

Type	PERFORMANCE ITEM	Controller		
		SPOGA-FLBPI	SPOGA-FLBPID	SPOGA-FLIC
1a	$ITAE_{vp}$	5.81E+01	5.43E+01	8.55E+01
	$ITAE_p$	5.95E+03	5.92E+03	5.93E+03
1b	$ITAE_p$	1.24E+01	1.05E+01	1.21E+01
2	$ITAE_{vp}$	1.15E+02	1.06E+02	1.72E+02
	$ITAE_p$	3.80E+02	3.72E+02	3.90E+02
3a	$ITAE_{vp}$	9.98E+01	9.03E+01	1.48E+02
	$ITAE_{vl}$	1.16E+03	1.91E+03	2.13E+03
	$ITAE_p$	1.30E+03	1.36E+03	1.49E+03
3b	$ITAE_{vp}$	1.30E+02	2.40E+02	3.23E+02
	$ITAE_{vl}$	1.50E+03	1.42E+03	2.11E+03
	$ITAE_p$	1.30E+03	1.35E+03	1.30E+03
4a	IAE_v	8.89E+02	8.19E+02	1.30E+03
4b	$ITAE_{vp}$	1.17E+02	9.13E+01	2.10E+02
	$ITAE_p$	1.80E+03	1.79E+03	1.88E+03

$ITAE_{vp}$: Integral of time absolute value of error for the first 8 sec; $ITAE_p$: ITAE for position, $ITAE_{vl}$: ITAE for the 9-sec start loading speed; IAE_v : integral of absolute value of error for overall 90 sec.

The best total fitness value for speed and position control in the experiment 1a (f_{1a}) is SPOGA-FLBPID controller. Therefore, SPOGA-FLBPID is the best hybrid-fuzzy controller in the experiment 1a (extreme condition).

Comparing with the simulation results, the best performance is different because the tuning process was done on the transfer function as a result of s-modeling of the system, but the experiment results are more valid than the simulation results.

The simulation 1b presents the position performance since it is in the extreme condition with maximum speed and minimum position. The best SSEP and fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPID controller. The best total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPID controler.

The best total fitness value for speed and position control in the experiment 1b (f_{1b}) is SPOGA-FLBPID controller. Therefore, SPOGA-FLBPID controller is the best hybrid-fuzzy controller in the experiment 1b (extreme condition).

Comparing with the simulation results, the best performance is different because the tuning process was done on the transfer function as a result of s-modeling of the system, but the experiment results are more valid than the simulation results.

Table 5.12 Experiment results of SPOGA optimized hybrid-fuzzy controllers based on fitness value analysis

Type	PERFORMANCE ITEM	Controller		
		SPOGA-FLBPI	SPOGA-FLBPID	SPOGA-FLIC
1a	f_{vp}	0.47	0.53	0.00
	f_p	0.00	0.59	0.41
	fit_v	0.48	0.52	0.00
	fit_p	0.00	0.58	0.42
	fit_{1a}	0.32	0.54	0.14
1b	f_p	0.00	0.86	0.14
	fit_p	0.00	0.68	0.32
	fit_{1b}	0.00	0.68	0.32
2	f_{vp}	0.46	0.54	0.00
	f_p	0.36	0.64	0.00
	fit_v	0.35	0.38	0.26
	fit_p	0.33	0.67	0.00
	fit_2	0.35	0.48	0.17
3a	f_{vp}	0.46	0.54	0.00
	f_{vpl}	0.82	0.18	0.00
	f_p	0.60	0.40	0.00
	fit_v	0.57	0.34	0.09
	fit_p	0.55	0.45	0.00
	fit_{3a}	0.56	0.38	0.06
3b	f_{vp}	0.70	0.30	0.00
	f_{vpl}	0.47	0.53	0.00
	f_p	0.52	0.00	0.48
	fit_v	0.41	0.52	0.07
	fit_p	0.52	0.00	0.48
	fit_{3b}	0.44	0.35	0.21
4a	f_v	0.46	0.54	0.00
	fit_v	0.46	0.54	0.00
	fit_{4a}	0.46	0.54	0.00
4b	f_{vp}	0.44	0.56	0.00
	f_p	0.45	0.55	0.00
	fit_v	0.45	0.37	0.18
	fit_p	0.26	0.74	0.00
	fit_{4b}	0.39	0.49	0.12

f_{vp} : Fitness value for the first 8-sec starting speed based on $ITAE_{vp}$; f_p : Fitness value for position based on $ITAE_p$; f_{vpl} : Fitness value for the 9-sec start loading speed (from 14 to 23 sec) based on $ITAE_{vpl}$; f_v: Fitness value for overall 90 sec of speed based on IAE_v; fit_v : Total fitness value for speed control; fit_p : Total fitness value for position control; fit_x : Total fitness value for speed and position control in the experiment of Type x

The best overshoot for speed control in the experiment 2 is SPOGA-FLIC, but the best settling time and the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) for speed control is SPOGA-FLBPID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the SPOGA-FLBPID controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPID controller.

The best total fitness value for speed and position control in the experiment 2 (f_2) is SPOGA-FLBPID controller. Therefore, SPOGA-FLBPID is the best hybrid-fuzzy controller in the experiment 2 (moderate condition).

Comparing with the simulation results, the best performance is different because the tuning process was done on the transfer function as a result of s-modeling of the system, but the experiment results are more valid than the simulation results.

In the experiment 3a, the best overshoot, settling time and the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) for speed control is SPOGA-FLBPID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, undershoot, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is SPOGA-FLBPI controller. When start loading, the best undershoot is FLBPID controller.

In the position control, the best SSEP is SPOGA-FLBPID controller but the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPI controller.

The best total fitness value for speed and position control in the experiment 3a (f_{3a}) is SPOGA-FLBPI controller. Therefore, SPOGA-FLBPI controller is the best hybrid-fuzzy controller in the experiment 3a (variable load)

Comparing with the simulation results, the performance is not exactly the same but the best controller with the best total fitness value for speed and position control is the same.

The best overshoot and settling for speed control in the experiment 3b is SPOGA-FLBPID controller, but the the best fitness value for the first 8-second starting speed based on $ITAE_{vp}$ (f_{vp}) is SPOGA-FLBPI controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, overshoot 2, $ITAE_{vpl}$ and $ITAE_{vp}$ and the best is SPOGA-FLBPID controller. The best overshoot 2 when start unloading is FLBPIGA controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPI controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPI controller.

The best total fitness value for speed and position control in the experiment 3b (f_{3b}) is SPOGA-FLBPID controller. Therefore, SPOGA-FLBPID controller is the best hybrid-fuzzy controller in the experiment 3b (variable load condition).

Comparing with the simulation results, the best performance is different because the tuning process was done on the transfer function as a result of s-modeling of the system, but the experiment results are more valid than the simulation results.

The experiment 4a presents the speed performance without overshoot and settling time since it is in the variations of speed set point. The best fitness value based on IAE_v (f_v) is SPOGA-FLBPID controller. The best total fitness value for speed control (fit_v) is obtained based on the IAE_v only, and the best is the SPOGA-FLBPID controller.

The best total fitness value for speed and position control in the experiment 4a (f_{4a}) is SPOGA-FLBPID controller. Therefore, SPOGA-FLBPID controller is the best hybrid-fuzzy controller in the experiment 4a, as in the simulation experiment.

The best overshoot for speed control in the experiment 4b is SPOGA-FLIC. The best settling time and the best fitness value for the first 8-second starting speed based

on $ITAE_{vp}$ (f_{vp}) is SPOGA-FLBPID controller. The total fitness value for speed control (fit_v) is obtained based on the overshoot, settling time, and $ITAE_{vp}$ and the best is the SPOGA-FLBPI controller.

In the position control, the best SSEP and the best fitness value based on $ITAE_p$ (f_p) is SPOGA-FLBPID controller. The total fitness value for position control (fit_p) is obtained based on the SSEP and $ITAE_p$, and the best is the SPOGA-FLBPID controller.

The best total fitness value for speed and position control in the experiment 4b (f_{4b}) is SPOGA-FLBPI controller. Therefore, SPOGA-FLBPI controller is the best hybrid-fuzzy controller in the experiment 4b (variable set point condition).

Comparing with the simulation results, the performance is not exactly the same but the best controller with the best total fitness value for speed and position control is the same.

5.5.2 Experiment Results Summary of SPOGA Optimized Hybrid-Fuzzy Controllers

The speed and position control experiment of SPOGA optimized hybrid-fuzzy controller has been presented. SPOGA-FLBPI controller is the best speed controller as compared to SPOGA-FLBPID controller and SPOGA-FLIC for experiment of Type 3a and 4b. SPOGA-FLBPID controller is the best speed controller as compared to SPOGA-FLBPI controller and SPOGA-FLIC for experiment of Type 1a, 2, 3b, and 4a. SPOGA-FLIC is not as good as SPOGA-FLBPI controller and SPOGA-FLBPID controller for speed control. Using Eq. (3 - 45), the best overall speed controller is SPOGA-FLBPI as compared to SPOGA-FLBPID and SPOGA-FLIC.

In the both speed and position control, SPOGA-FLBPI controller is the best speed controller as compared to SPOGA-FLBPID controller and SPOGA-FLIC for experiment of Type 3a and 3b. SPOGA-FLBPID controller is the best speed controller as compared to SPOGA-FLBPI controller and SPOGA-FLIC for experiment of Type 1a, 1b, 2, 4a and 4b. SPOGA-FLIC is not as good as SPOGA-

FLBPI controller and SPOGA-FLBPID controller for speed and position control. Using Eq. (3 - 45), the best overall speed and position controller is SPOGA-FLBPID controller as compared to SPOGA-FLBPI and SPOGA-FLIC.

Comparing with the simulation results, the real-time implementation results tend to be similar to the simulation results.

5.6 Performance Comparisons of SPOGA to non-SPOGA Controllers

This section presents the performance improvements of non-SPOGA hybrid-fuzzy controllers and SPOGA optimized hybrid-fuzzy controllers based on the fitness values where the performance items are based on Table 5.7 to Table 5.12.

There are seven types of hardware experiment as explained in Section 3.7 which are summarized in Table 5.3.

The graphs of input-output characteristic of speed error, speed, and position between the best improvement of SPOGA optimized hybrid controllers and the corresponding hybrid controllers are presented in this section for each experiment type.

5.6.1 Results of Performance Comparisons of SPOGA to non-SPOGA Controllers in Real-time Implementation

The comparison on the improvement of SPOGA optimized and non-SPOGA hybrid-fuzzy controllers based on the performance metrics for hardware experiment is presented in Table 5.13 where the performance items are based on Table 5.9 and Table 5.12.

It is shown in Table 5.13 that SPOGA-FLBPI in the experiment 1a makes the best fit improvement for speed control. This means that SPOGA-FLBPI makes the best fit improvement in the experiment 1a. The best position controller is FLBPI but it is not optimized by SPOGA. The improvement of SPOGA-FLIC is less than zero, this

means that the SPOGA-FLIC cannot make improvement in the experiment 1a. The graphical comparisons of SPOGA-FLBPI to FLBPI are shown in Fig. 5.1 and Fig. 5.2.

It is shown in Fig. 5.1 and Fig. 5.2 that SPOGA-FLBPI has no overshoot and the settling time is faster than FLBPI. This makes the absolute error of SPOGA-FLBPI is smaller than FLBPI.

In the experiment 1b, SPOGA-FLBPID has the best fit for position control but the SPOGA is not optimize the position controller.

Table 5.13 Performance improvement comparison of SPOGA optimized and non-SPOGA hybrid-fuzzy controllers for hardware experiment

Type	PERFORMANCE ITEMS	Controller					
		FLBPI	SPOGA-FLBPI	FLBPID	SPOGA-FLBPID	FLIC	SPOGA-FLIC
1a	fit_v	0.08	0.26	0.20	0.27	0.14	0.05
	fit_p	0.30	0.02	0.17	0.16	0.24	0.12
	fit	0.15	0.18	0.19	0.23	0.17	0.08
	I_{pv1a}		0.18		0.07		-0.08
1b	fit_v	-	-	-	-	-	-
	fit_p	0.10	0.24	0.10	0.28	0.00	0.27
	fit	0.10	0.24	0.10	0.28	0.00	0.27
	I_{pv1b}		-		-		-
2	fit_v	0.22	0.20	0.21	0.21	0.01	0.15
	fit_p	0.21	0.14	0.08	0.29	0.29	0.00
	fit	0.21	0.19	0.16	0.24	0.10	0.10
	I_{pv2}		-0.01		0.01		0.14
3a	fit_v	0.18	0.30	0.13	0.22	0.04	0.12
	fit_p	0.26	0.25	0.19	0.21	0.09	0.00
	fit	0.20	0.29	0.15	0.21	0.06	0.08
	I_{pv3a}		0.12		0.09		0.08
3b	fit_v	0.19	0.20	0.16	0.25	0.09	0.03
	fit_p	0.27	0.25	0.14	0.00	0.12	0.23
	fit	0.22	0.21	0.15	0.17	0.10	0.10
	I_{pv3b}		0.01		0.09		-0.06
4a	fit_v	0.22	0.22	0.22	0.26	0.09	0.00
	fit_p	-	-	-	-	-	-
	fit	0.22	0.22	0.22	0.26	0.09	0.00
	I_{pv4a}		-0.00		0.04		-0.09
4b	fit_v	0.13	0.22	0.24	0.19	0.09	0.12
	fit_p	0.32	0.15	0.18	0.24	0.08	0.04
	fit	0.19	0.20	0.22	0.21	0.09	0.09
	I_{pv4b}		0.09		-0.05		0.03

In the experiment 2, SPOGA-FLIC makes the best fit improvement in the experiment 1a eventhough the performance of SPOGA-FLIC is not as good as

SPOGA-FLBPI and SPOGA-FLBPID. The improvement of SPOGA-FLBPI is less than zero, this means that both SPOGA-FLBPI cannot make improvement in the experiment 2. The graphical comparisons of SPOGA-FLIC to FLIC are shown in Fig. 5.3 and Fig. 5.4.

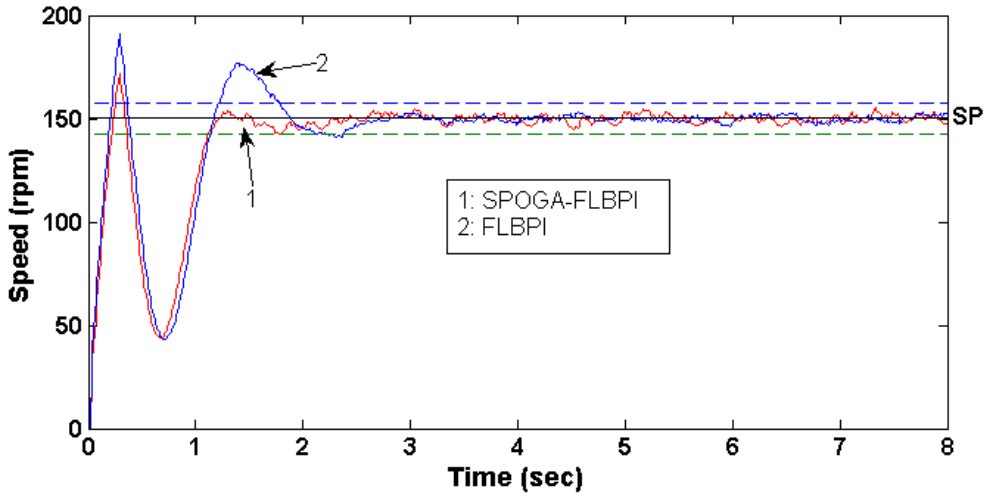


Fig. 5.1 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for experiment 1a (see Fig. D.6)

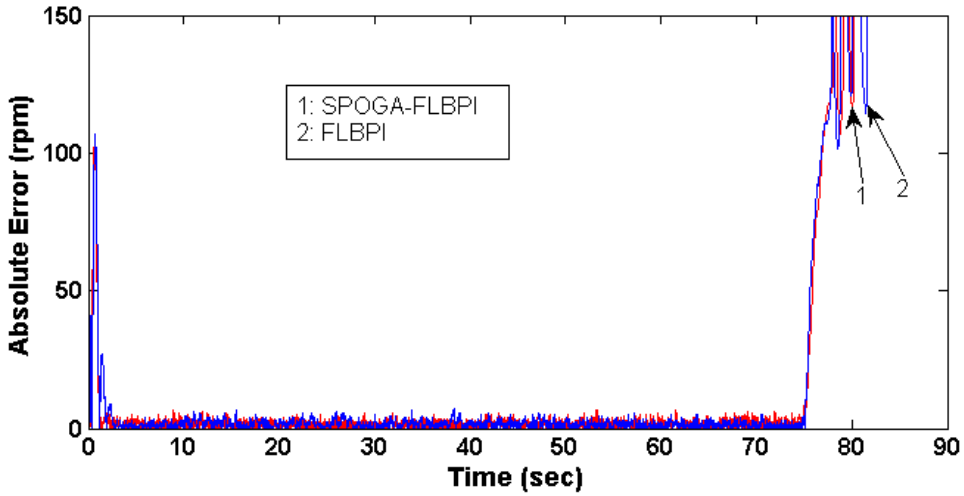


Fig. 5.2 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for experiment 1a

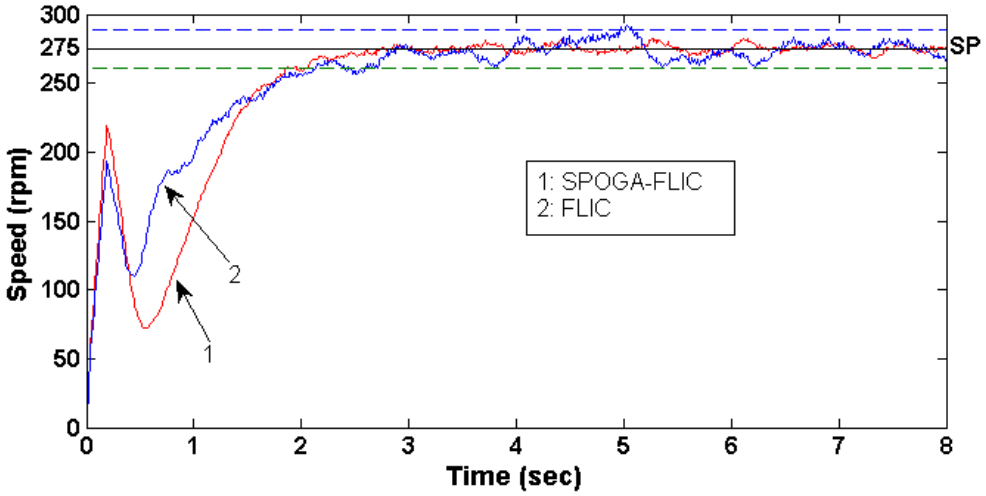


Fig. 5.3 Step response of speed control of DC servomotor using SPOGA-FLIC vs. FIC for experiment 2 (see Fig. D.7)

It shown in the Fig. 5.3 and Fig. 5.4 that SPOGA-FLIC has smaller overshoot and settling time than FLIC. This makes the absolute error of SPOGA-FLIC smaller than FLIC. Actually, SPOGA-FLIC has not the best performace as compared to SPOGA-FLBPI and SPOGA-FLBPID but the the SPOGA improvement is the best as compared to FLBPI and FLBPID.

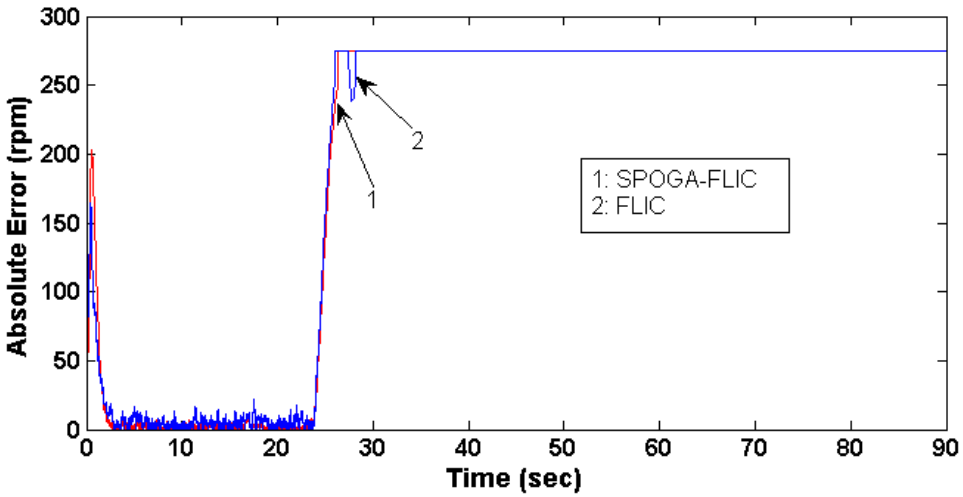


Fig. 5.4 Absolute error of speed control of DC servomotor using SPOGA-FLIC vs. FLIC for experiment 2

In the experiment 3a, SPOGA-FLBPI makes the best fit improvement in the experiment 3a. The graphical comparisons of SPOGA-FLBPI to FLBPI are shown in Fig. 5.5 and Fig. 5.6.

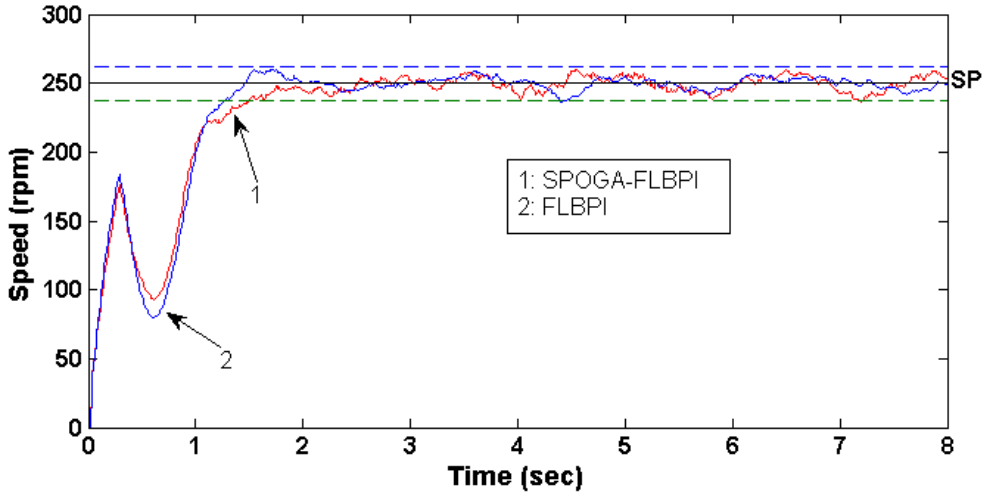


Fig. 5.5 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for experiment 3a (see Fig. D.8)

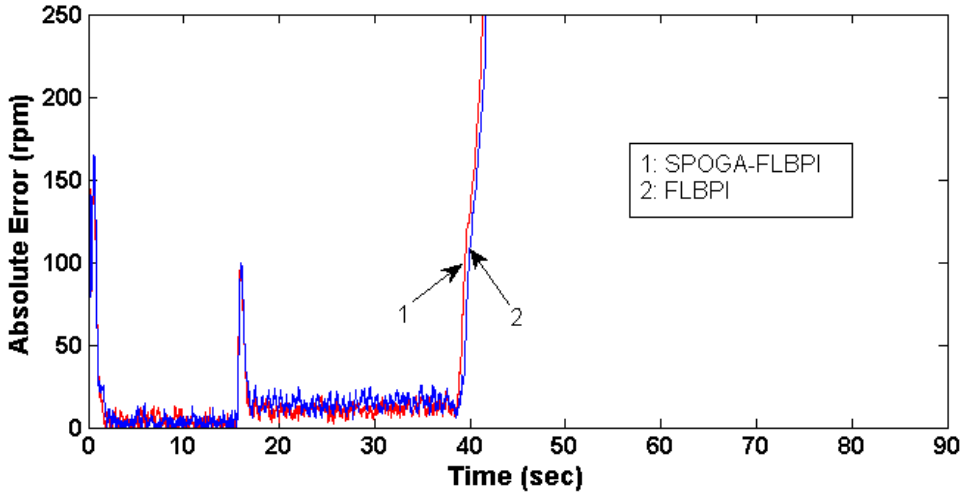


Fig. 5.6 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for experiment 3a

It shown in the Fig. 5.5 and Fig. 5.6 that SPOGA-FLBPI is similar to FLBPI, based on Table 5.8 and Table 5.11, the ITAE of SPOGA-FLBPI is smaller than the ITAE of FLBPI. This makes the fit of SPOGA-FLBPI better than FLBPI.

In the experiment 3b, SPOGA-FLBPID makes the best fit improvement. The improvement of SPOGA-FLIC is less than zero, this means that SPOGA-FLIC cannot make improvement in the experiment 3b. The graphical comparisons of SPOGA-FLBPID to FLBPID are shown in Fig. 5.7 and Fig. 5.8.

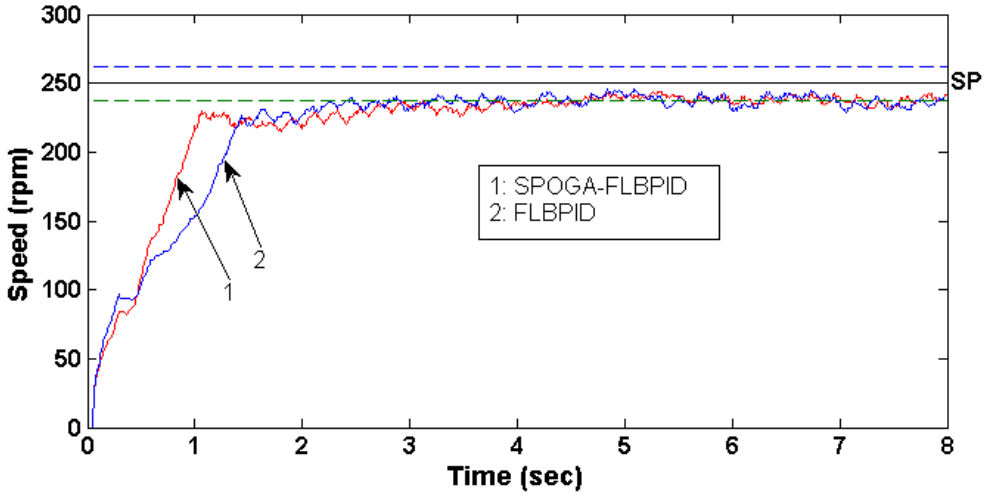


Fig. 5.7 Step response of speed control of DC servomotor using SPOGA-FLBPID vs. FLBPID for experiment 3b (see Fig. D.9)

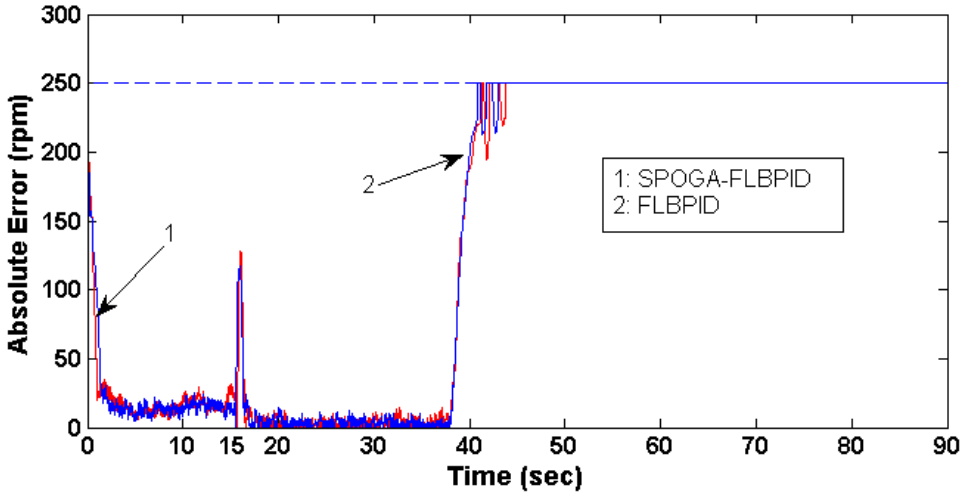


Fig. 5.8 Absolute error of speed control of DC servomotor using SPOGA-FLBPID vs. FLBPID for experiment 3b

It shown in the Fig. 5.7 and Fig. 5.8 that SPOGA-FLBPI has smaller settling time and overshoot 2 than FLBPI.

In the experiment 4a, SPOGA-FLBPID makes the best fit improvement. The improvement of SPOGA-FLIC and GAFLBPI are less than zero, this means that both SPOGA-FLIC and SPOGA-FLBPI cannot make improvement in the experiment 4a. The graphical comparisons of SPOGA-FLBPID to FLBPID are shown in Fig. 5.9 and Fig. 5.10.

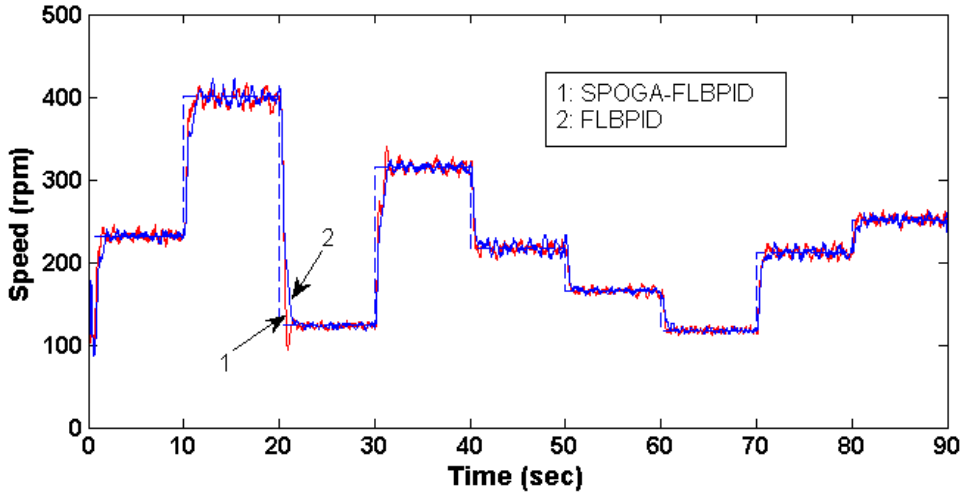


Fig. 5.9 Speed control of DC servomotor using SPOGA-FLBPID vs. FLBPID for experiment 4a

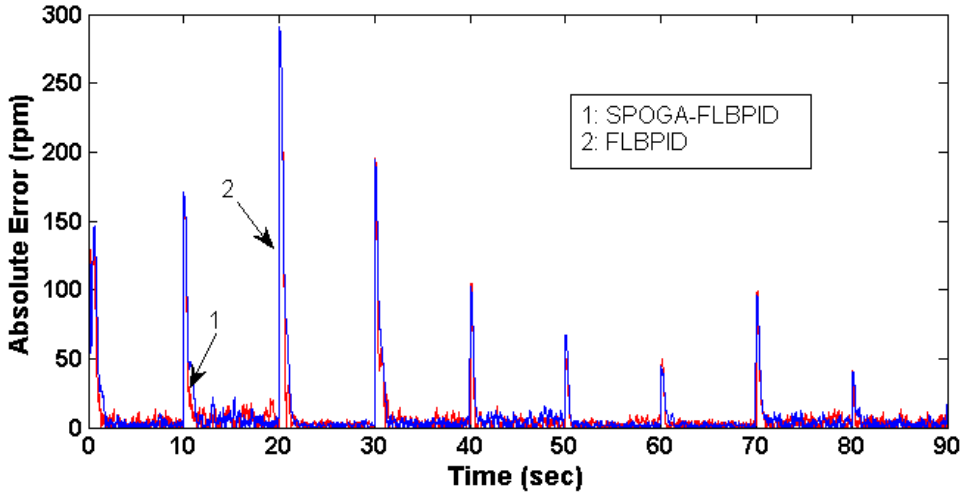


Fig. 5.10 Absolute error of speed control of DC servomotor using SPOGA-FLBPID vs. FLBPID for experiment 4a

It shown in the Fig. 5.9 and Fig. 5.10 that SPOGA-FLBPI has smaller settling time than FLBPI. This makes the absolute error of SPOGA-FLBPI smaller than FLBPI.

In the experiment 4b, SPOGA-FLBPI makes the best fit improvement. The improvement of SPOGA-FLBPID is less than zero, this means that SPOGA-FLBPID cannot any improvement in the experiment 4b. The graphical comparisons of SPOGA-FLBPI to FLBPI are shown in Fig. 5.11 to Fig. 5.13.

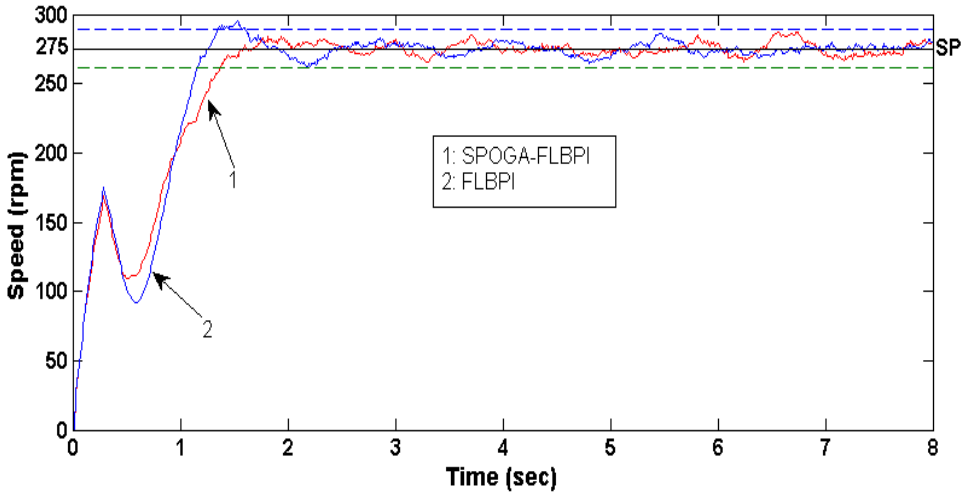


Fig. 5.11 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for experiment 4b (see Fig. D.10)

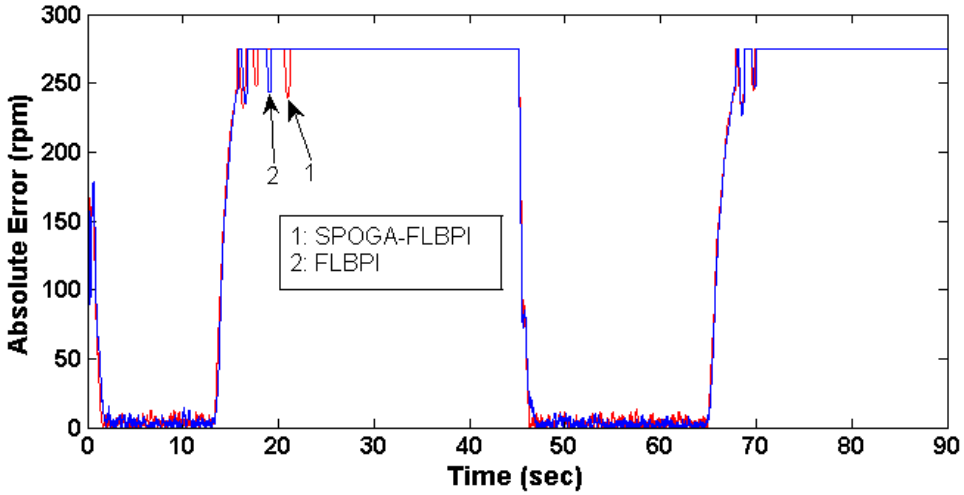


Fig. 5.12 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. FLBPI for experiment 4b

It shown in the Fig. 5.11 and Fig. 5.12 that SPOGA-FLBPI has smaller overshoot than FLBPI. This makes the absolute error of SPOGA-FLBPI smaller than FLBPI.

It is shown in Fig. 5.13 that SPOGA-FLBPI is similar to FLBPI. This means that SPOGA-FLBPI does not make any improvement in position control, but the SPOGA is for optimizing the speed controller only.

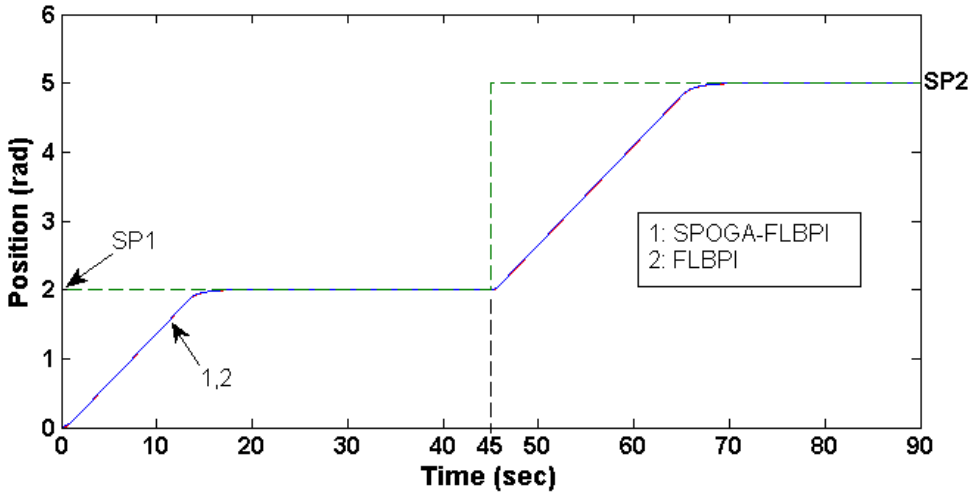


Fig. 5.13 Position control of DC servomotor using SPOGA-FLBPI vs. FLPI for experiment 4b

5.6.2 Experiment Results Summary of SPOGA optimized and non-SPOGA optimized Hybrid-Fuzzy Controllers

The improvement tests of GA optimized hybrid-fuzzy controller to non-GA optimized hybrid PID-fuzzy controller have been presented. It was shown that SPOGA-FLBPI can make fit improvement in all of the experiment types except in the simulation of Type 2 and 4a which failed. SPOGA-FLBPID can make improvement in all of the experiment types except in the simulation of Type 4b which failed. SPOGA-FLIC can make improvement in the experiment of Type 2, 3a, and 4b, and fails to make improvement in the experiment of Type 1a, 3b, and 4a

The experiment results show that based on Eq. (3 - 47), the best total improvement of speed control is SPOGA-FLBPI. This means that the SPOGA can optimize the parameters of FLBPI successfully.

Comparing with the simulation results, the real-time implementation results tend to be similar to the simulation results.

5.7 Performance Comparisons of Conventional, Fuzzy, and SPOGA Optimized Hybrid-Fuzzy Controllers

This section presents the performance comparisons of conventional (PI and PID controllers), FLC, and SPOGA optimized hybrid-fuzzy controllers based on the fitness values where the performance items are based on Table 5.6 and Table 5.12.

There are seven types of hardware experiment as explained in Section 3.7 which are summarized in Table 5.3.

The graphs of input-output characteristic of speed error, position error, speed, and position between the best conventional and fuzzy controllers and the best SPOGA optimized hybrid controllers are presented in this section for each experiment type.

5.7.1 Results on Performance Comparisons of Conventional, Fuzzy, and SPOGA Optimized Hybrid-Fuzzy Controllers

The comparison on the effectiveness of implementing conventional, fuzzy, and hybrid-fuzzy controllers based on the performance metrics for hardware experiment is presented in Table 5.14 where the performance items are based on Table 5.6 and Table 5.12.

It is shown in Table 5.14 that the best speed controller in the experiment 1a is PI for conventional controller and SPOGA-FLBPID for hybrid controller, and the best position controller is PID for conventional controller and SPOGA-FLBPID for hybrid controller. As a speed and position controller, PI is the best conventional controller and SPOGA-FLBPID is the best hybrid controller. Comparison on the best conventional to the best hybrid are shown in the table that SPOGA optimized hybrid-fuzzy controllers are better then conventional controllers in the experiment 1a. The graphical comparisons are shown in the Fig. 5.14 to Fig. 5.17.

It is shown in Fig. 5.14 and Fig. 5.15 that SPOGA-FLBPI has the faster settling time than PI where both have no overshoot. Consequently, the absolute error of SPOGA-FLBPI is smaller than PI.

Table 5.14 Performance comparisons of conventional, fuzzy, and SPOGA optimized hybrid-fuzzy controllers for hardware experiment

TYPE	CONTROLLERS	Controller		
		fit_v	fit_p	fit_{1a}
1a	PI	0.20	0.12	0.18
	PID	0.16	0.17	0.16
	FLC	0.06	0.00	0.04
	SPOGA-FLBPI	0.22	0.22	0.22
	SPOGA-FLBPID	0.22	0.25	0.23
	SPOGA-FLIC	0.14	0.24	0.17
1b	PI	-	0.18	0.18
	PID	-	0.19	0.19
	FLC	-	0.00	0.00
	SPOGA-FLBPI	-	0.21	0.21
	SPOGA-FLBPID	-	0.21	0.21
	SPOGA-FLIC	-	0.21	0.21
2	PI	0.16	0.19	0.17
	PID	0.20	0.20	0.20
	FLC	0.00	0.00	0.00
	SPOGA-FLBPI	0.22	0.20	0.21
	SPOGA-FLBPID	0.22	0.21	0.22
	SPOGA-FLIC	0.20	0.19	0.20
3a	PI	0.16	0.15	0.16
	PID	0.16	0.26	0.19
	FLC	0.00	0.04	0.01
	SPOGA-FLBPI	0.28	0.27	0.27
	SPOGA-FLBPID	0.22	0.24	0.23
	SPOGA-FLIC	0.18	0.04	0.13
3b	PI	0.18	0.20	0.18
	PID	0.16	0.21	0.18
	FLC	0.01	0.00	0.01
	SPOGA-FLBPI	0.20	0.21	0.20
	SPOGA-FLBPID	0.24	0.17	0.22
	SPOGA-FLIC	0.11	0.21	0.14
4a	PI	0.17	-	0.18
	PID	0.22	-	0.22
	FLC	0.00	-	0.00
	SPOGA-FLBPI	0.22	-	0.22
	SPOGA-FLBPID	0.23	-	0.23
	SPOGA-FLIC	0.15	-	0.15
4b	PI	0.17	0.14	0.16
	PID	0.21	0.15	0.19
	FLC	0.00	0.00	0.00
	SPOGA-FLBPI	0.22	0.16	0.20
	SPOGA-FLBPID	0.21	0.18	0.20
	SPOGA-FLIC	0.19	0.09	0.16

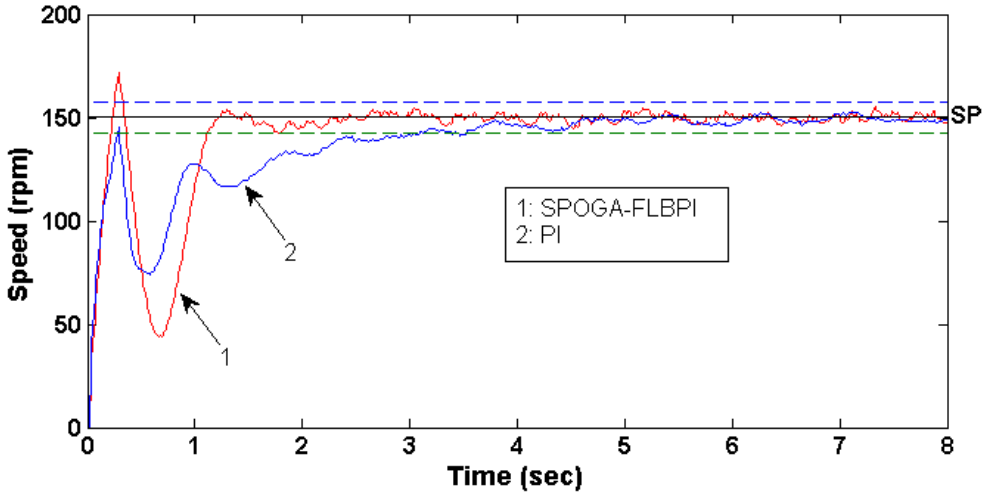


Fig. 5.14 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. PI for experiment 1a (see Fig. D.11)

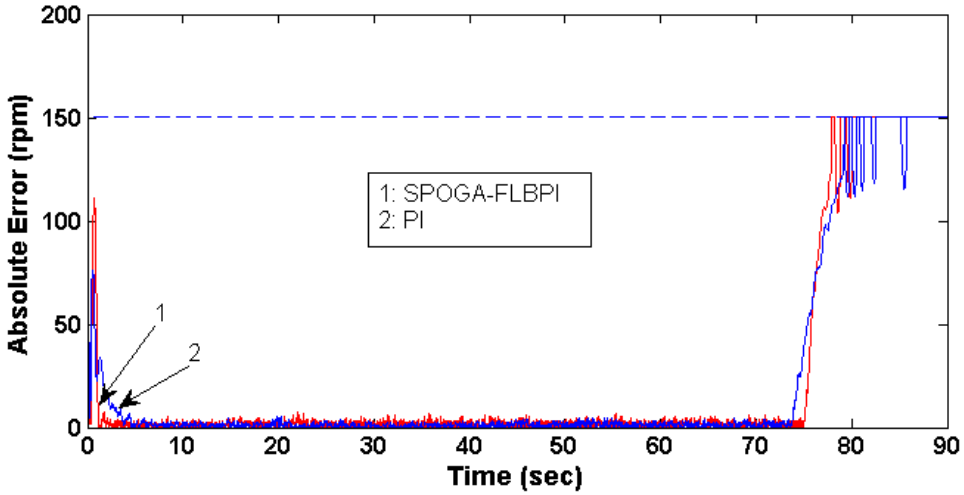


Fig. 5.15 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. PI for experiment 1a

It is shown in Fig. 5.16 and Fig. 5.17 that SPOGA-FLBPID has the faster settling time than the PID. This makes the absolute error of SPOGA-FLBPID is smaller than PID.

In the experiment 1b, the best position controller is PID for conventional controller and SPOGA-FLBPID for hybrid controller. As a speed and position controller, PID is the best conventional controller and SPOGA-FLBPID is the best hybrid controller. Comparison on the best conventional to the best hybrid are shown in the table that SPOGA optimized hybrid-fuzzy controllers are better then

conventional controllers in the experiment 1b. The graphical comparisons are shown in the Fig. 5.18 and Fig. 5.19.

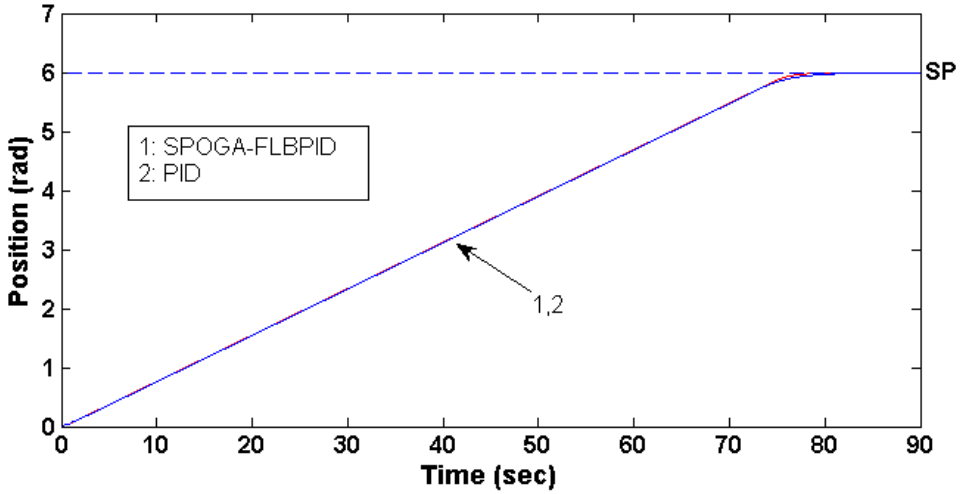


Fig. 5.16 Position control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 1a

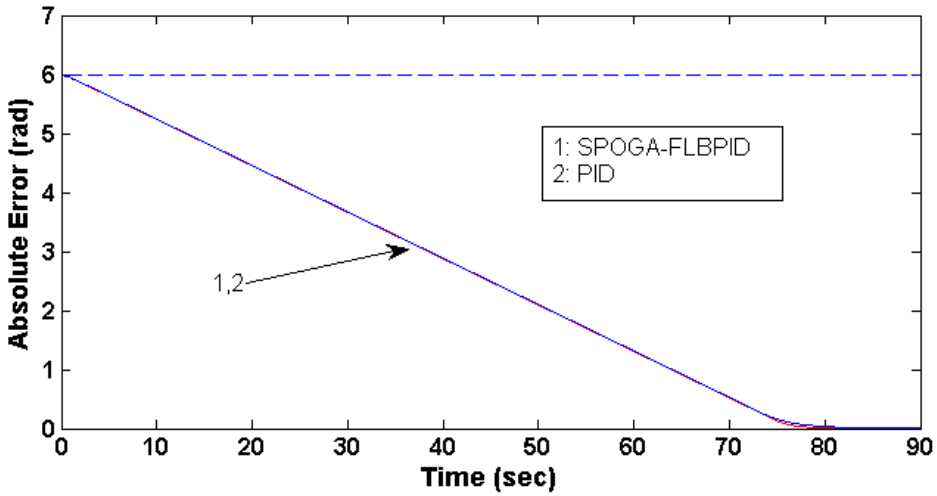


Fig. 5.17 Absolute error of position control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 1a

It is shown in Fig. 5.18 and Fig. 5.19 that SPOGA-FLBPID can reach the setpoint while PID has steady state error. This makes the absolute error of PID larger than SPOGA-FLBPID.

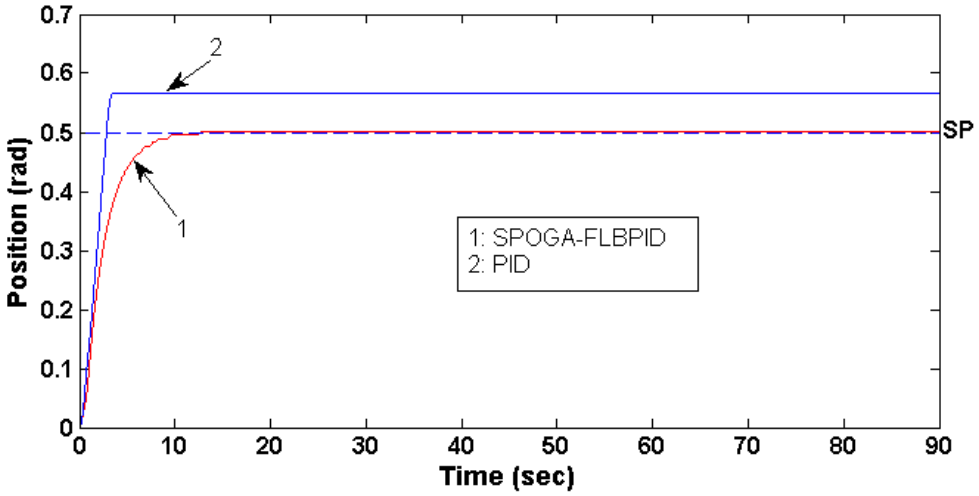


Fig. 5.18 Position control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 1b

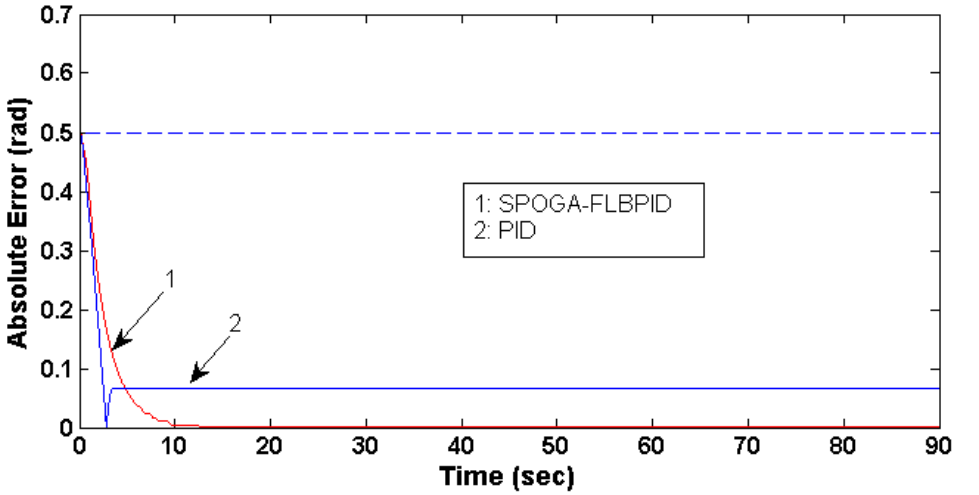


Fig. 5.19 Absolute error of position control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 1b

In the experiment 2, the best speed controller is PID for conventional controller and SPOGA-FLBPID for hybrid-fuzzy controller, and the best position controller is PID for conventional controller and SPOGA-FLBPID for hybrid-fuzzy controller. As a speed and position controller, PID is the best conventional controller and SPOGA-FLBPID is the best hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that SPOGA optimized hybrid-fuzzy controllers are better then conventional controllers in the experiment 2. The graphical comparisons are shown in the Fig. 5.20 to Fig. 5.23.

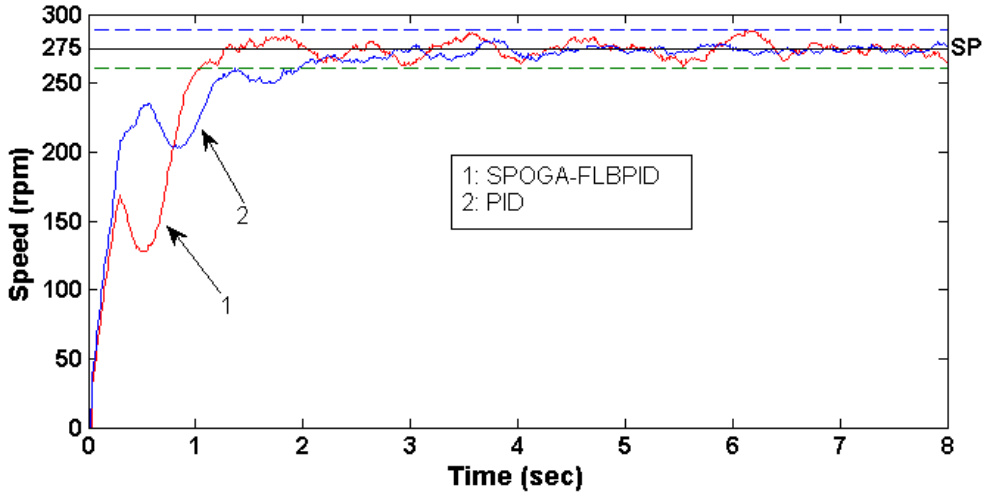


Fig. 5.20 Step response of speed control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 2 (see Fig. D.12)

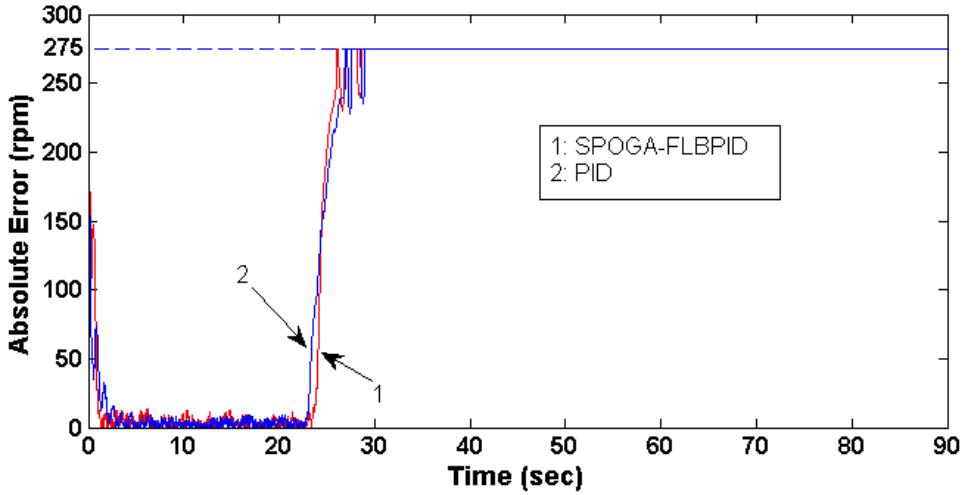


Fig. 5.21 Absolute error of speed control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 2

It is shown in Fig. 5.20 and Fig. 5.21 that SPOGA-FLBPID has the faster settling time than PID. Consequently, the absolute error of SPOGA-FLBPID is smaller than PID.

It is shown in Fig. 5.22 and Fig. 5.23 that SPOGA-FLBPID has the faster settling time than PID. This makes the absolute error of SPOGA-FLBPID smaller than PID.

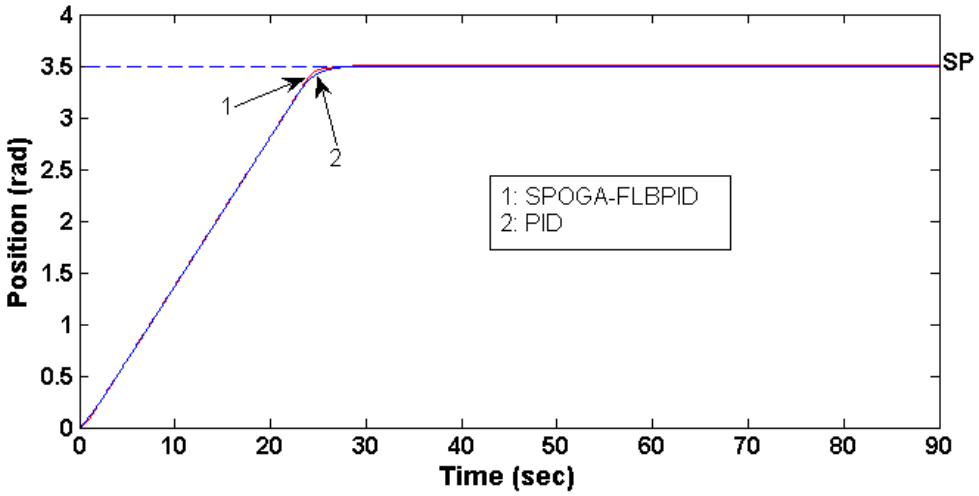


Fig. 5.22 Position control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 2

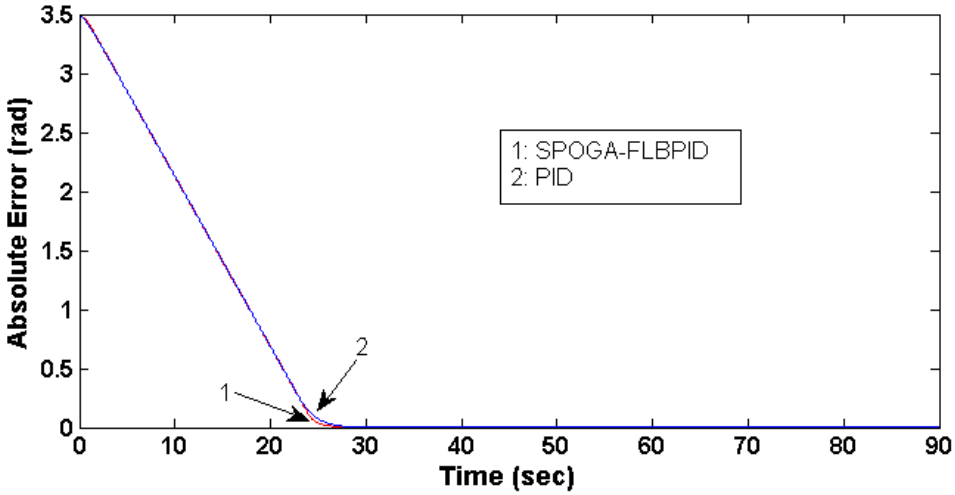


Fig. 5.23 Absolute error of position control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 2

In the experiment 3a, the best speed controller is PID for conventional controller and SPOGA-FLBPI for hybrid-fuzzy controller, and the best position controller is PID for conventional controller and SPOGA-FLBPI for hybrid-fuzzy controller. As a speed and position controller, PID is the best conventional controller and SPOGA-FLBPI is the best hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that hybrid-fuzzy controllers are better then conventional controllers in the experiment 3a. The graphical comparisons are shown in the Fig. 5.24 to Fig. 5.27.

Based on Fig. 5.24 and Fig. 5.25, SPOGA-FLBPI has the faster settling time than PID either unloaded or loaded eventhough there is a very small of overshoot. Consequently, the absolute error of SPOGA-FLBPI is smaller than PID.

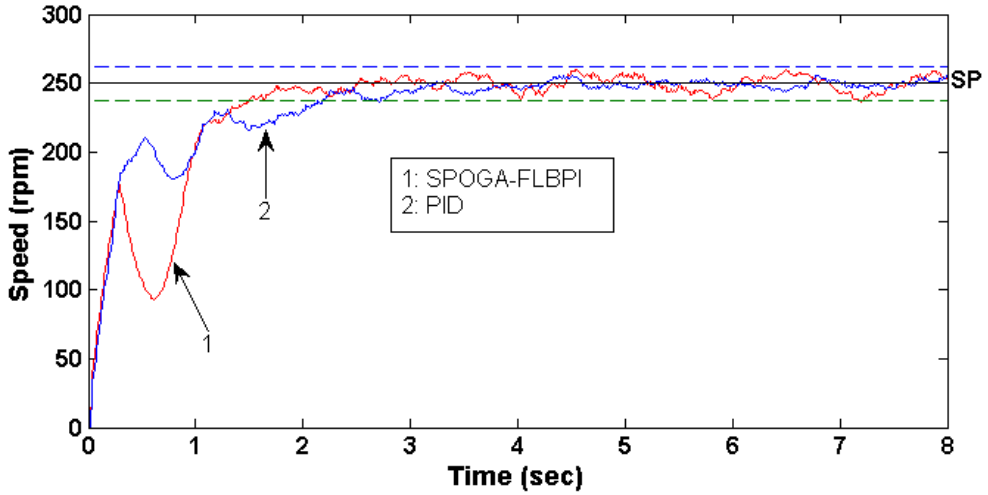


Fig. 5.24 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. PID for experiment 3a (see Fig. D.13)

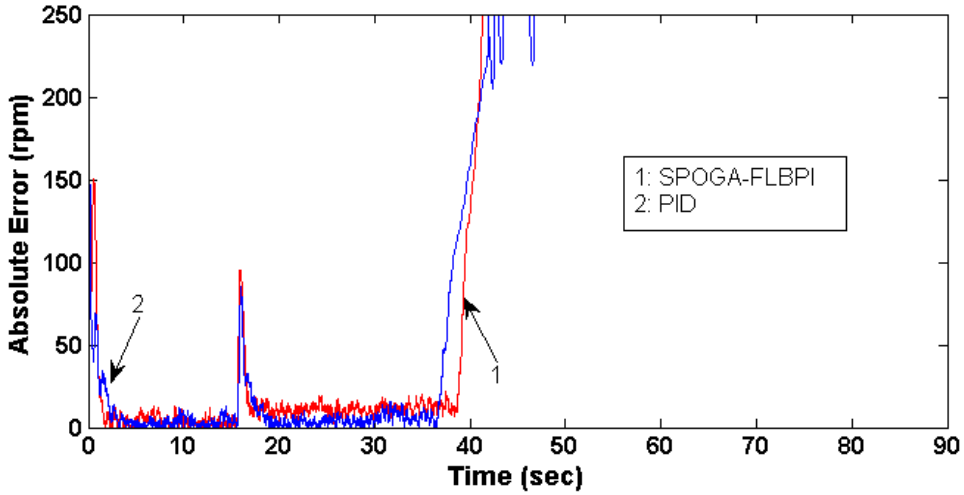


Fig. 5.25 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. PID for experiment 3a

Based on Fig. 5.26 and Fig. 5.27, SPOGA-FLBPI has the faster settling time than PID. This makes the absolute error of SPOGA-FLBPI smaller than PID.

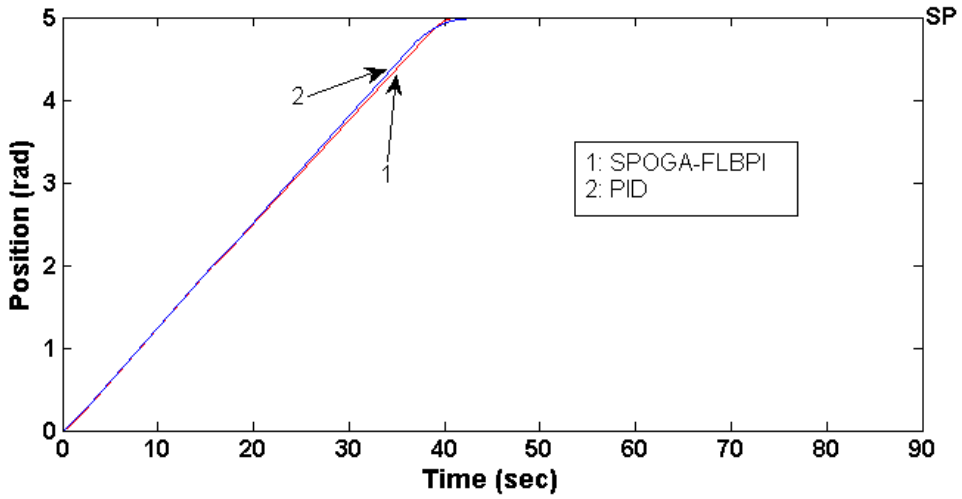


Fig. 5.26 Position control of DC servomotor using SPOGA-FLBPI vs. PID for experiment 3a

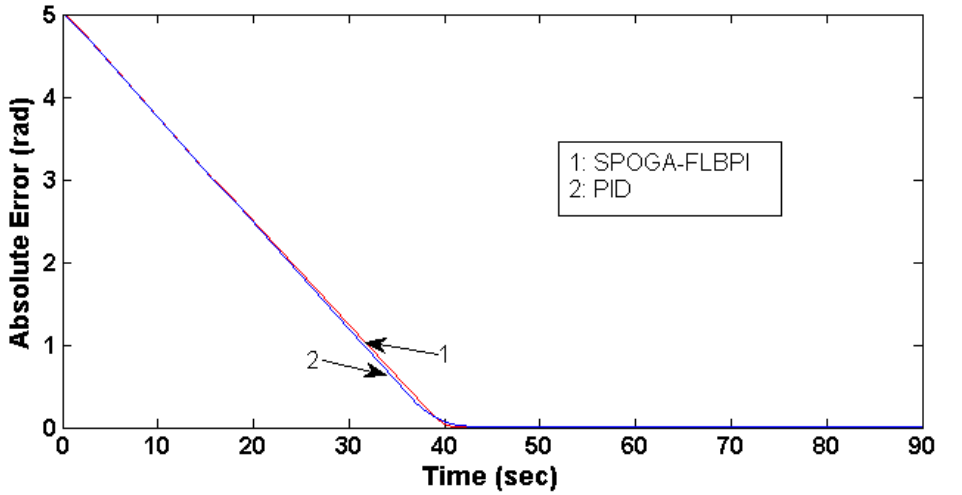


Fig. 5.27 Absolute error of position control of DC servomotor using SPOGA-FLBPI vs. PID for experiment 3a

In the experiment 3b, the best speed controller is PI for conventional controller and SPOGA-FLBPID for hybrid-fuzzy controller, and the best position controller is PID for conventional controller and SPOGA-FLBPI for hybrid-fuzzy controller. As a speed and position controller, PI is the best conventional controller and SPOGA-FLBPID is the best hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that hybrid-fuzzy controllers are better then conventional controllers in the experiment 3b. The graphical comparisons are shown in the Fig. 5.28 to Fig. 5.31.

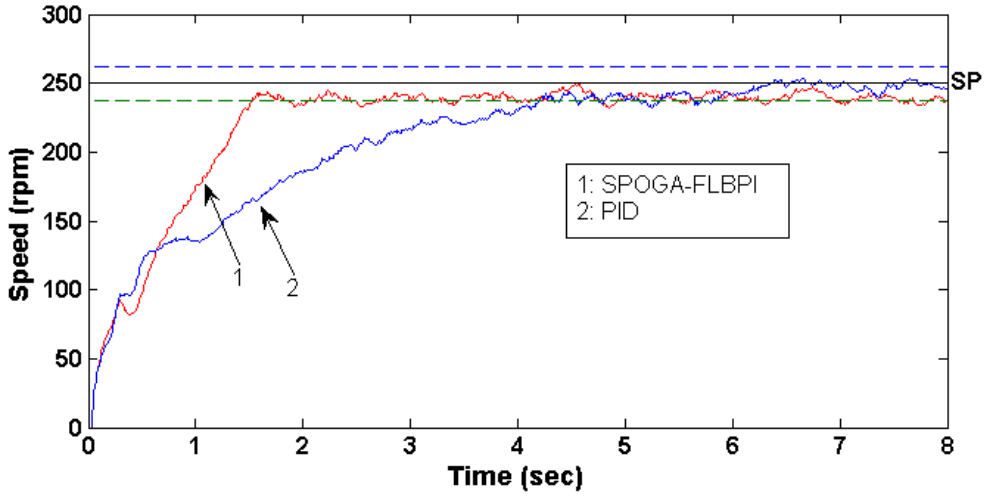


Fig. 5.28 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. PID for experiment 3b (see Fig. D.14)

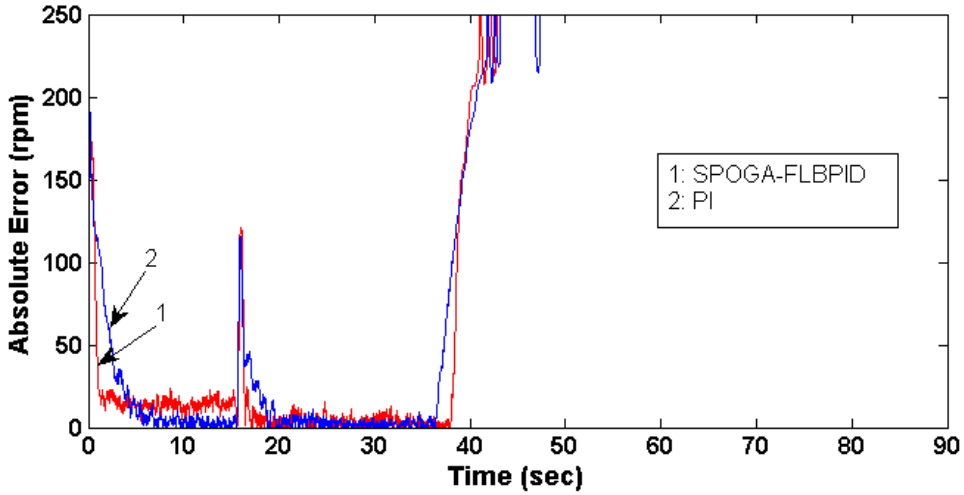


Fig. 5.29 Absolute error of speed control of DC servomotor using SPOGA-FLBPID vs. PI for experiment 3b

Based on Fig. 5.28 and Fig. 5.29, SPOGA-FLBPID has the faster settling time than PI. Consequently, the absolute error of SPOGA-FLBPID smaller than PI.

Based on Fig. 5.30 and Fig. 5.31, SPOGA-FLBPI has the faster settling time than PID. This makes the absolute error of SPOGA-FLBPI smaller than PID.

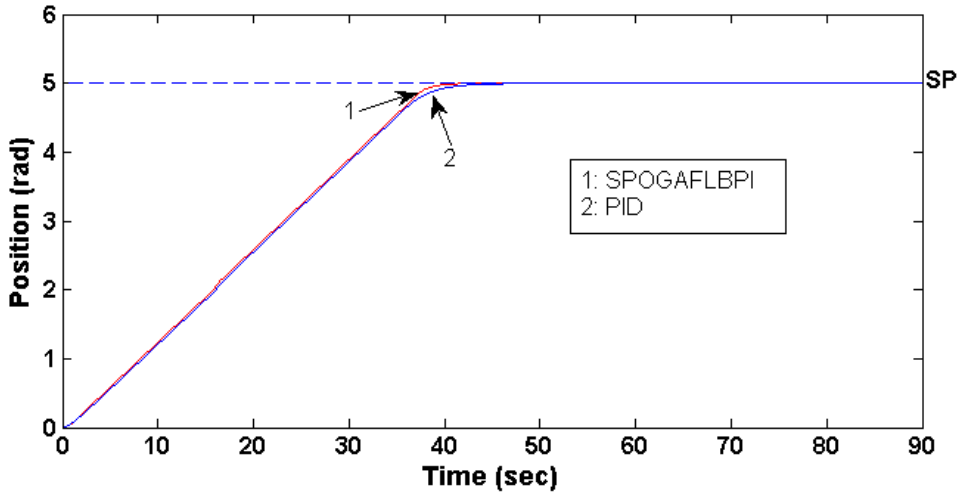


Fig. 5.30 Position control of DC servomotor using SPOGA-FLBPI vs. PID for experiment 3b

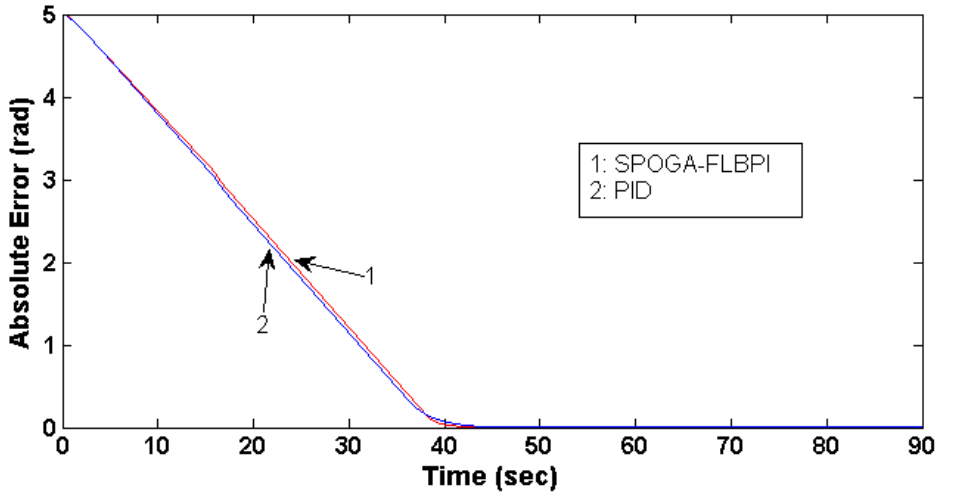


Fig. 5.31 Absolute error of position control of DC servomotor using SPOGA-FLBPI vs. PID for experiment 3b

In the experiment 4a, the best speed controller is PID for conventional controller and SPOGA-FLBPID for hybrid-fuzzy controller. As a speed and position controller, PID is the best conventional controller and SPOGA-FLBPID is the best hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that SPOGA optimized hybrid-fuzzy controllers are better then conventional controllers in the experiment 4a. The graphical comparisons are shown in the Fig. 5.32 and Fig. 5.33.

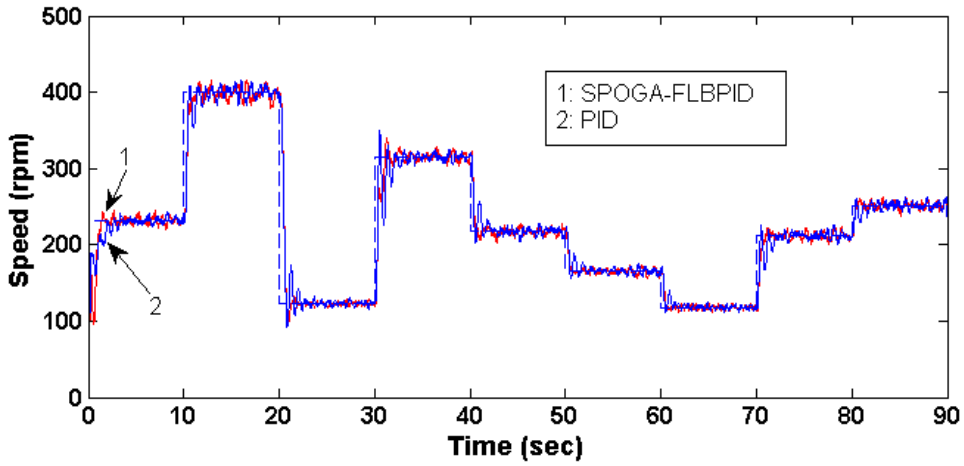


Fig. 5.32 Speed control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 4a

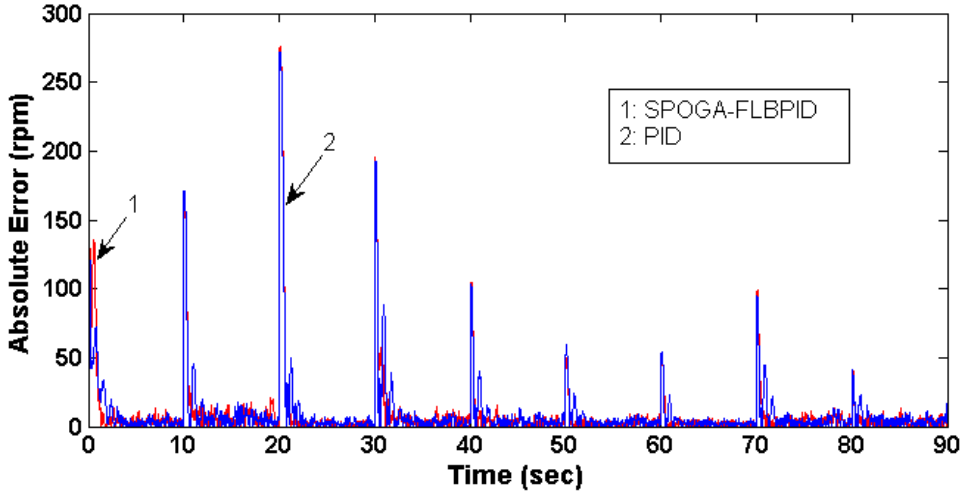


Fig. 5.33 Absolute error of speed control of DC servomotor using SPOGA-FLBPID vs. PID for experiment 4a

Based on Fig. 5.32 and Fig. 5.33, SPOGA-FLBPID has faster settling time than PID. This makes the absolute error of SPOGA-FLBPID smaller than PID.

In the experiment 4b, the best speed controller is PID for conventional controller and SPOGA-FLBPI for hybrid-fuzzy controller, and the best position controller is PID for conventional controller and SPOGA-FLBPID for hybrid-fuzzy controller. As a speed and position controller, PID is the best conventional controller and SPOGA-FLBPI is the best hybrid-fuzzy controller. Comparison on the best conventional to the best hybrid are shown in the table that SPOGA optimized hybrid-fuzzy controllers are

better than conventional controllers in the experiment 4b. The graphical comparisons are shown in the Fig. 5.34 to Fig. 5.37.

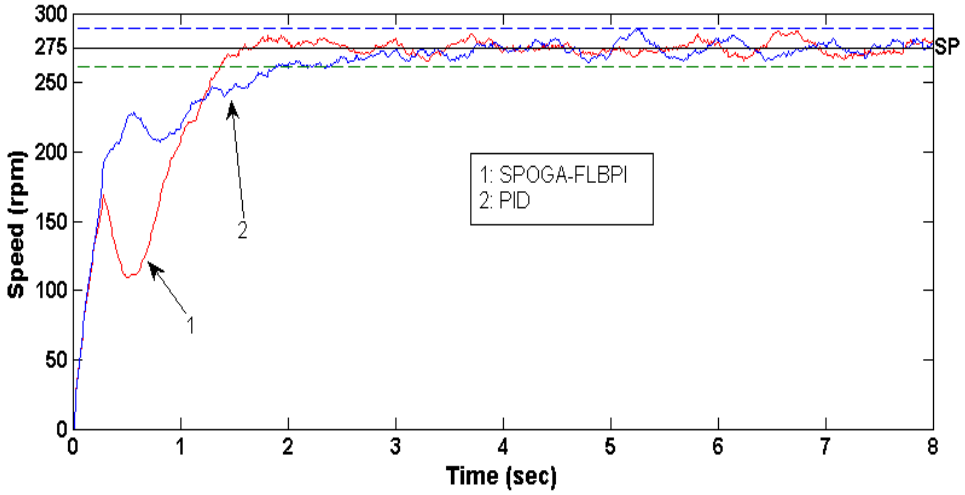


Fig. 5.34 Step response of speed control of DC servomotor using SPOGA-FLBPI vs. PID for experiment 4b (see Fig. D.15)

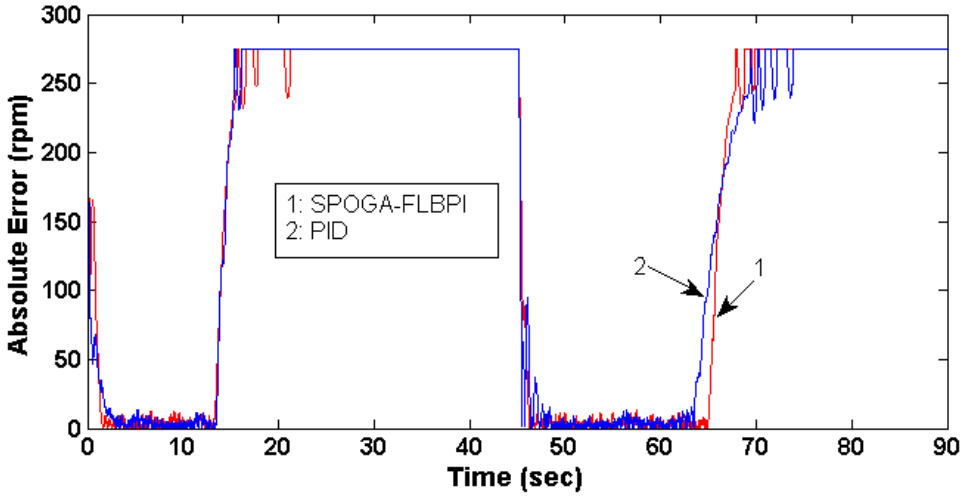


Fig. 5.35 Absolute error of speed control of DC servomotor using SPOGA-FLBPI vs. PID for experiment 4b

Based on in Fig. 5.34 and Fig. 5.35, SPOGA-FLBPI has the faster settling time than PID and on the second start, PID has overshoot. Consequently, the absolute error of SPOGA-FLBPI is smaller than PID.

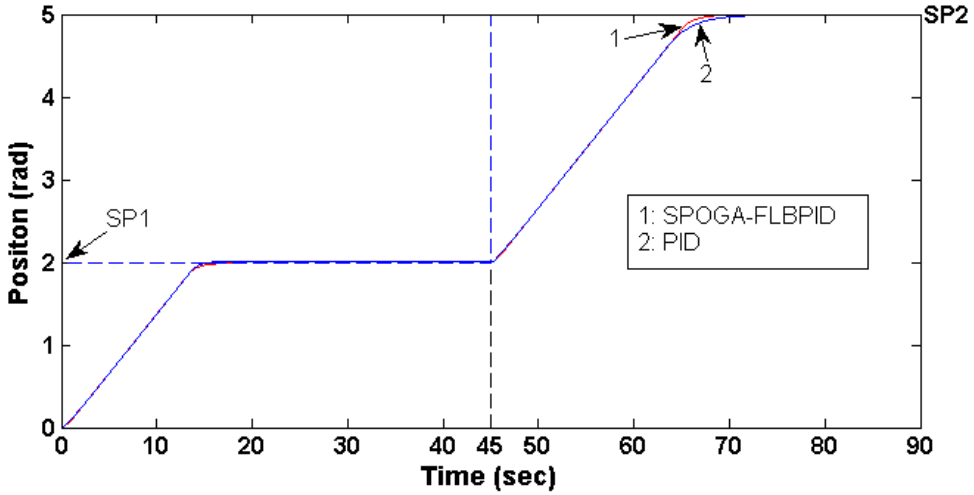


Fig. 5.36 Position control of DC servomotor using SPOGA-FLBPID vs. PID for simulation 4b

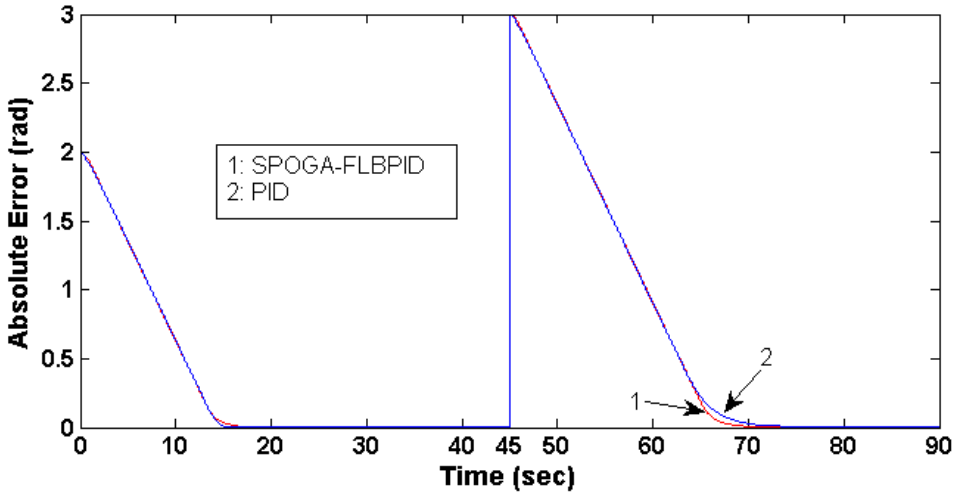


Fig. 5.37 Absolute error of position control of DC servomotor using SPOGA-FLBPID vs. PID for simulation 4b

Based on Fig. 5.36 and Fig. 5.37, SPOGA-FLBPID has faster settling time than PID on the second set point. This makes the absolute error of SPOGA-FLBPID smaller than PID.

5.7.2 Experiment Results Summary of Coventional, Fuzzy, and SPOGA Optimized Hybrid-Fuzzy Controllers

The speed and position control experiment of SPOGA optimized hybrid-fuzzy controller compared to conventional and fuzzy controller has been presented. SPOGA-FLBPID controller is the best speed controller as compared to SPOGA-FLBPI controller, SPOGA-FLIC, conventional controllers and fuzzy controller for the experiment of Type 2, 3b, and 4a. SPOGA-FLBPI is the best speed controller as compared to SPOGA-FLBPID controller, SPOGA-FLIC, conventional controllers and fuzzy controller for the experiment of Type 1a, 3a and 4b. Using Eq. (3 - 45), the best overall speed controller is SPOGA-FLBPI.

In both speed and position control, SPOGA-FLBPID controller is the best controller as compared to SPOGA-FLBPI controller, SPOGA-FLIC, conventional controllers and fuzzy controller for the experiment of Type 1a, 1b, 2, 3b, and 4a. SPOGA-FLBPI is the best controller as compared to SPOGA-FLIC, SPOGA-FLBPID controller, conventional controllers and fuzzy controller for the experiment of Type 3a and 4b. Using Eq. (3 - 45), PID controller is the best overall speed and position controller as compared conventional and fuzzy controller, and the best overall speed and position controller as compared to conventional, fuzzy, and hybrid controllers is SPOGA-FLBPI.

It is concluded that SPOGA optmized hybrid controllers have the better performance than conventional and fuzzy controllers. Therefore, it is proved that the SPOGA can be applied to optimize the parameters of hybrid controllers with the performance improvements.

5.8 Summary

This Chapter has presented the experiment of speed and position controllers using conventional controllers (PI and PID controllers), FLC, and hybrid-fuzzy controllers (FLBPI, FLBPID, and FLIC) based on the parameters as discussed in Chapter 4.

The experiment results show that hybrid-fuzzy controllers have the better performance than conventional controllers and FLC is shown to be having not as good performance as conventional and hybrid-fuzzy controllers. The results are basically the same as in the simulation experiments but the detailed performance and the performance results of each experiment type are not exactly the same since the tuning process was done in the *s*-modeled DC servomotor which is the approximation of real DC servomotor. The *s*-modeling of DC servomotor is very useful for running the SPOGA in the optimization process.

Experimentally, the appropriate p_m for optimization process of FLBPI, FLBPID, and FLIC is 0.01. The simulation results of GA and SPOGA show that to fulfil the minimum criteria, SPOGA can reduce 50% of the test runs and to fulfil the good criteria, SPOGA can reduce 9.46 % of the test runs. Therefore, SPOGA is selected as an algorithm to optimize the hybrid PID-fuzzy controllers using minimum criteria.

Experiment results of SPOGA optimized hybrid-fuzzy controllers show that SPOGA can optimize the parameters of FLBPI successfully, but in some experiments, the SPOGA-FLBPID is better than SPOGA-FLBPI.

Based on the simulation and experiment conducted in Chapter 4 and 5, the results reveal the following:

- The FLBPID developed based on FLBPI helps in improving the performance, in several cases.
- SPOGA has the influence to decrease the number of test runs for the optimization of some parameters.
- The population initialization in the GA using the principle of twisted ring counter would lead to more consistent outcomes from the genetic process.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The aim of this thesis has been to design the optimization algorithm to improve the controller's performance for a DC servomotor. The controller's performance is evaluated based on the following criterion: minimizing the overshoot, minimizing the settling time, IAE, ITAE and achieving a zero steady state error.

It discusses fundamental issues in the development of a simulation model for a DC servomotor, the simulation and experimental design and the optimization of the controller used for the servomotor. In particular it has been organised to answer questions such as what controlling method is needed to achieve the effective speed and position control, how to optimize the controller, and how to devise and evaluate some performance criterion for different operating conditions.

GA is effective in acquiring the optimal or near-optimal for solving optimization problem in control engineering, specifically achieving the best values for a predefined set of priorities defining a process model or a control law. However, GA shows indication of several limitations such as premature convergence and occurance of local maxima, an increase of population size without a corresponding increase in fitness, and undesirable searching speed.

Chapter 3 describes a new proposed algorithm to reduce the iteration number and the optimization process duration time. The proposed GA is called semi-parallel operation genetic algorithm (SPOGA), and the application of SPOGA to optimize the parameters of three hybrid-fuzzy controllers are presented. The three hybrid fuzzy-controllers optimized are: FLBPI, FLBPID, and FLIC. The discussion on the optimization process conducted in simulation model and the application of the

optimized parameters to the hardware experimental test rig are presented in Chapter 4 and 5, respectively.

In this study, the hybrid controllers were shown to be better controllers as compared to the conventional and fuzzy controllers for both speed and position control in both the simulation and in real-time implementation. The performance comparison based on simulations conducted for GA and SPOGA reveals that SPOGA can reduce the number of test runs (iterative number) and the duration time of the optimization process.

In real-time optimization, SPOGA has shown to be able to improve the performance of the hybrid controllers, namely FLBPI and FLBPID, in which during the earlier evaluations have proven to be better than the conventional (PI and PID) and fuzzy controllers. A closer observation shows that SPOGA-optimized FLBPI performs better than the non-SPOGA-optimized FLBPI. The main contributions of this work are:

- This work proposes a new GA optimization algorithm and demonstrates the steps taken in developing the semi-parallel operation GA (SPOGA) based on the hierarchical GA (HGA) and parallel GA (PGA) to decrease the number of test runs during the optimization process, for some parameters.
- This work demonstrates the step taken in developing the population initialization in the GA using the principle of twisted ring counter where the origin uses the random principle. The purpose is to make the outcomes of the genetic process more consistent.
- By adapting the hybrid control system viewpoint, FLBPI and FLBPID were developed and the controller parameters for the control of a servomotor were optimized using SPOGA. This paves way for the similar idea to be utilized in a more complex system.
- In SPOGA each parameter to be optimized has its own sub-chromosome and each sub-chromosome is processed separately in consequent with the others. In contrast with PGA which use multiprocessor, SPOGA work with one processor only.

6.2 Directions for Future Work

Future work should include :

- Improvement in the convergence speed:

  The improved optimized algorithm proposed in this work used fixed crossover and mutation rates. Applying variable crossover and mutation rates can speed up the convergence and restrain the premature convergence. The improved fitness value in GA is also relevant to SPOGA.

- Sensorless position control :

  The hardware test rig developed use a DC motor in which the speed was detected using speed sensor and the position was simulated using mathematical approach. Practically, position control is better to be sensorless as this will reduce cost and size and increase reliability of the overall system.

- SPOGA performance analysis :

  There exists several size and ratings of DC servomotors. The analysis of the SPOGA performance in optimizing the hybrid-fuzzy controller for the control of the different ratings of servomotors, would be very useful.

- Real-time implementation on actual robotic arm :

  The extension of the hardware test rig to have the DC motor shaft connected to a robotic arm with real gear ratio such that the arm's movement follows the desired position and speed, would still be required.

- Evaluation of GA optimization algorithm:

  The evaluation of other GA optimization algorithms that already exist, and comparison with SPOGA to provide better understanding of the performances, would need to be explored.

REFERENCES

[1] Meijuan Gao and Jingwen Tian, "Path Planning for Mobile Robot Based on Improved Simulated Annealing Artificial Neural Network," *Natural Computation, 2007. ICNC 2007. Third International Conference on*, 2007, pp. 8-12.

[2] S. Terentiev, E. Povernov, and E. Sypin, "The direct-current servomotor control system," *Electron Devices and Materials, 2004. Proceedings. 5th Annual. 2004 International Siberian Workshop on*, 2004, pp. 184-186.

[3] M. Shieh and T.S. Li, "Design and implementation of integrated fuzzy logic controller for a servomotor system," *Mechatronics*, vol. 8, Apr. 1998, pp. 217-240.

[4] L.A. Zadeh, "Fuzzy Logic," *Computer*, vol. 21 No. 24, Apr. 1988, pp. 83-93.

[5] Zhen-Yu Zhao, M. Tomizuka, and S. Isaka, "Fuzzy gain scheduling of PID controllers," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 23, 1993, pp. 1392-1398.

[6] E. Yeh and Y. Tsao, "A fuzzy preview control scheme of active suspension for rough road," *International Journal of Vehicle Design*, vol. 15, 1994, pp. 166-180.

[7] S. He, S. Tan, F. Xu, and P. Wang, "Fuzzy self-tuning of PID controllers," *Fuzzy Sets Syst.*, vol. 56, 1993, pp. 37-46.

[8] X. Gao and Z. Feng, "Design study of an adaptive Fuzzy-PD controller for pneumatic servo system," *Control Engineering Practice*, vol. 13, Jan. 2005, pp. 55-65.

[9] S. Jee and Y. Koren, “Adaptive fuzzy logic controller for feed drives of a CNC machine tool,” *Mechatronics*, vol. 14, Apr. 2004, pp. 299-326.

[10] O. Castillo, G. Huesca, and F. Valdez, “Evolutionary computing for optimizing type-2 fuzzy systems in intelligent control of non-linear dynamic plants,” *Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American*, 2005, pp. 247-251.

[11] H. Huang, M. Pasquier, and C. Quek, “Financial Market Trading System With a Hierarchical Coevolutionary Fuzzy Predictive Model,” *Evolutionary Computation, IEEE Transactions on*, vol. 13, 2009, pp. 56-70.

[12] Jun Zhang, Henry Shu-Hung Chung, and Wai-Lun Lo, “Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms,” *Evolutionary Computation, IEEE Transactions on*, vol. 11, 2007, pp. 326-335.

[13] H. Lau, T. Chan, and W. Tsui, “Item-Location Assignment Using Fuzzy Logic Guided Genetic Algorithms,” *Evolutionary Computation, IEEE Transactions on*, vol. 12, 2008, pp. 765-780.

[14] E. Mininno, F. Cupertino, and D. Naso, “Real-Valued Compact Genetic Algorithms for Embedded Microcontroller Optimization,” *Evolutionary Computation, IEEE Transactions on*, vol. 12, 2008, pp. 203-219.

[15] P. Whigham and G. Dick, “Implicitly Controlling Bloat in Genetic Programming,” *IEEE Transaction on Evolutionary Computation*, vol. 14, Apr. 2010, pp. 173-190.

[16] W. Wang and Y. Li, “Evolutionary Learning of BMF Fuzzy-Neural Networks Using a Reduced-Form Genetic Algorithm,” *IEEE Transactions on Systems, Man, and Cybernatics*, vol. 33, Dec. 2003, pp. 966-976.

[17] Gui Yang, Yujun Lu, Ren-wang Li, and Jin Han, “Adaptive genetic algorithms for the Job-Shop Scheduling Problems,” *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, 2008, pp. 4501-4505.

[18] P. Day and A. Nandi, "Binary String Fitness Characterization and Comparative Partner Selection in Genetic Programming," *Evolutionary Computation, IEEE Transactions on,* vol. 12, 2008, pp. 724-735.

[19] K. Duzinkiewics, M.A. Brdys, W. Kurek, and R. Piotrowski, "Genetic Hybrid Predictive Controller for Optimized Dissolved-Oxygen Tracking at Lower Control Level," *IEEE Transaction on Control System Technology,* vol. 17, Sep. 2009, pp. 1183-1192.

[20] B. Karanayil, F. Rahman, and C. Gratham, "Online Stator and Rotor Resistance Estimation Scheme Using Artificial Neural Networks for Vector Controlled Speed Sensorless Induction Motor Drive," *IEEE Transaction on Industrial Electronics,* vol. 54, Feb. 2007, pp. 167-176.

[21] Yao Jinyong, Su Haibo, and Li Xiaogang, "Using Simulated Annealing Embedded Modified Gauss-Newton Algorithm to identify parameters of nonlinear degradation model," *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, 2010, pp. V10-653-V10-656.

[22] K.M. Takami and J. Mahmoudi, "Identification of a Best Thermal Formula and Model for Oil and Winding of Power Transformers Using Prediction Methods," *The 48th Scandinavian Conference on Simulation and Modeling (SIMS 2007),* Sweden: 2007, pp. 182-188.

[23] A. Cavallo, G. De Maria, C. Natale, and S. Pirozzi, "Gray-Box Identification of Continuous-Time Models of Flexible Structures," *IEEE Transaction on Control System Technology,* vol. 15, Sep. 2007, pp. 967-981.

[24] D. Pereira and J. Pinto, "Genetic algorithm based system identification and PID tuning for optimum adaptive control," *Advanced Intelligent Mechatronics. Proceedings, 2005 IEEE/ASME International Conference on*, 2005, pp. 801-806.

[25] H. Pongpairoj and F. Pourboghrat, "Real-Time Optimal Control of Flexible Structures Using Subspace Techniques," *IEEE Transaction on Control Systems Technology,* vol. 14, Nov. 2006, pp. 1021-1033.

[26] R. Battacharya and G.J. Balas, "Control in Computationally Constrained Environments," *IEEE Transaction on Control System Technology*, vol. 17, May. 2009, pp. 589-599.

[27] T. Yamamoto, K. Takao, and T. Yamada, "Design of a Data-Driven PID Controller," *IEEE Transaction on Control System Technology*, vol. 17, Jan. 2009, pp. 29-39.

[28] B. Lacevic, J. Velagic, and N. Osmic, "Design of Fuzzy Logic Based Mobile Robot Position Controller Using Genetic Algorithm," *International Conference on Advanced Intelligent Mechatronics, IEEE/ASME 2007*, 2007, pp. 1-6.

[29] L. Zadeh, "Is there a need for fuzzy logic?," Fuzzy Information Processing Society, 2008. NAFIPS 2008. Annual Meeting of the North American, 2008, pp. 1-3.

[30] E. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," *Electrical Engineers, Proceedings of the Institution of*, vol. 121, 1974, pp. 1585-1588.

[31] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases," *Advances in Fuzzy Systems-Applications and Theory*, PO Box 128, Farrer Road, Singapore 912805: World Scientific Publishing Co. Pte. Ltd., .

[32] Shuqing Wang, Zipeng Zhang, and Liqin Xue, "Knowledge Acquisition of Fuzzy Control System Based on Improved Genetic Algorithm and Neural Networks," *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on*, 2008, pp. 95-99.

[33] M.I. Solihin, Wahyudi, and A. Legowo, "Fuzzy-tuned PID Anti-swing Control of Automatic Gantry Crane," *Journal of Vibration and Control*, vol. 16, 2010, pp. 127-145.

[34] K. Saridakis and A. Dentsoras, “Integration of fuzzy logic, genetic algorithms and neural networks in collaborative parametric design,” *ScienceDirect, Advanced Engineering Informatics*, vol. 20, 2006, pp. 379-399.

[35] T. Park and K.R. Ryu, “A Dual-Population Genetic Algorithm for Adaptive Diversity Control,” *Evolutionary Computation, IEEE Transactions on*, vol. PP, 2010, p. 1.

[36] K. Man, K. Tang, and S. Kwong, “Genetic algorithms: concepts and applications [in engineering design],” *Industrial Electronics, IEEE Transactions on*, vol. 43, 1996, pp. 519-534.

[37] P. Balasubramaniam and A.V.A. Kumar, “Solution of matrix Riccati differential equation for nonlinear singular system using genetic programming,” *Springer, Genet Program Evolvable Mach*, vol. 10, 2009, pp. 71-89.

[38] M.D. Schmidt and H. Lipson, “Coevolution of Fitness Predictors,” *IEEE Transaction on Evolutionary Computation*, vol. 12, Dec. 2008, pp. 736-749.

[39] Z. Xiu and G. Ren, “Optimization Design of TS-PID Fuzzy Controllers Based on Genetic Algorithms,” *5th World Congress on Intelligent Control and Automation*, Hangzhou, P.R. China: 2004, pp. 2476-2480.

[40] E. Bagheri and H. Deldari, “Dejong Function Optimization by means of a Parallel Approach to Fuzzified Genetic Algorithm,” *Computers and Communications, 2006. ISCC '06. Proceedings. 11th IEEE Symposium on*, 2006, pp. 675-680.

[41] G. Di Fatta, G. Lo Re, and A. Urso, “Parallel Genetic Algorithms for the Tuning of a Fuzzy AQM Controller,” *Computational Science and Its Applications — ICCSA 2003*, Springer Berlin / Heidelberg, 2003, p. 965.

[42] J. Lis, “Parallel genetic algorithm with the dynamic control parameter,” *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, 1996, pp. 324-329.

[43] K. Worapradya and S. Pratishthananda, “Fuzzy supervisory PI controller using hierarchical genetic algorithms,” *Control Conference, 2004. 5th Asian*, 2004, pp. 1523-1528 Vol.3.

[44] I. Bousserhane, A. Hazzab, M. Rahli, M. Kamli, and B. Mazari, “Adaptive PI Controller using Fuzzy System Optimized by Genetic Algorithm for Induction Motor Control,” *International Power Electronics Congress, 10th IEEE*, 2006, pp. 1-8.

[45] M. Mannan, T. Murata, J. Tamura, and T. Tsuchiya, “Fuzzy-logic-based self-tuning PI controller for speed control of indirect field-oriented induction motor drive,” *SICE 2004 Annual Conference*, 2004, pp. 466-470 vol. 1.

[46] T. Kissel, *Motor Control Technology for Industrial Maintenance*, New Jersey 07458: Prentice-Hall, Inc., Upper Saddle River, 2002.

[47] A. Rubaai, M.J. Castro-Sitiriche, M. Garuba, and L. Burge, “Implementation of Artificial Neural Network-Based Tracking Controller for High-Performance Stepper Motor Drives,” *IEEE Transaction on Industrial Electronics*, vol. 54, Feb. 2007, pp. 218-227.

[48] M. Bodson, J.S. Sato, and S.R. Silver, “Spontaneous Speed Reversals in Stepper Motors,” *IEEE Transaction on Control System Technology*, vol. 14, Mar. 2006, pp. 369-373.

[49] C. Erdal, “A sensitivity measure for armature-controlled DC servomotors and calculating optimum parameter tolerances,” *Electrotechnical Conference, 1996. MELECON '96., 8th Mediterranean*, 1996, pp. 342-345 vol.1.

[50] D. Karagiannis, E. Mendes, A. Astolfi, and R. Ortega, “An Experimental Comparison of Several PWM Controllers for a Single-Phase AC–DC Converter,” *IEEE Transaction on Control System Technology*, vol. 11, Nov. 2003, pp. 940-947.

[51] C. Hsieh and J. Chou, "Design of Optimal PID Controllers for PWM Feedback Systems With Bilinear Plants," *IEEE Transaction on Control System Technology*, vol. 15, Nov. 2007, pp. 1075-1079.

[52] Anonim, "Control Tutorial for Matlab: DC Motor Speed Modeling," 1997.

[53] M. Jamshidi and M. Zavarei, *Linear Control Systems : A Computer-Aided Approach.*, Great Britain: Wheaton & Co.Ltd., 1986.

[54] N.S. Nise, *Control System Engineering*, Hoboken, NJ: John Wiley & Sons Ltd., 2004.

[55] J. Jacob, *Industrial Control Electronics*, Englewood Cliffs, NJ.: Prentice-Hall International Editions, 1989.

[56] G. Olsson and G. Piani, *Computer Systems for Automation and Control*, UK.: Prentice-Hall, 1992.

[57] C. Johnson, *Process Control Instrumentation Technology*, New Jersey: Prentice Hall International, Inc. Englewood Cliffs., 1993.

[58] I.R. Petersen, "Actuator Saturation and Integrator Windup," 2002.

[59] L. Wang, *A Course in Fuzzy System and Control*, Upper Saddle River, New Jersey 07458: Prentice-Hall, Inc, A Division of Simon and Schuster, 1997.

[60] T.J. Ross, *Fuzzy Logic with Engineering Applications*, West Sussex, England: John Wiley & Sons Ltd., 2004.

[61] D. Pham and D. Karaboga, Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks, Great Britain: Springer-Verlag, 1998.

[62] H. Shan, S. Li, D. Gong, and P. Lou, "Genetic simulated annealing algorithm-based assembly sequence planning," *Technology and Innovation Conference, 2006. ITIC 2006. International*, 2006, pp. 1573-1579.

[63] F. Rodríguez-Díaz, C. García-Martínez, and M. Lozano, "A GA-based multiple simulated annealing," *Evolutionary Computation (CEC), 2010 IEEE Congress on*, 2010, pp. 1-7.

[64] Kunlei Lian, Chaoyong Zhang, Xinyu Li, and Liang Gao, "An Effective Hybrid Genetic Simulated Annealing Algorithm for Process Planning Problem," *Natural Computation, 2009. ICNC '09. Fifth International Conference on*, 2009, pp. 367-373.

[65] Zhufang Wang and Donghong Cui, "A Hybrid Algorithm Based on Genetic Algorithm and Simulated Annealing for Solving Portfolio Problem," *Business Intelligence and Financial Engineering, 2009. BIFE '09. International Conference on*, 2009, pp. 106-109.

[66] Enlu Zhou and Xi Chen, "A new population-based simulated annealing algorithm," *Winter Simulation Conference (WSC), Proceedings of the 2010*, 2010, pp. 1211-1222.

[67] Li-li Dong, Ni Li, and Guang-hong Gong, "Adaptive & parallel simulated annealing genetic algorithm based on cloud model," *Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on*, 2010, pp. 7-11.

[68] Pengjun Wang, Hui Li, and Zhenhai Wang, "MPRM expressions minimization based on simulated annealing genetic algorithm," *Intelligent Systems and Knowledge Engineering (ISKE), 2010 International Conference on*, 2010, pp. 261-265.

[69] Ming-Hao Hung, Li-Sun Shu, Shinn-Jang Ho, Shiow-Fen Hwang, and Shinn-Ying Ho, "A Novel Intelligent Multiobjective Simulated Annealing Algorithm for Designing Robust PID Controllers," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 38, 2008, pp. 319-330.

[70] Sun Fengjie, Wang He, and Fan Jieqing, “2D Otsu Segmentation Algorithm Based on Simulated Annealing Genetic Algorithm for Iced-Cable Images,” *Information Technology and Applications, 2009. IFITA '09. International Forum on*, 2009, pp. 600-602.

[71] Shan Hong-Bo and Li Shuxia, “The Comparison Between Genetic Simulated Annealing Algorithm and Ant Colony Optimization Algorithm for ASP,” *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, 2008, pp. 1-6.

[72] Qi Ji-Yang, “Application of improved simulated annealing algorithm in facility layout design,” *Control Conference (CCC), 2010 29th Chinese*, 2010, pp. 5224-5227.

[73] Nguyen Thanh Trung and Duong Tuan Anh, “Comparing Three Improved Variants of Simulated Annealing for Optimizing Dorm Room Assignments,” *Computing and Communication Technologies, 2009. RIVF '09. International Conference on*, 2009, pp. 1-5.

[74] Zhonghai Lu, Lei Xia, and A. Jantsch, “Cluster-based Simulated Annealing for Mapping Cores onto 2D Mesh Networks on Chip,” *Design and Diagnostics of Electronic Circuits and Systems, 2008. DDECS 2008. 11th IEEE Workshop on*, 2008, pp. 1-6.

[75] Huang Dong and Qiao Jian-Ping, “Hybrid of ant colony algorithm and simulated annealing algorithm and its application to the slope stability analysis,” *Natural Computation (ICNC), 2010 Sixth International Conference on*, 2010, pp. 3329-3333.

[76] Meijuan Gao and Jingwen Tian, “Network Intrusion Detection Method Based on Improved Simulated Annealing Neural Network,” *Measuring Technology and Mechatronics Automation, 2009. ICMTMA '09. International Conference on*, 2009, pp. 261-264.

[77] Kai Bai and Jing Xiong, "A Method of Improved BP Neural Algorithm Based on Simulated Annealing Algorithm," *Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on*, 2009, pp. 765-768.

[78] Anonim, "Evoulutionary Algorithm," *Wikipedia*, 2010.

[79] H. Zhuang and S. Wongsoontorn, Knowledge-based Tuning I: Design and Tuning of Fuzzy Control Sufaces with Bezier Function, London: Springer-Verlag, 2006.

[80] K. Man, S. Tang, and W. Halang, *Genetic Algorithms for Control and Signal Processing*, Britain: Springer-Verlag London Limited, 1997.

[81] Z. Michalewicz, Genetic Algorithm + Data Structures = Evolution Program, Springer-Verlag, 1994.

[82] M. Bodur, A. Acan, and T. Akyol, "Fuzzy System Modeling with the Genetic and Differential Evolutionary Optimization," Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on, 2005, pp. 432-438.

[83] J. Gomez, R. Poveda, and E. Leon, "Grisland: a parallel genetic algorithm for finding near optimal solutions to the traveling salesman problem," *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, Montreal, Quebec, Canada: ACM, 2009, pp. 2035-2040.

[84] T.E. Marlin, Process Control: Designing Processes and Control Systems for Dynamic Performance, Singapore: McGraw-Hill Book Companies, Inc., 2000.

[85] M. Montanari, S.M. Peresada, C. Rossi, and A. Tilli, "Speed Sensorless Control of Induction Motors Based on a Reduced-Order Adaptive Observer," *IEEE Transaction on Control System Technology*, vol. 15 No. 6, Nov. 2007, pp. 1049-1064.

[86] Anonim, “Op-Amp Differential Amplifier.svg,” 2009.

[87] Anonim, “USB-1208FS: USB-based Analog and Digital I/O Module User's Guide,” Jul. 2007.

[88] N. Thomas and P. Poongodi, “Position Control of DC Motor Using Genetic Algorithm Based PID Controller,” *Proceedings of the World Congress on Engineering 2009 Vol II, WCE 2009*, London, UK.: 2009.

[89] K. Chua, W. Hew, C. Foo, and K. Lai, “A Comparative Analysis of PI, Fuzzy Logic and ANFIS Speed Control of Permanent Magnet Synchronous Motor,” *International Conference on Robotics, Vision, Signal Processing & Power Applications (RoViSP 09)*, Awana Porto Malai, Langkawi, Kedah, Malaysia: 2009.