COMPUTATIONAL INTELLIGENCE IN CONTROL APPLICATIONS

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree
of

MASTER OF TECHNOLOGY
in
ELECTRONICS AND COMPUTER ENGINEERING
(With Specialization in Control & Guidance)

By

TUSHAR JAIN

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2009

CANDIDATE'S DECLARATION

I here by declare that the work presented in this dissertation entitled **"Computational Intelligence in Control Applications "** being submitted in partial fulfillment of the requirements for the award of the degree of **Master of Technology** with specialization in **Control & Guidance,** in the Department of Electronics & Computer Engineering, **Indian Institute of Technology, Roorkee**, under the guidance of **Dr. M.J. Nigam,** Department of Electronics & Computer Engineering, Indian Institute of Technology Roorkee, Roorkee.

Date: 3 June, 2009
Place: Roorkee

[illegible]
(Tushar Jain)

CERTIFICATE

This is to certify that the above statement made by the candidate is true to the best of my knowledge and belief.

[illegible]

Date: 3.06.09
Place: Roorkee

(Dr. M.J. Nigam)
Associate Professor
Department of Electronics & Computer Engineering,
Indian Institute of Technology Roorkee,
Roorkee – 247667, India.

ACKNOWLEDGEMENT

I express my foremost and deepest gratitude to **Dr. M.J. Nigam**, Department of Electronics & Computer Engineering, Indian Institute of Technology Roorkee, Roorkee for his valuable guidance, support and motivation throughout this work. The valuable hours of discussion and suggestions that I had with him have undoubtedly helped in supplementing my thoughts in the right direction for attaining the desired objective. I consider myself extremely fortunate for having got the opportunity to learn and work under his able supervision over the entire period of my association with him.

My sincere thanks to all faculty members of Control & Guidance for their constant encouragement, caring words, constructive criticism and suggestions towards the successful completion of this work. My sincere thanks to laboratory staff to access the computers and other resources at will for completion of this work.

Last but not the least, I'm highly indebted to my parents and family members, whose sincere prayers, best wishes, moral support and encouragement have a constant source of assurance, guidance, strength, and inspiration to me.

(Tushar Jain)

ABSTRACT

Sensitivity and Robustness is the primary issue while designing the controller for non-linear systems. One of the performance objectives for controller design is to keep the error between the controlled output and the set-point as small as possible. The control of many non-linear, inherently unstable systems using conventional methods is both difficult to design and marginally satisfactory in implementation. The introduction of optimization techniques in control engineering that makes use of evolutionary computation and an implicit imprecision is successful in counteracting these limitations. The field of computational intelligence has incorporated to such systems with an objective to achieve higher optimality and satisfactory performance

The main aim of this work is to design and implement biologically inspired optimization algorithms based control system. The performance of Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Bacterial Foraging (BF) Optimization has been improved using hybridization. The novel algorithms developed are hybrid BF-PSO or adaptive BF, Genetically Bacterial Swarm Optimization (GBSO). The algorithms are first tested on basic mathematical functions and then implemented on various control engineering problems: Inverted Pendulum system, Ball and Beam System and trajectory tracking in robot manipulators. The proposed algorithms also played a vital role in eliminating the curse of dimensionality to an acceptable value.

The knowledge base of a Fuzzy Logic Controller (FLC) encapsulates expert knowledge and consists of the data base (membership functions) and rule-base of the controller. Optimization of both of these knowledge base components is critical to the performance of the controller and has traditionally been achieved through a process of trial and error and certain classical optimization techniques. In this work, the superiority of novel hybrid algorithms is demonstrated through offline tuning of rule base and scaling factor over the experts' design of rule base and design using classical optimization techniques.

Light weight flexible arms will most likely constitute the next generation robots due to their large payload carrying capacities at high speeds and less power demand. Control problem of robots with flexible members is more complex compared to rigid robots due to vibrations during the motion. Here the trajectory control of two link rigid-flexible manipulator is presented in two phases. In first phase, only the scaling factors of hybrid fuzzy precompensated PD control are optimized using the optimization techniques. While in second phase, the rule base of fuzzy precompensator is optimized using characteristic parameters keeping the PD controller parameter constant.

Declaration........ i

Acknowledgement........ ii

Abstract........ iii

Content........ v

1. **Introduction**........ 1

1.1. Evolutionary Computation........ 1

1.2. Swarm Intelligence........ 2

1.3. Fuzzy Systems........ 3

1.4. Problem Statement........ 3

1.5. Literature Review........ 4

2. **Intelligent Computational Techniques**........ 7

2.1. Genetic Algorithm........ 7

2.1.1. Basic Construction of GA's........ 7

2.2. Particle Swarm Optimization........ 11

2.3. Bacterial Foraging........ 13

2.3.1. Components of bacterial foraging........ 14

3. **Hybridization of Intelligent Computational Techniques**........ 20

3.1. Hybrid Bacterial Foraging........ 20

3.1.1. Hybrid Chemotaxis........ 21

3.1.2. Hybrid Reproduction........ 23

3.2. Hybrid Bacterial Foraging-Particle Swarm Optimization........ 23

3.3. Genetically-Bacterial Swarm Optimization........ 26

3.3.1. Ranking and Selection in GBSO........ 27

3.3.2. Crossover operation in GBSO........ 27

3.3.3. Mutation operation in GBSO........ 27

3.3.4. Particle Swarm Optimization........ 27

3.3.5. The GBSO algorithm........ 28

4. **Fuzzy Logic and Fuzzy Control**........ 33

4.1. Fuzzy Logic Controller........ 33

4.1.1. Linguistic Variable, Rule Bases and Membership Functions..33
4.1.2. Fuzzification.......... 35
4.1.3. Inference.......... 35
4.1.4. Rule Base.......... 37
4.1.5. Defuzzification.......... 38
4.2. Designing a Fuzzy Logic Controller.......... 38
4.2.1. Assumptions and Constraints.......... 38
4.2.2. Spacing Parameter.......... 39
4.2.3. Designing the Rule-Base.......... 40
4.3. Hybrid Fuzzy Precompensated PD Controller Design.......... 42
4.3.1. PD Control.......... 42
4.3.2. Fuzzy Precompensation.......... 43
5. **Applications of Intelligent Computational Techniques**.......... 45
5.1. Inverted Pendulum.......... 45
5.2. Ball and Beam System.......... 46
5.2.1. Mechanical Model of Ball & Beam.......... 47
5.3. Two Link Rigid-Flexible Manipulator Dynamics.......... 49
6. **Results and Their Interpretation**.......... 54
6.1. Experimental results using test functions.......... 54
6.2. Inverted Pendulum.......... 59
6.3. Ball and Beam system.......... 62
6.4. Two link rigid-flexible robot manipulator.......... 65
7. **Conclusion and Future Scope**.......... 78
References.......... 79
Research papers published by the author.......... 83

1. INTRODUCTION

A major thrust in the algorithmic development is the design of algorithmic models to solve increasingly complex problems. Enormous successes have been achieved through the modeling of biological and natural intelligence, resulting in so-called "intelligent systems" [35]. These intelligent algorithms include artificial neural networks, evolutionary computation, swarm intelligence, artificial immune systems, and fuzzy systems. Together with logic, deductive reasoning, expert systems, case-based reasoning and symbolic machine learning systems, these intelligent algorithms form part of the field of Computational Intelligence (CI).

1.1. Evolutionary Computation

Evolutionary Computation (EC) has its objective to mimic processes from natural evolution, where the main concept is survival of the fittest: the weak must die. In natural evolution, survival is achieved through reproduction. Offspring, reproduced from two parents (sometimes more than two), contain genetic material of both (or all) parents – hopefully the best characteristics of each parent. Those individuals that inherit bad characteristics are weak and lose the battle to survive. This is nicely illustrated in some bird species where one hatchling manages to get more food, gets stronger, and at the end kicks out all its siblings from the nest to die.

Evolutionary algorithms use a population of individuals, where an individual is referred to as a chromosome. A chromosome defines the characteristics of individuals in the population. Each characteristic is referred to as a gene. The value of a gene is referred to as an allele. For each generation, individuals compete to reproduce offspring. Those individuals with the best survival capabilities have the best chance to reproduce. Offspring are generated by combining parts of the parents, a process referred to as crossover. Each individual in the population can also undergo mutation which alters some of the allele of the chromosome. The survival strength of an individual is measured using a fitness function which reflects the objective and constraints of the problem to be solved. After each generation, individuals may undergo culling, or individuals may survive to the next generation (referred to as elitism). Additionally, behavioral characteristics (as encapsulated in phenotypes) can be used to influence the evolutionary process in two ways: phenotypes may influence genetic changes, and/or behavioral characteristics evolve separately.

Evolutionary computation has been used successfully in real-world applications, for example, data mining, combinatorial optimization, fault diagnosis, classification, clustering, scheduling, and time series approximation.

1.2. Swarm Intelligence

Swarm intelligence (SI) originated from the study of colonies, or swarms of social organisms. Studies of the social behavior of organisms (individuals) in swarms prompted the design of very efficient optimization and clustering algorithms. For example, simulation studies of the graceful, but unpredictable, choreography of bird flocks led to the design of the particle swarm optimization algorithm, and studies of the foraging behavior of *E.coli* bacteria resulted in bacterial foraging optimization algorithms.

Particle swarm optimization (PSO) is a stochastic optimization approach, modeled on the social behavior of bird flocks. PSO is a population-based search procedure where the individuals, referred to as particles, are grouped into a swarm. Each particle in the swarm represents a candidate solution to the optimization problem. In a PSO system, each particle is "flown" through the multidimensional search space, adjusting its position in search space according to its own experience and that of neighboring particles. A particle therefore makes use of the best position encountered by it and the best position of its neighbors to position itself toward an optimum solution. The effect is that particles "fly" toward an optimum, while still searching a wide area around the current best solution. The performance of each particle (i.e. the "closeness" of a particle to the global minimum) is measured according to a predefined fitness function which is related to the problem being solved. Applications of PSO include function approximation, clustering, optimization of mechanical structures, and solving systems of equations.

Studies of bacterial foraging have contributed in abundance to the set of intelligent algorithms. The modeling of running, tumbling, and swimming by bacteria in their search for the shortest paths to food sources resulted in the development of shortest path optimization algorithms. Other applications of bacterial foraging optimization include routing optimization in telecommunication networks, graph coloring, scheduling and solving the quadratic assignment problem.

1.3. Fuzzy Systems

Traditional set theory requires elements to be either part of a set or not. Similarly, binary-valued logic requires the value of parameters to be either 0 or 1, with similar constraints on the outcome of an inference process. Human reasoning is, however, almost always not this exact. The observations and reasoning usually include a measure of uncertainty.

Fuzzy sets and fuzzy logic allow what is referred to as approximate reasoning. With fuzzy sets, an element belongs to a set to a certain degree of certainty. Fuzzy logic allows reasoning with these uncertain facts to infer new facts. In a sense, fuzzy sets and logic allow the modeling of common sense.

The uncertainty in fuzzy systems is referred to as nonstatistical uncertainty, and should not be confused with statistical uncertainty. Statistical uncertainty is based on the laws of probability, whereas nonstatistical uncertainty is based on vagueness, imprecision and/or ambiguity. Statistical uncertainty is resolved through observations. For example, when a coin is tossed we are certain what the outcome is, while before tossing the coin, we know that the probability of each outcome is 50%. Nonstatistical uncertainty, or fuzziness, is an inherent property of a system and cannot be altered or resolved by observations.

Fuzzy systems have been applied successfully to control systems, gear transmission and braking systems in vehicles, controlling lifts, home appliances, controlling traffic signals, and many others.

1.4. Problem Statement

In light of the discussion, the prime objectives of the research work will focus on the developing simple but commercially attractive and viable methods for the purpose of designing optimal control systems. The objectives of the thesis can be summarized as follows:

1. Develop particular Evolutionary Algorithms and Swarm Intelligence Techniques that can enable automation to provide optimal control system.
2. Develop a novel approach using hybridization of basic optimization techniques by which their performance can be enhanced in terms of convergence of fitness function in less generation.
3. Implement the developed hybrid algorithms over various control engineering problems, for example, Inverted Pendulum, two link rigid flexible robot manipulator, ball and beam system.
4. Provide a comparison of the developed hybrid algorithms adopted to achieve the optimal control.

1.5. Literature Review

The era of evolutionary computation started with genetic algorithms in the past three decades. Amounts of applications have benefited from the utilization of GA [1]. Potter and De Jong [2] have demonstrated the use of co-operative co-evaluation GA in multivariable functional optimization. Breeder genetic algorithm (BGA) [3] is first introduced by Muhlenbein et al. The major difference lies in the method of selection in comparison to simple GA. A typical task of GA is to find the best values of a predefined set of free parameters associated with either a process model or a control vector. The GA uses the basic reproduction operators such as crossover and mutation to produce the genetic composition of a population. Efforts are being made in the enhancement of conventional algorithm [4-7]. GA with neural network and fuzzy control [8] has also been used extensively to optimize nonlinear and multivariable systems. In the past, researches have been carried out in using hybrid genetic algorithm approaches for optimization problems. Buczak and Uhrig proposed a novel hierarchal fuzzy-genetic [9] information fusion technique. Constraint handling is one of the major concern for solving the optimization problems using GA. Chootinan and Chen proposed a gradient information [10], derived from the constraint set, to systematically repair infeasible solutions.

Though the GA methods have been successful to solve complex optimization problems, recent search has identified some deficiencies in GA performance [11]. This degradation in efficiency is apparent in applications with highly *epistatic* objective functions (i.e. where the parameters being optimized are highly correlated), the genetic operators alone cannot ensure better fitness of offspring because chromosomes in the population have similar structure and their average fitness are high toward the end of evolutionary process. Researches are being done to increase the efficiency of GA by hybridization [12]. Yang et al [13] proposes a hybrid immigrants scheme that combines the concept of elitism, dualism and random immigrants to address dynamic optimization problems.

Particle Swarm Optimization (PSO) was originally introduced by J. Kennedy et al in 1995, which is an evolutionary algorithm based on the swarm intelligence [14-16], and motivated from the simulation of social behavior. The PSO technique can generate a high-quality solution within shorter calculation time and stable convergence characteristic than other stochastic methods. Many researchers are working in the direction of improving the search phenomena of PSO technique. A novel PSO algorithm based on immunity-clonal strategies, called clonal particle swarm optimization (CPSO) [17] is proposed by Y. Tan et al with applications and comparison analysis with standard PSO. Teng-Bo Chen modifies the classical PSO technique in four phases [18]: first, a contractive factor is introduced to the position update equation, and the particles are limited in search region. A new strategy for updating velocity is then adopted, in which the velocity is weakened linearly. Thirdly, two modified PSO algorithms are intersected. Finally, adding an item of integral control in the modified algorithm improves its global search ability. The particle swarm optimizer is also used for solving constrained optimization problems [19] which adopt a very small population size. PSO is hybridized with differential evolution [20] and GA [21-23] to increase the effectiveness in finding the optimal solution.

Natural selection tends to eliminate animals with poor foraging strategies through methods for locating, handling, and ingesting food and favors the propagation of genes of those animals that have successful foraging strategies, since they are more likely to obtain reproductive success [24, 25]. After many generations, poor foraging strategies are either eliminated or shaped into good ones. Since a foraging organism takes actions to maximize the energy utilized per unit time spent foraging, considering constraints presented by its own physiology (e.g., sensing and cognitive capabilities) and environment (e.g., density of prey, risks from predators, physical characteristics of the search area). It is essentially this idea that could be applied to complex optimization problems. In 2007, Kim et al. proposed a hybrid approach involving GA and BF for function optimization [26, 27]. PSO-BF [28] hybridized algorithm is discussed in Arijit et al. work where the mutation is added using PSO in the classical BF optimization.

1.6 Organization of the thesis

The report has been organized into seven chapters. **Chapter 1** gives an impression of the subject, basics, literature survey and objective of the study. **Chapter 2** briefly discussed the basic optimization techniques. **Chapter 3** describes algorithms developed using hybridization of basic optimization techniques. **Chapter 4** contains the introduction of fuzzy logic systems and its optimization. **Chapter 5** gives an overview of the applications used. **Chapter 6** presents the simulation results for controller designed using hybridized algorithms for the applications. **Chapter 7** presents the conclusion of the study and suggestions are given for further study of this subject.

string position based on the mutation probability. An example of the operator is shown in Table-2.1.3.

Table-2.1.2 An example of the crossover of GA's

Old Chromosome	Fitness value	New Chromosome
[101010]	0.3	[111011]
$\updownarrow$ $\updownarrow$		
[010101]	0.5	[000100]
[110110]	0.1	
[011011]	0.9	

Table-2.1.3 An example of the mutation of GA's

Old Chromosome	Fitness value	New Chromosome
[101010]	0.3	
[010101]	0.5	
[110110]	0.1	[110011]
$\uparrow$ $\uparrow$		
0 1		

5. ***Elite Method:*** Due to random process, GA's could loss the best chromosome in a population, so the Elite method is adopted. The best individual in each generation is selected without the three basic operations, and make it as a member of the new generation. Until the other better string is found in the new population, the elite string is superseded. Thus the superior individual in each population is always preserved. So the fitness of the generation is gradually increased generation to generation.

6. ***Reinforced Search Method:*** As above, the individual with higher fitness is selected to copy to a new population. On the contrary, inferior string can be selected to operate. But it has some differences from the Elite method. The worst individual should be mutated with higher mutation rate. We will select the N inferior strings to mutate but the size of N should be small enough. This method has some advantages: 1) using the higher mutation rate makes the search space widely, 2) the probability in searching the global optimum is higher, 3) the method can avoid the population rarefied. Therefore, it can avoid that the solutions premature convergence to form a wrong local-optimum. The full search flowchart is shown in Figure-2.1.1. All elements in a population will be divided into three groups. The left path in Figure-2.1.1 implements the Elite method. The middle ones achieve the Reinforced search method, and the right executes simple GA's with three basic operations.

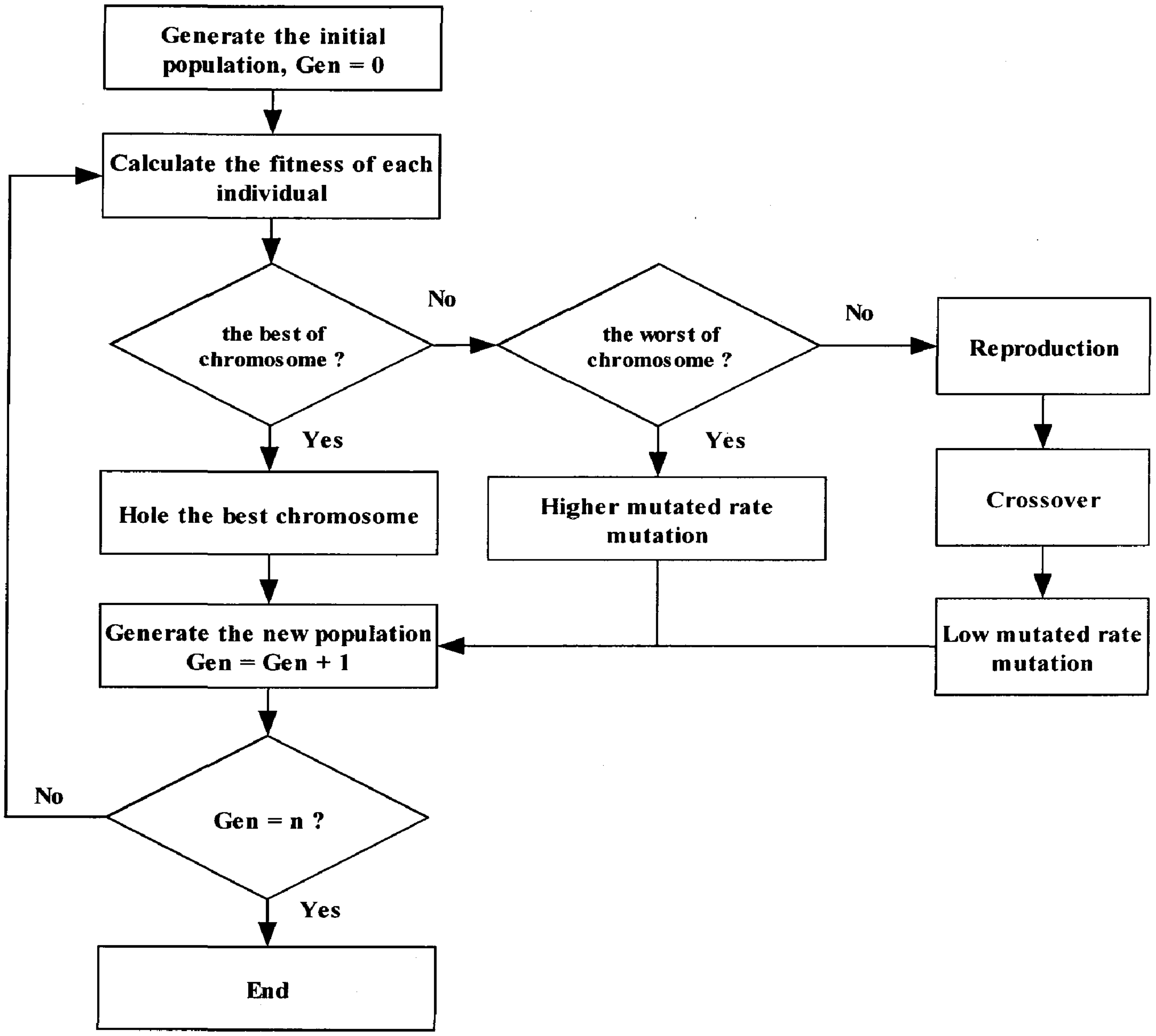

Generate the initial population, Gen = 0
Calculate the fitness of each individual
the best of chromosome ?
No
the worst of chromosome ?
No
Reproduction
Yes
Yes
Hole the best chromosome
Higher mutated rate mutation
Crossover
Generate the new population Gen = Gen + 1
Low mutated rate mutation
No
Gen = n ?
Yes
End

Figure-2.1.1 Flowchart of genetic algorithm a random term, with separate random numbers being generated for acceleration toward pbest and gbest locations.

For example, the *j*th particle is represented as $x_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,g})$ in the *g*-dimensional space. The best previous position of the *j*th particle is recorded and represented as $\text{pbest}_j = (\text{pbest}_{j,1}, \text{pbest}_{j,2}, \ldots, \text{pbest}_j)$. The index of the best particle among all of the particles in the group is represented by the gbest_g. The rate of the position change (velocity) for particle *j* is represented as $v_j = (v_{j,1}, v_{j,2},\ldots, v_{j,g})$. the modified velocity and position of each particle can be calculated using the current velocity and the distance from $\text{pbest}_{j,g}$ to $\text{gbest}_{j,g}$ as shown in the following formulas:

$$v_{j,g}^{(t+1)} = w \cdot v_{j,g}^{(t)} + c_1 * rand() * (pbest_{j,g} - x_{j,g}^{(t)}) + c_2 * rand() * (gbest_g - x_{j,g}^{(t)}) \qquad (2.2.1)$$

$$x_{j,g}^{(t+1)} = x_{j,g}^{(t)} + v_{j,g}^{(t+1)} \qquad (2.2.2)$$

$$j = 1, 2, \ldots, \text{n}$$
$$g = 1, 2, \ldots, \text{m}$$

where

n	number of particle in a group;
m	number of members in a particle;
t	pointer of iterations (generations);
$v_{j,g}^{(t)}$	velocity of particle *j* at iteration *t*, $V_g^{\min} \le v_{j,g}^{(t)} \le V_g^{\max}$;
w	inertia weight factor;
c_1, c_2	acceleration constant;
rand()	random number 0 and 1;
$x_{j,g}^{(t)}$	current position of particle *j* at iteration *t;*
pbest_j	pbest of particle *j;*
gbest	gbest of the group.

In the above procedures, the parameter V^{max} determined the resolution, or fitness, with which regions be searched between the present position and the target position. If V^{max} is too high, particles might fly past good solutions. If V^{max} is too small, particles may not explore sufficiently beyond local solutions. In many experiences with PSO, V^{max} was often set at 10-20% of the dynamic range of the variable on each dimension.

The constants c_1 *and* c_2 represent the weighting of the stochastic acceleration terms that pull each particle toward pbest and gbest positions. Low values allow particles to roam far from the target regions before being tugged back. On the other hand, high values result in abrupt movement toward, or past, target regions. Hence, the acceleration constants c_1 *and* c_2 were often set to be 2.0 according to past experiences.

Suitable selection of inertia weight w in (eq no. 2.2.3) provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution. As originally developed, w often decreases linearly from about 0.9 to 0.4 during a run. In. general, the inertia weight w is set according to the following equation:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter \qquad (2.2.3)$$

Where $iter^{max}$ is the maximum number of iterations (generations), and *iter* is the current number of iterations. The full search flowchart is shown in Figure-2.2.1

2.3. Bacterial Foraging

Bacterial foraging was formally introduced in 2002 by Kevin M. Passino in [24]. The BF is a stochastic search and optimization technique based on the foraging habits of *Escherichia coli*, more commonly known as *E. coli*, a bacterium commonly found in the gut of human beings. A fundamental part of the BF is the movement of the bacterium termed as *chemotaxis*. The chemotactic motion exhibited alternately by the *E. coli*, whilst searching for better forage, is of two distinct types: (1) tumble and (2) runs (swims). The chemotactic motion of *E. coli* is modeled within the BF algorithm according to the possible mediums the bacteria encounters and its response within such mediums. This is summarized as follows:

1. Neutral substance medium: Bacterium tumbles and runs alternately

2. Noxious substance medium: Bacterium tumbles more than it swims as it attempts to get out of the noxious substance (climb down the noxious substance gradient). It essentially seeks favorable conditions.
3. Nutrient substance medium: Bacterium swims more than it tumbles while it searches for even more favorable nutrient mediums (up the nutritious substance gradient).

The implementation of the BF optimization algorithm is summarized in Figure-2.3.1. The highlighted stages – initialization, chemotaxis, swarming, reproduction and elimination/dispersal are described in the following sections.

2.3.1 Components of bacterial foraging

1. ***Initialization:*** At the start of BF algorithm, all the parameters required for its implementation are specified. These include the number of bacteria within the population, the positions of each bacterium within the sample space, the number of chemotactic steps taken during each bacterium lifetime, the number of reproduction and elimination/dispersal events.

2. ***Chemotaxis:*** *E. coli* has the proclivity to convene at nutrient-rich areas by an activity called chemotaxis. They achieve chemotaxis in two different ways: Each bacterium can either 'run', which is movement in a specified direction, or it can 'tumble', which is a movement in a random direction. In order to search for the positions with best nutrient concentrations, each bacterium takes a specified number of chemotactic steps. At the initialization of the algorithm or at the beginning of each chemotactic step, tumble or run/swim within the chemotactic loop, each bacterium has, what is referred to as, its 'initial' position. Nutrient concentration function is used to indicate a progressive search procedure.

The subsequent nutrient concentration values are determined each time a bacterium tumbles or run/swim from its initial position. In the chemotactic step, the tumble must always occur before the run/swim. After a tumble, which is a move in a random direction to a 'new' position, the nutrient concentration is evaluated. The comparison between 'new' nutrient concentration values with 'initial' nutrient

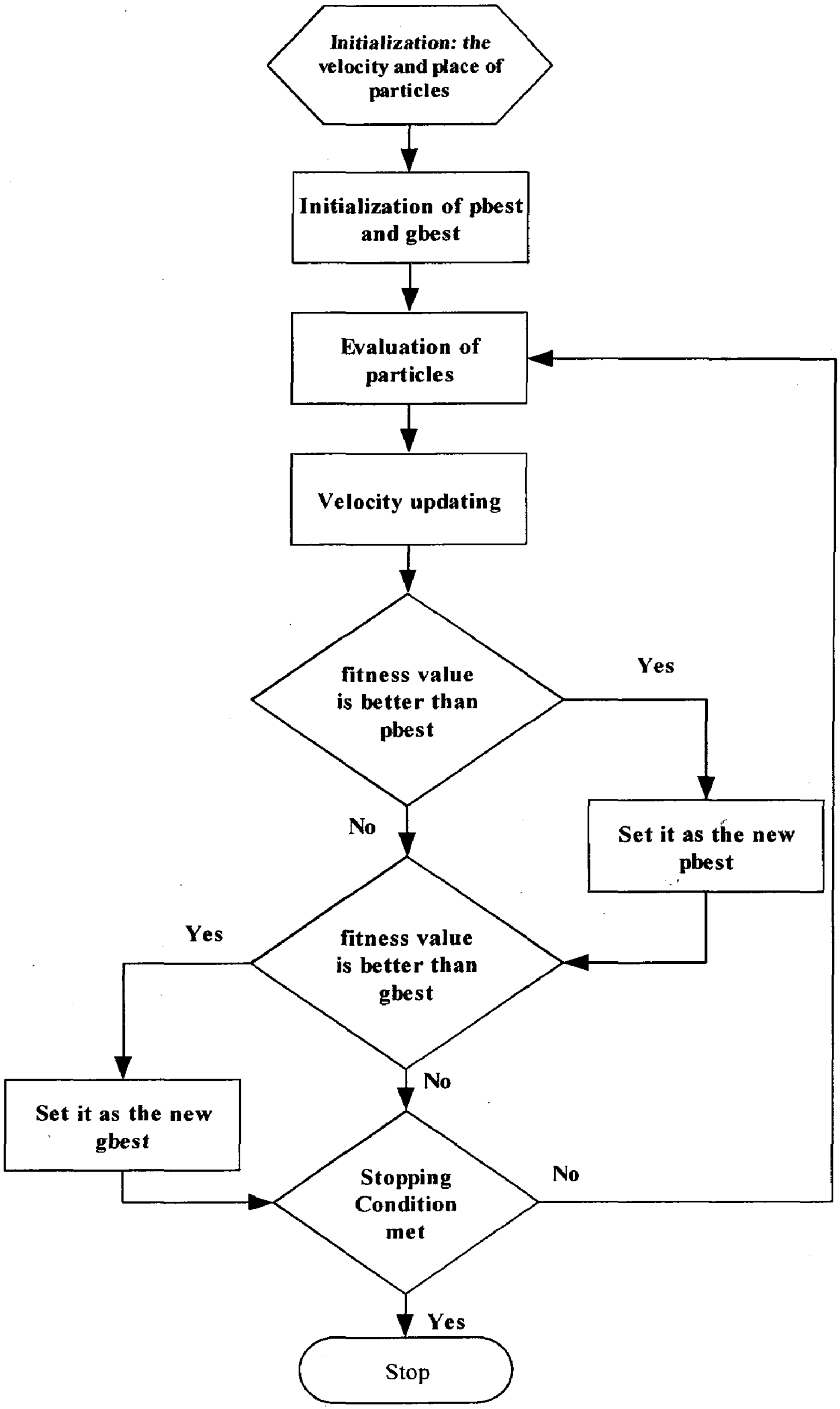
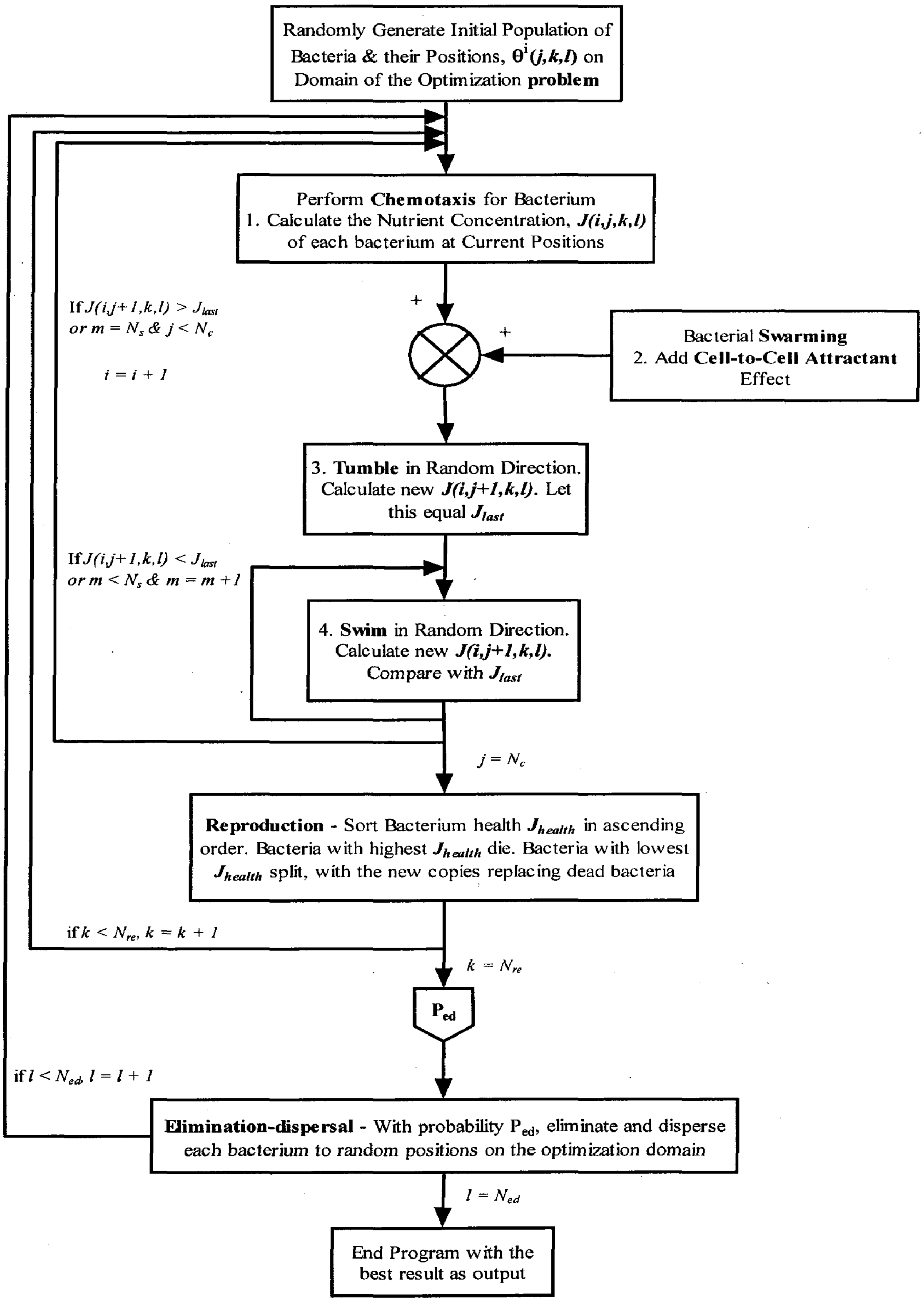

Initialization: the velocity and place of particles
Initialization of pbest and gbest
Evaluation of particles
Velocity updating
fitness value is better than pbest
Yes
Set it as the new pbest
No
fitness value is better than gbest
Yes
Set it as the new gbest
No
Stopping Condition met
No
Yes
Stop

Figure-2.2.1 Flowchart of particle swarm optimization



Randomly Generate Initial Population of Bacteria & their Positions, $\theta^i(j,k,l)$ on Domain of the Optimization problem
Perform Chemotaxis for Bacterium
1. Calculate the Nutrient Concentration, $J(i,j,k,l)$ of each bacterium at Current Positions
+
+
Bacterial Swarming
2. Add Cell-to-Cell Attractant Effect
If $J(i,j+1,k,l) > J_{last}$ or $m = N_s$ & $j < N_c$
$i = i + 1$
3. Tumble in Random Direction. Calculate new $J(i,j+1,k,l)$. Let this equal J_{last}
If $J(i,j+1,k,l) < J_{last}$ or $m < N_s$ & $m = m + 1$
4. Swim in Random Direction. Calculate new $J(i,j+1,k,l)$. Compare with J_{last}
$j = N_c$
Reproduction - Sort Bacterium health J_{health} in ascending order. Bacteria with highest J_{health} die. Bacteria with lowest J_{health} split, with the new copies replacing dead bacteria
if $k < N_{re}$, $k = k + 1$
$k = N_{re}$
P_{ed}
if $l < N_{ed}$, $l = l + 1$
Elimination-dispersal - With probability P_{ed}, eliminate and disperse each bacterium to random positions on the optimization domain
$l = N_{ed}$
End Program with the best result as output

Figure-2.3.1 Flowchart of the Bacterial Foraging Optimization Algorithm concentration value determines whether a swim will follow subsequently. The smaller nutrient concentration value sets the reference point that subsequently follows, for example, say first run/swim. After the first swim occurs, a new bacterium position is obtained. The bacterial position, represented by θ, is given by eq-2.3.1. Within the equation, *i* represent the counter for each bacteria within the population; *j* represent the number of chemotactic steps that each bacteria has undertaken during its lifetime; *k* represent the counter for the reproduction steps while *l* is the indicator for the elimination and dispersal events that occurring.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi(j) \tag{2.3.1}$$

Where $\theta^i(j,k,l)$ represents the position of the i^{th} bacteria at the j^{th} chemotactic step, the k^{th} reproductive step and the l^{th} elimination and dispersal step. *C(i)* is the size of the chemotactic step taken in a random direction by the i^{th} bacteria.

3. ***Swarming:*** When cells of the *E. coli* are randomly distributed in a solution that has varying concentrations of nutrients and noxious substances randomly distributed within it, each bacterium would secrete attractants to signal other cells if it finds that it is swimming in areas with good nutrient concentration. This facilitates the convergence of cells of bacteria to form groups around areas in the solution with high nutrient concentration. This enhances the effectiveness of the search and foraging procedure.

Also, when cells of bacteria experience noxious substances, each cell would secrete repellants to divert the search and foraging process away from the areas with noxious substances. This causes divergence of the bacteria cells, ensuring that they spread out to other areas, this improving the effectiveness of the search procedure. This behavior of the foraging of bacteria termed swarming has been modeled within the BF optimization algorithm. The mathematical expression for swarming can be represented as in eq-2.3.2.

$$J_{cc}(\theta, P(j,k,l)) = \sum_{i=1}^{S} J_{cc}^{i}(\theta, \theta^{i}(j,k,l))$$

$$= \sum_{i=1}^{S}\left[-d_{attract}\exp\left(-w_{attract}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right]$$

$$+\sum_{i=1}^{S}\left[-h_{repellent}\exp\left(-w_{repellent}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2\right)\right]$$

(2.3.2)

$J_{cc}(\theta, P(j,k,l))$ is the cell-to-cell attraction/repulsion function value that is to be added to the nutrient concentration function, which is to be optimized. S is the total number of bacteria, p is the number of parameters to be optimized which are present in each bacterium, $d_{attract}$ and $w_{attract}$ represent the quantification of the depth and width of the repellent secreted by each bacterium.

4. ***Reproduction*:** Reproduction occurs when every cell in the bacteria population has moved the specified number of chemotactic steps. It is necessary to cause all the bacteria within the population to converge at the bacteria population with the best nutrient concentration value. Reproduction is achieved in two stages: first stage essentially involves assigning a fitness value to each bacterium within the population. The fitness value determines which bacteria is fit enough to reproduce. Having achieved the fitness assignment, the bacteria are then ranked according to their fitness values. The bacterium with smaller nutrient concentration values are ranked higher than those with larger values. In order to reproduce, the one-half of the population of bacteria, ranked the least, is kicked off. The remaining half which has the better ranks is replicated – each surviving bacterium splits into two copies of itself. The new population then serves as initial position for the next chemotactic process or the next elimination-dispersal event.

5. ***Elimination and Dispersal*:** The process simply involves randomly killing off some of the poorly performing bacteria within the population. The need for this is simply to provide room for new members of the bacterium population that potentially are situated in areas ith higher nutrient concentration. The obvious effect of elimination is a reduction in the total bacterial population. In order to counter the reduction, a complimentary process termed Dispersal occurs.

The eliminated bacteria are randomly replaced by new ones, which are probably situated in different (and possibly better) locations than the previously existing members of the population. These new locations might be better because they may be situated closer to spaces with better nutrient concentration.

3. HYBRIDIZATION OF INTELLIGENT COMPUTATIONAL TECHNIQUES

The description of basic techniques for intelligent computation has been highlighted in chapter 2. This chapter focuses on the hybridization of three basic algorithms employed for the optimal automated control. For several problems a simple optimization algorithm might be good enough to find the desired solution. As reported in the literature, there are several types of problems where a direct evolutionary algorithm could fail to obtain a convenient (optimal) solution. This clearly paves way to the need for hybridization of basic optimization algorithms. Some of the possible reasons for hybridization are as follows:

1. To improve the performance of the evolutionary algorithm (example: speed of convergence)
2. To improve the quality of the solutions obtained by the evolutionary algorithm
3. To incorporate the evolutionary algorithm as part of a larger system.

3.1. Hybrid Bacterial Foraging

The Hybrid Bacteria Foraging (HBF) is an example of a hybrid evolutionary algorithm. Its name has been chosen for two reasons:

1. it is formed from the combination of the GA and the BF
2. it is largely similar in implementation to the BF.

Similar hybrid evolutionary algorithms have been developed and used in [32], [33], [26] and [34] but the novelty of the developed algorithm lies especially in the uniqueness of the application it has been specifically developed for. The circumstance that led to the development of the HBF stemmed from the process of investigating ways to improve the effectiveness of the simple BF optimization algorithm (described in section 2.3) as an approach for the design of robust controllers for the electronic drive. In the formation of the HBF optimization algorithm, the aim was to combine specific desirable functions of operator uses only the 'social' component and eliminates the 'cognitive' component as the local search in different regions of the search space is already taken care of by the chemotactic steps of the BF algorithm.

Here the approach used is somewhat different from the original paper, instead of mutation step using PSO in the basic BF algorithm the search direction vector is made adaptive using PSO. The search vector is made constant in [28] and in this thesis, it gets changed after all the bacteria gets tumble and swim in the particular direction. This vector helps in early convergence of the fitness function as the direction vector is not constant; the individual knows how much it should move to reach the optimal solution. Basic velocity and position equation as described in section 2.2 are used to update the search vector. This combination of both BFO and PSO aims to make use of PSO ability to exchange social information and BFO ability in finding a new solution by elimination and dispersal. The (BFO-PSO) models bacterial Population Chemotaxis, swarming, reproduction, elimination and dispersal oriented by PSO, is shown in flowchart form as in Figure-3.2.1. (Initially, $j = k = l = 0$).

Initialize parameters n, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, $c(i)(i = 1, 2... S)$, $Delta$, C_1, C_2, R_1, R_2. where,

- n : Dimension of the search space,
- S : The number of bacteria in the population,
- S_r : Half the total number of bacteria ,
- N_s : Maximum number of swim length,
- N_c : Chemotactic steps,
- N_{re} : The number of reproduction steps,
- N_{ed} : Elimination and dispersal events,
- P_{ed} : Elimination and dispersal with probability,
- $c(i)$: The step size taken in the random direction,
- C_1, C_2 : PSO random parameter,
- R_1, R_2: PSO random parameter.

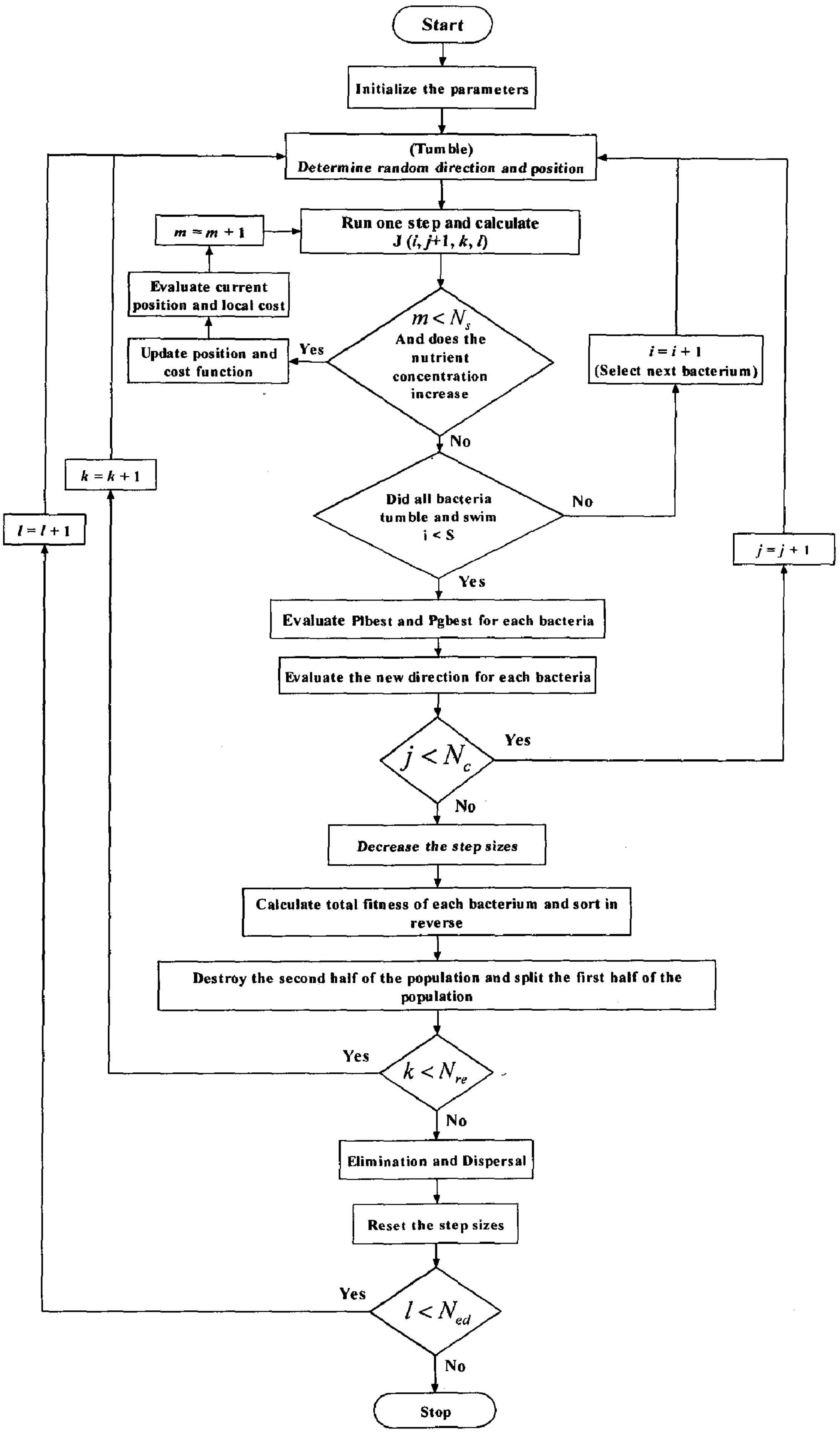

Start
Initialize the parameters
(Tumble)
Determine random direction and position
Run one step and calculate
J (i, j+1, k, l)
m = m + 1
Evaluate current position and local cost
Update position and cost function
Yes
$m < N_s$
And does the nutrient concentration increase
No
i = i + 1
(Select next bacterium)
Did all bacteria tumble and swim
i < S
No
Yes
k = k + 1
l = l + 1
j = j + 1
Evaluate Plbest and Pgbest for each bacteria
Evaluate the new direction for each bacteria
$j < N_c$
Yes
No
Decrease the step sizes
Calculate total fitness of each bacterium and sort in reverse
Destroy the second half of the population and split the first half of the population
Yes
$k < N_{re}$
No
Elimination and Dispersal
Reset the step sizes
Yes
$l < N_{ed}$
No
Stop

Figure-3.2.1. Flowchart of Hybrid Bacterial Foraging and Particle Swarm based Optimization

N_{re},	No. of reproduction steps
N_{ed},	No. of elimination-dispersal events
P_{ed},	Elimination-dispersal with probability
$C(i)$,	size of the step taken in the random direction specified by the tumble
$\vec{\theta}^{i}$,	Position vector of the i-th bacterium, in j-th chemotactic step, k-th reproduction

[Step 2] Elimination-dispersal loop: $l = l + 1$.

[Step 3] Reproduction loop: $k = k + 1$.

[Step 4] Chemotaxis loop: $j = j + 1$.

[substep a] For $i = 1,2,...S$, take a chemotactic step for bacterium i as follows.

[substep b] Compute fitness function, $J(i, j, k, l)$

[substep c] Let $J_{last} = J(i, j, k, l)$ to save this value since we may find a better cost via a run

[substep d] Tumble: generate a random vector $\Delta(i) \in R^{p}$ with each element $\Delta_{m}(i), m = 1,2,..., p,$ a random number on [-1,1]

[substep e] Move: Let $\theta^{i}(j+1,k,l) = \theta^{i}(j,k,l) + C(i)\dfrac{\Delta(i)}{\sqrt{\Delta^{T}(i)\Delta(i)}}$

[substep f] Compute $J(i, j+1, k, l)$, and then let

$$J(i, j+1, k, l) = J(i, j+1, k, l) + J_{cc}(\theta^{i}(j+1, k, l), P(j+1, k, l))$$

[substep g] Swim

i) Let m=0 (counter for swim length)

ii) While $m < N_{s}$ (if have not climbed down too long)

- Let m=m+1
- If $J(i, j+1, k, l) < J_{last}$ (if doing better)

  let $J_{last} = J(i, j+1, k, l)$ and let

$$\theta^{i}(j+1,k,l) = \theta^{i}(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^{T}(i)\Delta(i)}}$$

and use this $\theta^i(j+1,k,l)$ to compute the new $J(i,j+1,k,l)$ as we did in [substep f]

$P_{current}(i,j+1) = P(i,j+1)$
$J_{local}(i,j+1) = J_{last}(i,j+1)$

- Else, $\begin{matrix} P_{current}(i,j+1) = P(i,j+1) \\ J_{local}(i,j+1) = J_{last}(i,j+1) \end{matrix}$, let $m = N_s$. This is the end of while statement.

[substep h] Ranking and Select individuals from population using SUS.

[substep i] Recombine selected individuals using crossover operator.

[substep j] Mutate offsprings.

[substep k] Go to next bacterium $(i,1)$ if $i \neq S$ (i.e. go to [substep b] to process the next bacterium)

[substep l] Evaluate the local best position (P_{lbest}) for each bacteria and global best position (P_{gbest})

[substep m] Evaluate the new direction for each bacteria

$$V = \omega * V + C_1 * R_1(P_{lbest} - P_{current}) + C_2 * R_2(P_{gbest} - P_{current})$$

$$\Delta = V$$

[Step 5] If $j < N_c$, go to step 3. In this case, continue chemotaxis, since the life of bacteria is not over

[Step 6] Reproduction:

[substep a] For the given k and l, and for each $i = 1,2,...,S$, let

$$J^i_{health} = \sum_{j=1}^{N_c+1} J(i,j,k,l)$$

be the health of the bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances). Sort bacteria and chemotactic parameters $C(i)$ in order of ascending cost J_{health} (higher cost means lower health)

[substep b] The S_r bacteria with the highest J_{health} values die and the other S_r

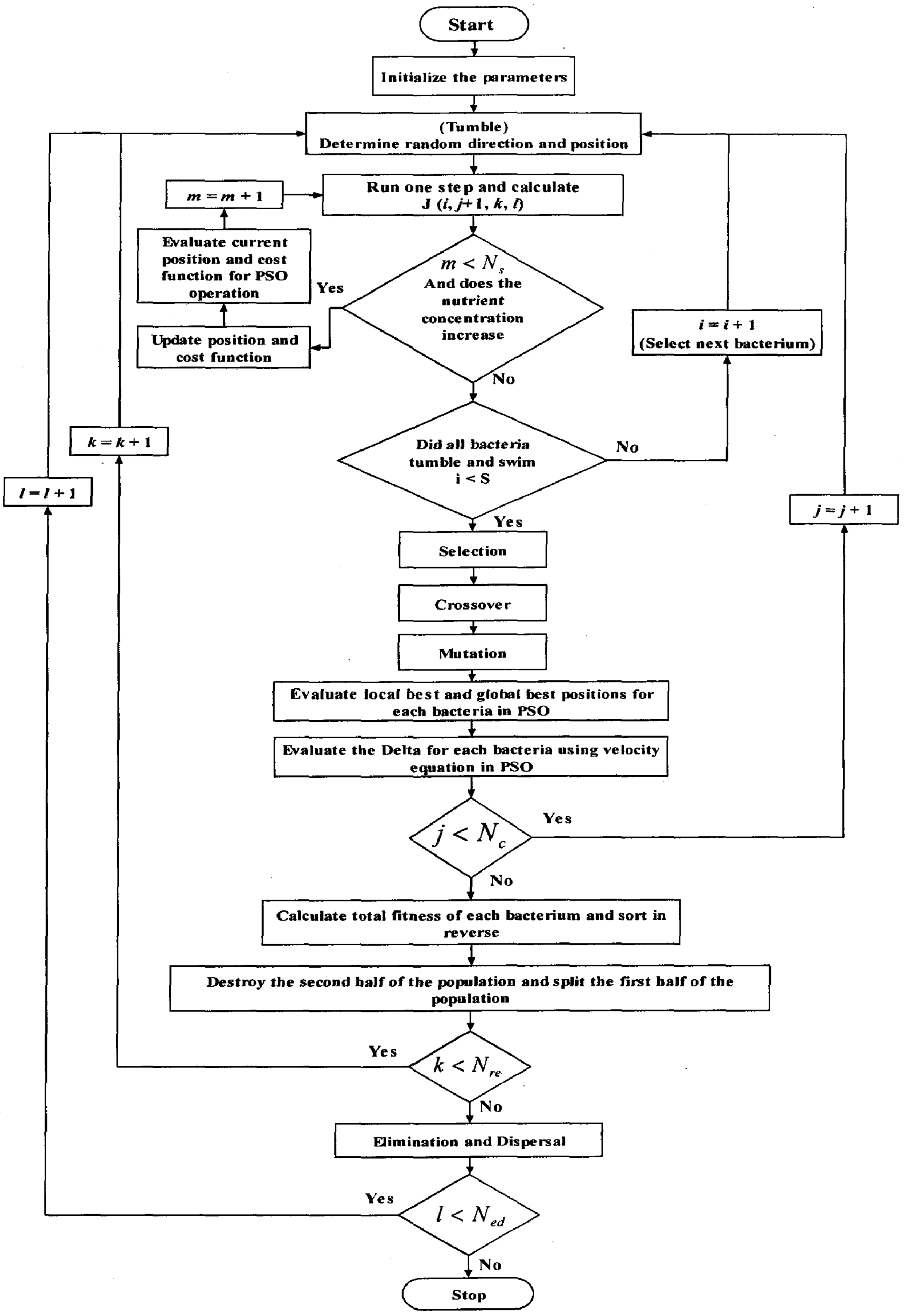

Start
Initialize the parameters
(Tumble) Determine random direction and position
Run one step and calculate $J(i, j+1, k, l)$
$m = m + 1$
Evaluate current position and cost function for PSO operation
Update position and cost function
Yes
$m < N_s$ And does the nutrient concentration increase
No
$i = i + 1$ (Select next bacterium)
$k = k + 1$
$l = l + 1$
Did all bacteria tumble and swim $i < S$
No
Yes
$j = j + 1$
Selection
Crossover
Mutation
Evaluate local best and global best positions for each bacteria in PSO
Evaluate the Delta for each bacteria using velocity equation in PSO
$j < N_c$
Yes
No
Calculate total fitness of each bacterium and sort in reverse
Destroy the second half of the population and split the first half of the population
Yes
$k < N_{re}$
No
Elimination and Dispersal
Yes
$l < N_{ed}$
No
Stop

Figure3.3.1. Flowchart of Hybrid Genetically-Bacterial Foraging converged by Particle swarm optimization

Bacteria with the best values split (and the copies that are made are placed at the same location as their parent).

[Step 7] If $k < N_{re}$, go to step 3. We have not reached the specified number of reproduction steps. So we start the next generation in the chemo-taxis loop.

[Step 8] Elimination-dispersal: For $i = 1,2,...,S$, with probability P_{ed}, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). If $l < N_{ed}$, then go to step 2; otherwise end.

The flowchart depicting the step by step algorithm discussed above is shown in Figure-3.3.1.

4. FUZZY LOGIC AND FUZZY CONTROL

Conventional control system design depends upon the development of a mathematical description of the system's behavior. This usually involves assumptions being made in relation to the system dynamics and any non-linear behavior that may occur. In cases where assumptions in respect of non-linear behavior cannot be made, the need to describe mathematically, ever increasing complexity becomes difficult and perhaps infeasible.

Fuzzy logic [36] is the application of logic to imprecision and has found application in control system design in the form of Fuzzy Logic Controllers (FLCs). Fuzzy logic controllers facilitate the application of human expert knowledge, gained through experience, intuition or experimentation, to a control problem. Such expert knowledge of a system's behavior and the necessary intervention required to adequately control that behavior is described using imprecise term known as "linguistic variables". The imprecision of linguistic variables reflects the nature of human observation and judgment of objects and events within our environment, and there use in FLCs thus allows the mapping of heuristic, system-related information to actions observed to provide adequate system control. In this way, FLCs obviate the need for complex mathematical descriptions of non-linear behavior to the n^{th} degree and thus offer an alternative method of system control.

4.1. Fuzzy Logic Controller

Figure-4.1 shows the structure of Fuzzy controller. It consists of a preprocessing, fuzzification interface, knowledge base, fuzzy inference system, defuzzification interface and a post processing unit. The preprocessing block transforms the input (e and $\dot{e}$) on the actual universe of discourse (UOD) to the normalized universe of discourse, using the input scaling factors K_P, K_D and K_C for computational simplicity. The fuzzification block converts crisp inputs to appropriate fuzzy sets using the membership functions.

4.1.1. Linguistic Variable, Rule Bases and Membership Functions

Linguistic variables are descriptive terms that might be used, and best understood, by an expert of the system under consideration, which describe the behavior of a system and the applied actions required to control that system. For the FLC in this study, the linguistic variables are based upon the error *e(t)*, and the rate-of-change of error, *de/dt*.

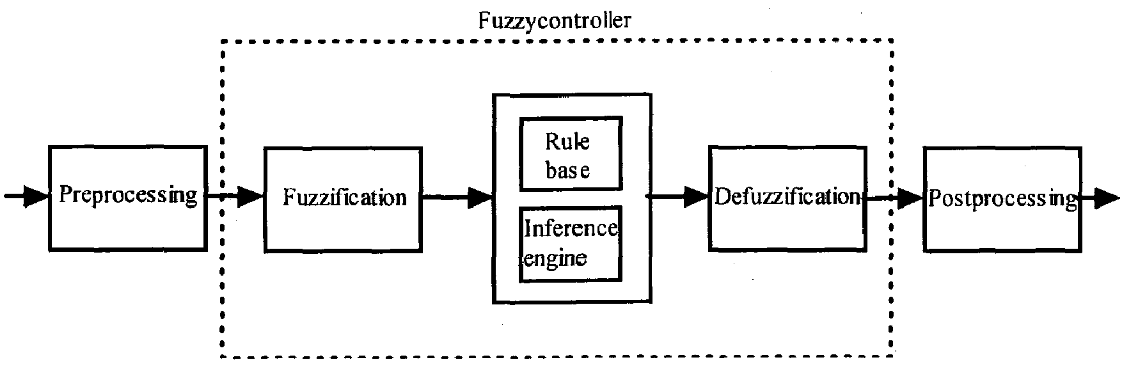


Figure-4.1 Structure of fuzzy controller

The rule base [36] of a FLC consists of a set of behavior/action constructs that describe the action to be taken on the occurrence of particular observed/measured system behavior or state. The constructs consist of a *premise* (i.e. system behavior/state) and the associated *consequent* (i.e. the action to be taken in order to achieve adequate system control under the observed system behavior/state) used in an '*if* ***premise*** *then* ***consequent***' form. Combinations of multiple premises and consequents are possible which enhance the precision of the rule -base. The rule base of a FLC must adequately cover all possible system behavior in respect of applied actions, in order for the FLC to provide reliable system control.

The above descriptions of linguistic variables and rule -bases do not in themselves render the controller 'fuzzy', since, as defined, they could be adequately used in a Boolean-based system. What makes the controller 'fuzzy' is the use of membership functions (MFs) [36] to quantify to what *degree of certainty* each rule is true (i.e. fired) in respect of the system state at any particular time. The 'shapes' and relative spacing of the MFs form a critical element of the FLC and describe expert *understanding* of the meaning of the linguistic variables. Typical MF shapes are triangular, trapezoidal, sigmoid or custom-based, with several MFs used to partition the domain of the numeric value under consideration (i.e. the universe of discourse UOD).

The use of MFs ensures that certainty, as defined within a FLC, is based upon the *subjective interpretation* of an expert rather than upon a probability distribution. Degrees of certainty (i.e. degree of membership of a fuzzy set) range from 0 to 1 in value and hence partial membership is possible. The FLC aggregates the levels of certainty for the entire rule -base to obtain an aggregate fuzzy output set, which is subsequently used to obtain a crisp (i.e. numerically valued), control action. The combination of the rule -base (RB), and associated membership functions (MF), constitute the controller knowledge base (KB), which in effect represents the embedded expert system knowledge. In general, two forms of FLC are defined [37],

- Mamdani
- Sugeno

Both of these architectures are similar in all respects except for the formulation of the output crisp value. In the Mamdani FLC, the output is formulated using fuzzy sets whereas the Sugeno type FLC uses single -spike output MFs (i.e. singletons) rather than distributed functions [37].

4.1.2. Fuzzification

This is the process of transforming numeric inputs to fuzzy values [36]. The premise(s) of each rule is evaluated in respect of its degree of membership of the fuzzy sets defined across the range of possible values that the input may assume (i.e. the universe of discourse). For example, Figure-4.1.2 below shows the MFs for the error input as generated using the MATLAB fuzzy GUI. An error input value of 0.1 for the position controller, corresponds to a degree of membership of approximately 0.75 for the ZERO fuzzy set and a degree of membership of approximately of 0.25 for the PS (positive small) fuzzy set (i.e. $\mu ZERO[e(t)]=0.75$ $\mu NS[e(t)]=0.25$). Degree of membership of all other fuzzy sets in the universe of discourse for the error, where $e(t) = 0.1$, is zero.

4.1.3. Inference

Having fuzzified the controller inputs, the inference process consists of two phases;

- **Rule Matching**

The controller evaluates the applicability of each of the rules with respect to the current system state using fuzzy operators (e.g. min). Where a rule contains only a single premise then this stage will return the value obtained from the fuzzification process. FLCs commonly use multiple premises within each rule and therefore the certainty as to what degree the rule *as a whole* applies to the current system state must be evaluated. To perform the evaluation, the controller applies a logic operator to the fuzzified values of the inputs. Two operators commonly used for the AND conjunction are the *minimum* and *product* operators (for OR conjunction, the max operator is commonly used). For the position controller, the min operator was used.

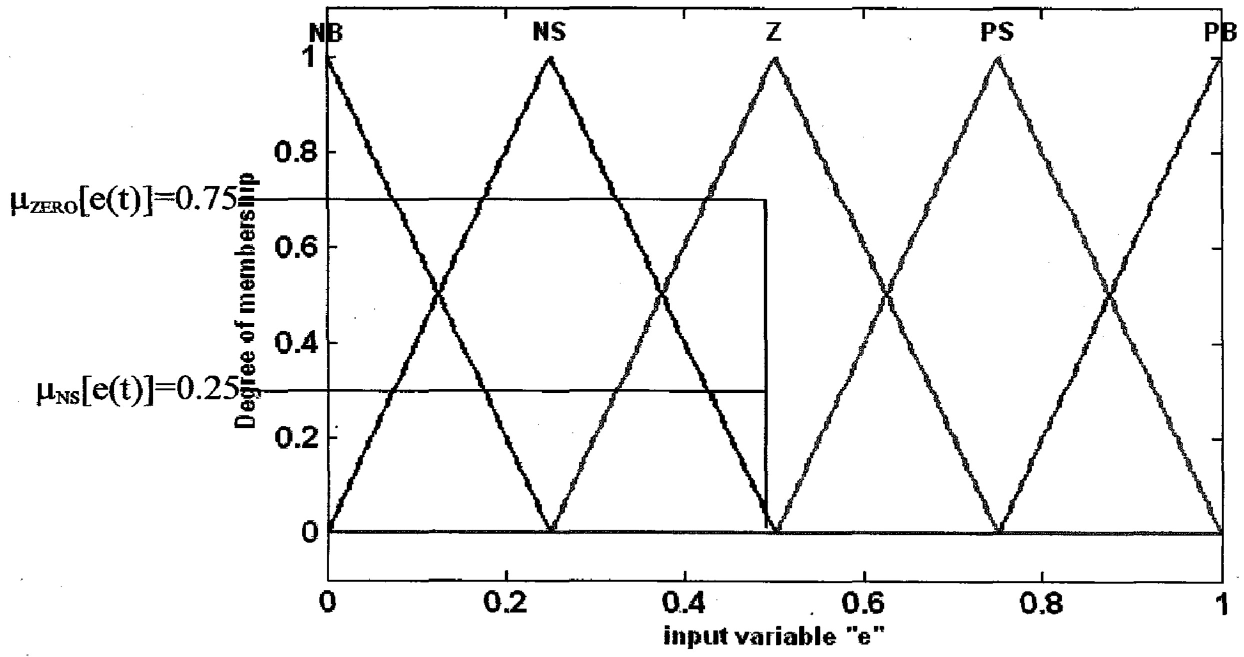


Figure-4.1.2 Degree of membership of Z and NS for input, e = 0.425

- **Implied Conclusions**

The consequent of each rule is a fuzzy set, which is truncated in accordance with the degree of certainty that the premise or conjunction of premises, of the rule applies to the current system state. The degree of certainty for the rule is evaluated by matching rules to the current system state using the FLC inputs as is outlined in the previous section. For all rules therefore deemed to be 'fired' (i.e. that apply) an implication operator is applied to the consequent fuzzy set in order to truncate the set relative to the degree of firing for the rule. So for the example above where the min operator was used to evaluate a degree of certainty for the rule to be 0.25, then accordingly the consequent fuzzy set is truncated by this amount.

4.1.4 Rule Base

As stated, the rule -base consists of a set of linguistic variable constructs in the form of;

if premise_1 and/or.... premise_n then consequent1 and/or..... consequent_m

which describes the system behavior or states to a level of resolution considered to adequately cover all expected states or behavior and the required actions. The number of rules is dependent upon the number of controller inputs and the number of linguistic variables used to describe those variables. For the position controller in this study, 2 inputs are used with 5 linguistic variables to describe the nature of those inputs relative to their universe of discourse, which results in at most 52 = 25 rules. Although in this case, every scenario has an associated entry, it is possible to leave a particular space blank, which would infer that the controller takes no action (i.e. output remains the same as previously).

For systems with 1, 2 or 3 inputs, a tabular form of the rule-base can be constructed. Figure-4.1.3 illustrates the rule -base used for the heuristic position controller in tabular form.

$\dot{e}$ \ e	**NB**	**NS**	**Z**	**PS**	**PB**
NB	NS	NS	NB	NB	Z
NS	NS	NS	NB	Z	PB
Z	NB	NB	Z	PB	PB
PB	NB	Z	PB	PS	PS
PS	Z	PB	PB	PS	PS

Figure-4.1.3 Heuristically-tuned FLC rule-base

The rule -base above was arrived at through intuition and trial, using Simulink, and is not necessarily optimal for the system. A feature of the rule-base used is the symmetry across the diagonal. This feature occurs in systems where the physical behavior of the system exhibits symmetry, which is consistent in the case of the cart-positioning model used in this study where the surface upon which it travels is even and considered identical in both possible directions of travel. Where systems display such symmetry, obtaining, or optimizing a rule -base may prove quicker if the symmetrical feature can be exploited to some extent.

4.1.5 Defuzzification

The final process of the FLC is to aggregate the fuzzy sets resulting from the inference mechanism to produce a decision (i.e. crisp output), which is the "most certain" in respect of the current system behavior.

A number of methods can be used for defuzzification (e.g. center-average, mean-of-maxima), however the most commonly used method is the equation for computation of center-of-gravity (COG), or centroid, which ensures a smooth control action but which requires more complex calculations particularly for non-linear MFs [36].

4.2. Designing a Fuzzy Logic Controller

In this chapter, a demonstration is given of how to automate the design of a Fuzzy Logic Controller. The assumptions used and the constraints introduced to simplify this process are explained.

4.2.1 Assumptions and Constraints

To apply the Fuzzy Logic Controller to various control engineering problems, certain properties of the system are exploited so that the design of the controller can be made easier. As the system used are symmetrical, it is assumed that symmetrical membership functions about the y-axis will provide a valid controller. A symmetrical rule-base is also assumed.

Other constraints are also introduced to the design of the FLC:

- All universes of discourses are normalized to lie between –1 and 1 with scaling factors external to the FLC used to give appropriate values to the variables.
- It is assumed that the first and last membership functions have their apexes at –1 and 1 respectively. This can be justified by the fact that changing the external scaling would have similar effect to changing these positions.
- Only triangular membership functions are to be used.
- The number of fuzzy sets is constrained to be an odd integer greater than unity. In combination with the symmetry requirement, this means that the central membership function for all variables will have its apex at zero.
- The base vertices of membership functions are coincident with the apex of the adjacent membership functions. This ensures that the value of any input variable is a member of at most two fuzzy sets, which is an intuitively sensible situation. It also ensures that when a variable's membership of any set is certain, i.e. unity, it is a member of no other sets.

Using these constraints the design of the membership functions can be described using two parameters:

- The number of membership functions
- The positioning of the triangle apexes

4.2.2. Spacing Parameter

The second parameter specifies how the centers are spaced out across the universe of discourse. A value of one indicates even spacing, while a value larger than unity indicates that the membership functions are closer together in the centre of the range and more spaced out at the extremes as shown in Figure-4.2.1. The position of each centre is calculated by taking the position the centre would be if the spacing were even and by raising this to the power of the spacing parameter. For example, in the case where there are five sets, with even spacing (p=1) the centre of one set would be at 0.5. If p is set to two, the position of this centre moves to 0.25. If the spacing parameter is set to 0.5 then this centre moves to 0.707 in the normalized universe of discourse.

This method of designing the membership functions is inspired by the work of Park et al. [38] and Cheong et al [39]. It does mean that there is a reduction in the number of possible FLCs than if the design was fully flexible but the trade-off is that the design process is made much simpler. Also it is felt that even within these constraints there is sufficient flexibility to allow a FLC that meets the design requirements to be built.

4.2.3. Designing the Rule-Base

As well as specifying the membership functions, the rule-base also needs to be designed. Again ideas presented by Park et al. [38] were used. In specifying a rule base, characteristic spacing parameters for each variable and characteristic angle for each input variable less one are used to construct the rules.

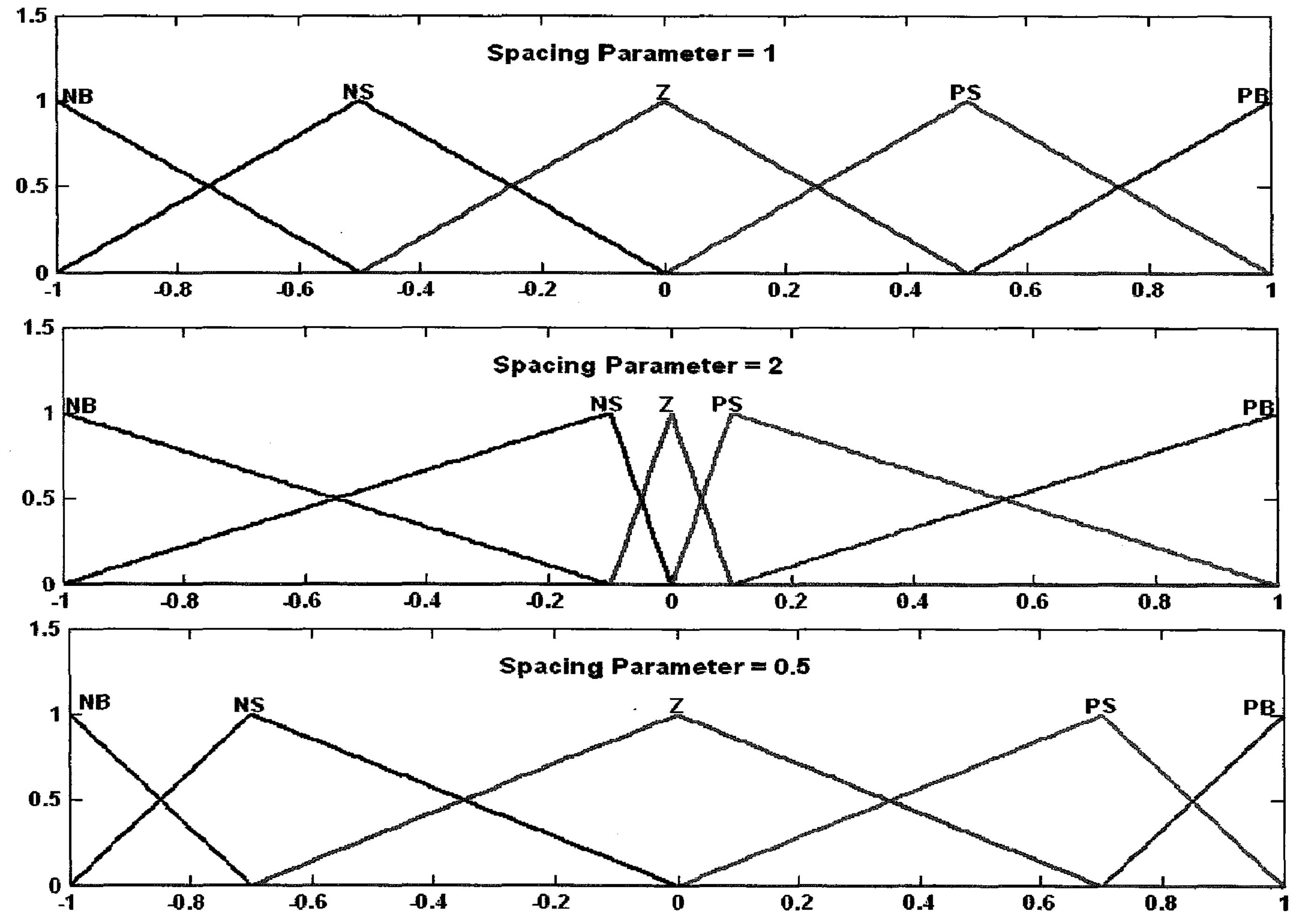


Figure-4.2.1 Effects of spacing parameters on Mfs

Grid points are also placed in the output space representing each possible combination of input linguistic values. These are spaced in the same way as before. The rule-base is determined by calculating which seed-point is closest to each grid point. The output linguistic value representing the seed-point is set as the consequent of the antecedent represented by the grid point. This is illustrated in Figure-4.2.2, which is a graph showing seed points (blue circles) and grid-points (red circles).

Table-4.2.1 shows the derived rule base. The lines on the graph delineate the different regions corresponding to different consequents. The parameters for this example are 0.9 for both input spacing, 1 for the output spacing and 45° for the angle.

4.3 Hybrid Fuzzy Precompensated PD Controller Design

Conventional control methods, such as PD and PID controllers, are widely used in industrial applications. Such controller suffers from the problem of large overshoots and undershoots in its output response when applied to systems containing distributed parameters, nonlinearity, strong coupling. In this work, a novel fuzzy logic-based precompensation approach for controlling system with high nonlinearity is attempted. The control structure consists of a fuzzy logic-based precompensator followed by a conventional PD controller.

Table-4.2.1 Derived Rule base

$\dot{e}$ / e	**NB**	**NS**	**Z**	**PS**	**PB**
NB	NB	NB	NS	NS	Z
NS	NB	NS	NS	Z	PS
Z	NS	NS	Z	PS	PS
PB	NS	Z	PS	PS	PB
PS	Z	PS	PS	PB	PB

4.3.1 PD Control

In this section we describe a general PD (Proportional-Derivative) controller for the control of rigid-flexible link arm. The control input to the i^{th} actuator is given by,

$$\tau_i = -K_{Pi}\hat{e}_i(t) - K_{Di}\dot{\hat{e}}_i(t) \tag{4.3.1}$$

where,

$$\hat{e}_i(t) = [\hat{q}_{di}(t) - q_i(t)]$$

In Equation above, $\hat{e}_i(t)$ and $\dot{\hat{e}}_i(t)$ represent the error in the joint angle and velocity for the i^{th} link. K_{Pi} and K_{Di} are the proportional and derivative gains, respectively. Sufficient literature dealt PD control investigated that better response has not been achieved in case of this distributed parameters, nonlinearity, strong coupled systems.

4.3.2 Fuzzy Precompensation

The fuzzy precompensator uses the desired position q_d, and the plant output q, to generate a precompensated modified desired position $\hat{q}_d$. The fuzzy pre-compensator modifies the desired position, to compensate undershoot and overshoot in the output response, to eliminate the steady-state error and improve the performance of the output response for PD control systems with nonlinearities by introducing a fuzzy logic controller in front of the PD controller. Fuzzy precompensation that proposed is indeed insensitive to nonlinearities, and exhibits good transient and steady-state behavior.

A Fuzzy Precompensated PD controller was designed and applied to control the position of the manipulator, since it is well known that Fuzzy PD gives a faster transient than Fuzzy PI type. The fuzzy precompensator is described by the equations,

$$e_i(t) = [q_{di}(t) - q_i(t)] \tag{4.3.2}$$

$$\gamma(t) = F[e_i(t), \dot{e}_i(t)] \tag{4.3.3}$$

$$\hat{q}_{di}(t) = q_{di}(t) + \gamma(t) \tag{4.3.4}$$

The inputs to the precompensator are the desired position $q_{di}(t)$ and the plant actual position $q_i(t)$. The term $F[e_i(t), \dot{e}_i(t)]$ is a nonlinear mapping of $e_i(t)$ and $\dot{e}_i(t)$ based on fuzzy logic (described below). The term $\gamma(t) = F[e_i(t), \dot{e}_i(t)]$ represents a compensation or correction term, so that the compensated/modified desired position $\hat{q}_{di}(t)$ is simply the sum of the external desired position $q_{di}(t)$ and $\gamma(t)$. The correction term is based on the error $e_i(t)$, and the change of error $\dot{e}_i(t)$. The compensated command signal $\hat{q}_{di}(t)$ is used as the input to the PD controller, as shown in Figure-4.3.1 in the overall block diagram of the hybrid fuzzy precompensated PD controller.

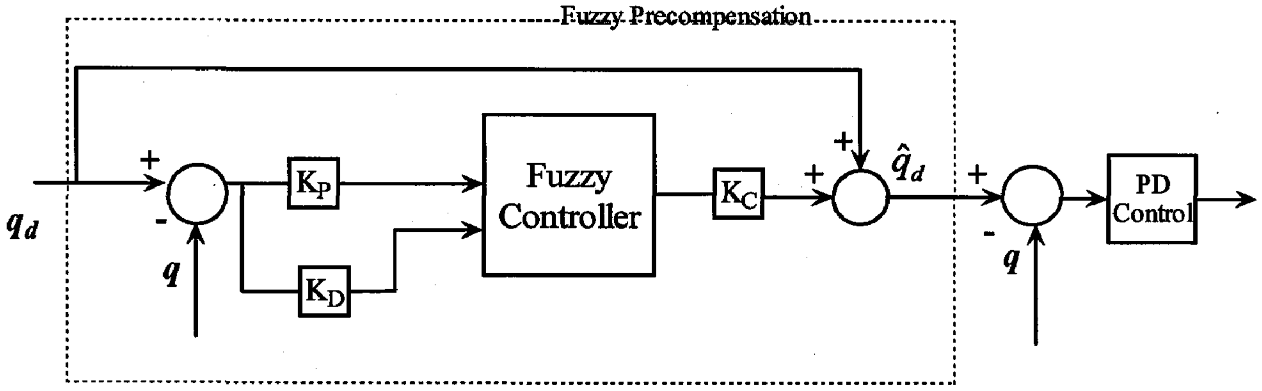


Figure-4.3.1 Overall block diagram of the proposed hybrid controller.

5. APPLICATIONS OF INTELLIGENT COMPUTATIONAL TECHNIQUES

In order to assess the performance of the optimization algorithms, various control engineering problems were considered. Some of the practical applications used are:

1. Inverted Pendulum
2. Ball and Beam System
3. Two link rigid-flexible robot manipulator

5.1. Inverted Pendulum

The inverted pendulum control problem [40] is usually presented as a pole balancing task. The system to be controlled consists of a cart and a rigid pole hinged to the top of the cart. The cart can move left or right on a one-dimensional bounded track, whereas the pole can swing in the vertical plane determined by the track. The linearized system equations around $\theta = \pi$ in the state space are:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{-(I+ml^2)b}{I(M+m)+Mml^2} & \dfrac{m^2gl^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{-mlb}{I(M+m)+Mml^2} & \dfrac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \dfrac{ml}{I(M+m)+Mml^2} \end{bmatrix} u \qquad (5.1.1)$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \qquad (5.1.2)$$

Where,

M = mass of cart = 0.5 kg

m = mass of pendulum = 0.2 kg

b = friction of cart = 0.1 N/m/sec

I = inertia of pendulum = 0.006 kgm^2

l = length of pendulum's center of mass = 0.3 m

F = force applied to cart

The state of the system is defined by values of four system variables: $(x, \dot{x}, \theta, \dot{\theta})$ the cart position, cart velocity, pendulum angle and angular velocity of the pendulum pole, respectively. Control force is applied to the system to prevent the pole from falling while keeping the cart within the specified limits.

5.2. Ball and Beam System

The ball-beam system is a frequently encountered example of nonlinear dynamical system. While the ideal system is indeed nonlinear, its practical implementation has additional non-linearities, including: deadband, backlash introduced by the DC motor and gearbox, discrete position sensing and uneven rolling surface. The system is a product from Googol Technology. The practical system is shown in Figure-5.2.1.

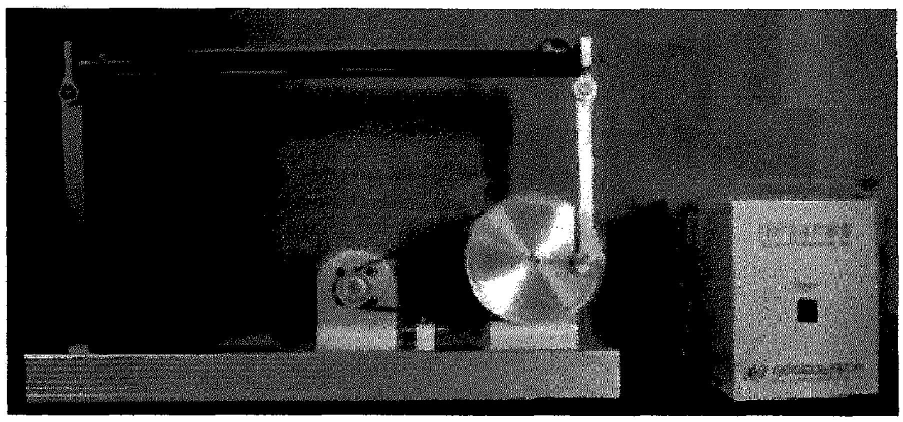


Figure-5.2.1. Functional components of mechanical plant

The motion of the motor's shaft is governed by IPM100 intelligent drive. This is a high precision, fully digital servo drive with embedded intelligence and 100W power amplifier suitable for brushless/brush motors. Based on feedback information from sensors, it computes and then applies appropriate PWM modulated voltage to the motor windings in such a way that a sufficient torque moves the motor shaft according the programmed control algorithm. This embedded intelligence provides a true real-time control performance independent of any delays caused by PC's non-real time Operating System.

The closed loop control algorithm employed for the application is given in Figure-5.2.2:

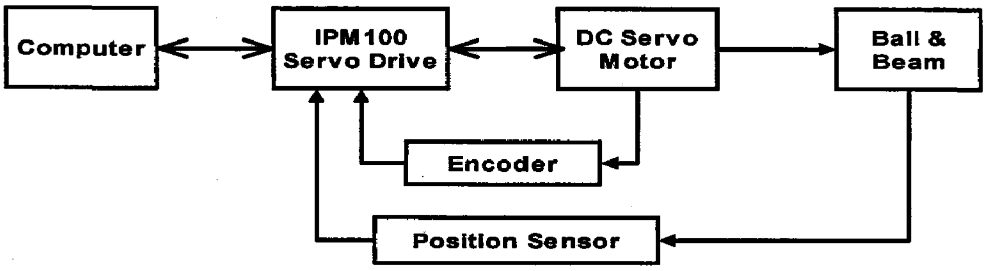


Figure-5.2.2. Structure of the control algorithm

The DC motor provides actuation of the beam via a gear. The PID control algorithm inside IPM100 intelligent drive is employed in an inner control loop as a motor position controller. The PID gains are tuned in such a way that the motor exhibits a fast response without overshoot. The flowchart of the control algorithm is depicted in Figure-5.2.3.

5.2.1 Mechanical Model of Ball & Beam

The schematics of the Ball&Beam mechanical system is shown in Figure-5.2.4.

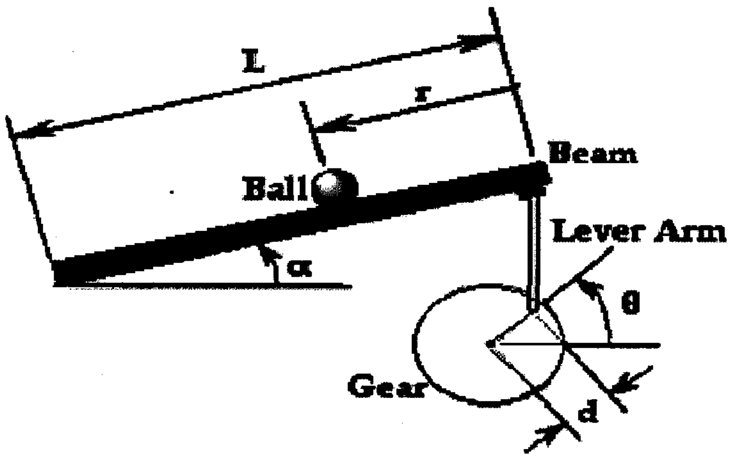
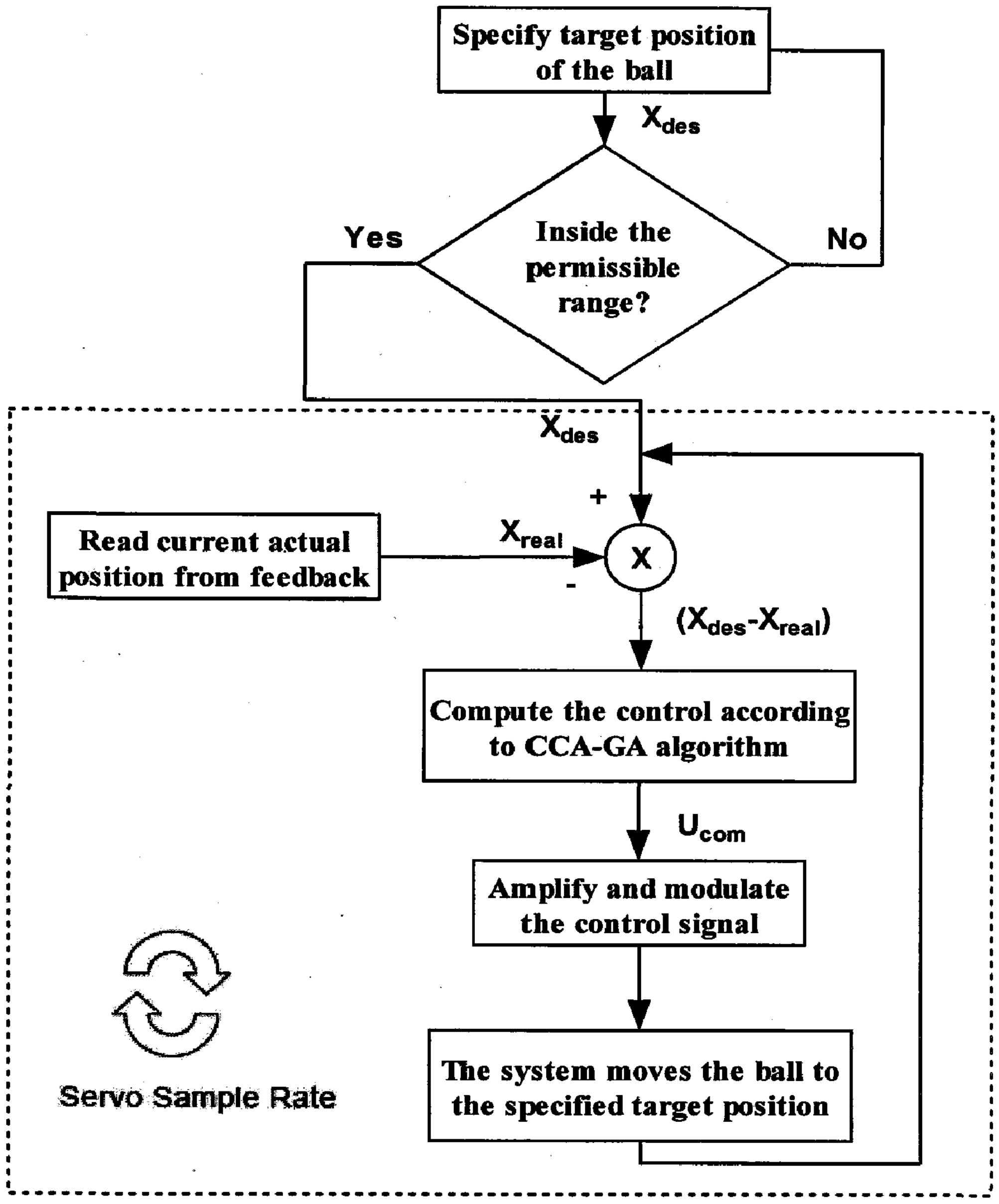


Figure-5.2.4. Ball&Beam Mechanical System



Specify target position of the ball
X_{des}
Inside the permissible range?
Yes
No
X_{des}
+
Read current actual position from feedback
X_{real}
-
X
$(X_{des}-X_{real})$
Compute the control according to CCA-GA algorithm
U_{com}
Amplify and modulate the control signal
The system moves the ball to the specified target position
Servo Sample Rate

Figure-5.2.3 Flowchart of the control program

- Gear ratio is 4.28:1 (107:25)
- Let the angle between the line that connects the joint of the lever arm with the center of the gear, and the horizontal line be θ (there should be some boundaries on its range so that it can reach the safe maximum and minimum limits); the distance between the center of the gear and the joint of the lever arm be *d*, and the length of the beam be *L*. Then the beam angle α can be expressed in terms of the rotation angle of the gear θ according to the following equation:

$$\alpha = \frac{d}{L}\theta$$

In turn, as it has just been noted above, the angle θ is connected with the rotational angle of motor shaft through reduction gear ratio n=4.28.

The controller design task is to keep the position of the ball r equal to the specified target position by properly manipulating the gear angle θ.

- The dynamics of the ball is subjected to the gravity, inertial and centrifugal forces. The ball linear acceleration along the beam is given by the following simple equation:

$$\left(\frac{J}{R^2}+m\right)r + mg\sin\alpha - m\dot{r}(\dot{\alpha})^2 = 0$$

Where

g	is the gravitational acceleration
m	is the mass of the ball
J	is the ball moment of inertia
r	is the position of the ball along the beam
R	is the radius of the ball

Here it is assumes that the ball rolls without slipping and the friction between the beam and ball is negligible.

5.3. Two Link Rigid-Flexible Manipulator Dynamics

The Flexible manipulator used for simulation purpose is shown in Figure-5.3.1. The arm used for simulation is a direct drive planar chain with two revolute joints and two links, the second of which the forearm is very flexible. A nonlinear dynamic model of the two link rigid flexible arm has been derived following a Lagrangian approach. Small deformations are assumed for the forearm, leading to a linear dynamics of the flexible part, so that the main nonlinearities in the model arise from the rigid body interactions between the two links. To compute the low frequency modes, the forearm link is considered as an Euler – Bernoulli beam of length *l2* and length of the rigid link as *l1*, uniform density ρ, and constant elastic properties *EI*. With reference to Figure-5.3.1, for a link point $x \in [0, l2]$; $w(x, t)$ is the bending deflection measured from the axis passing through the center of mass of the forearm. Accordingly, θ*2* is the angle between this same axis and the first rigid links and θ*1* is the angle of first rigid link with X coordinate as shown in Figure-5.3.1.

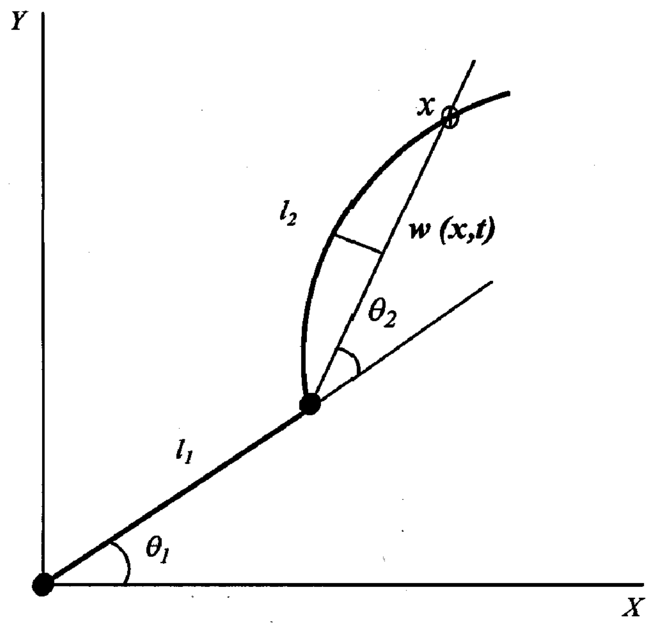


Figure-5.3.1. Schematic diagram of rigid-flexible manipulator and its variables

Considering the slewing nature of the forearm, deformation, Eigen functions have been obtained in Ref. 41. The second joint moment of inertia *J02* and the payload mass *MP* and the moment of inertia *JP* are explicitly included in the boundary conditions associated to the partial differential equation for *w(x, t)*. An approximation of order n of the deflection can be expressed as,

$$w(x,t) = \sum_{i=1}^{n} \phi_i(x)\delta_i(t) \tag{5.3.1}$$

with the time varying coordinates $\delta_i(t)$ associated to the mode shapes,

$$\phi_i(x) = C_{1,i}\sin(\beta_i x) + C_{2,i}\cos(\beta_i x) + C_{3,i}\sinh(\beta_i x) + C_{4,i}\cosh(\beta_i x) \tag{5.3.2}$$

The coefficients C's are determined, up to a scaling factor which is chosen through normalization, from the imposed boundary conditions. The values β_i are numerically obtained as the first *n* roots of the characteristic equation [20].

$$c \cdot sh - s \cdot ch - \frac{2M_p}{\rho}\beta_i s \cdot sh - \frac{2J_p}{\rho}\beta_i^3 c \cdot ch - \frac{J_{02}}{\rho}\beta_i^3(1 + c \cdot ch)$$
$$-\frac{M_p}{\rho^2}\beta_i^4(J_{02} + J_p)(c \cdot sh - s \cdot ch) + \frac{J_{02}J_p}{\rho^2}\beta_i^6(c \cdot sh + s \cdot ch) \qquad (5.3.3)$$
$$-\frac{J_{02}J_pM_p}{\rho^3}\beta_i^7(1 - c \cdot ch) = 0$$

where $s = \sin(\beta_i l_2)$, $c = \cos(\beta_i l_2)$, $sh = \sinh(\beta_i l_2)$, and $ch = \cosh(\beta_i l_2)$. The natural angular frequencies w_i of the flexible link are related to β_i through $\beta_i^4 = \rho w_i^2 / EI$ .Starting from the analysis, the lagrangian dynamics of the two link robot is derived in the standard way as

$$B(q)\ddot{q} + c(q,\dot{q}) + Kq + D\dot{q} = Gu \qquad (5.3.4)$$

where $q = (\theta_1, \theta_2, \delta_1, ...\delta_n) \in R^{n+2}$, and with positive definite symmetric inertia matrix *B*, coriolis and centripetal terms *c*, and elasticity matrix *K*. Joint viscous friction and modal damping coefficients are arranged on the diagonal of *D*, while input matrix *G*, transforms motor torques u into generalized forces performing work on q. To express the single dynamic terms in (4), the following notation will be used

$$v_i = \rho \int_0^{l_2} \phi_i(x)dx, \; i = 1, ...n, \qquad (5.3.5)$$

and,

$$\phi_{ie} = \phi_i(x)\big|_{x=l_2}, \quad \phi'_{i0} = \frac{\partial \phi_i(x)}{\partial(x)}\big|_{x=0}, \quad i = 1, ..., n \qquad (5.3.6)$$

Since the eigen functions $\phi_i(x)$ automatically satisfy proper orthonormality conditions, relevant simplifications arise in the dynamic model. For control design purposes, we will consider only two modes of deformation, so that

$q = (\theta_1, \theta_2, \delta_1, \delta_2) \in R^4$. Neglecting the Kinetic energy of the system which is quadratic or higher order in the deformation variables δ_i yields the inertia matrix,

$$B(q) = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{12} & b_{22} & 0 & 0 \\ b_{13} & 0 & 1 & 0 \\ b_{14} & 0 & 0 & 1 \end{bmatrix} \qquad (5.1.7)$$

with elements,

$$\begin{aligned}
b_{11} &= J_{1Tot} + J_{1Tot} + 2h_3 \cos\theta_2 - 2(h_1\delta_1 + h_2\delta_2)\sin\theta_2 \\
b_{12} &= J_{2Tot} + h_3 \cos\theta_2 - (h_1\delta_1 + h_2\delta_2)\sin\theta_2 \\
b_{13} &= h_1 \cos\theta_2 \\
b_{14} &= h_2 \cos\theta_2 \\
b_{22} &= J_{2Tot}
\end{aligned}$$

in which,

$$\begin{aligned}
h_i &= (v_i + M_P\phi_{ie})l_2 \quad , i=1,\ 2 \\
h_3 &= (M_2 d_2 + M_P l_2)l_1 \\
J_{1Tot} &= J_{01} + J_1 + M_1 d_1^2 + (M_2 + M_{02} + M_P)l_1^{\ 2} \\
J_{2Tot} &= J_{02} + J_2 + M_2 d_2^2 + J_P + M_P l_2^{\ 2}
\end{aligned}$$

where, in addition to previous definitions, l_i is the length of link i, M_i and M_{0i} are the mass of the link i and of joint i, J_i and J_{0i} are their moments of inertia referred to the respective center of mass, and d_i is the distance of the center of mass of link i from joint axis i. The components of the coriolis and centripetal force vector $c(q,\dot{q})$ are,

$$c_1 = -(2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2^{\,2}) + [h_3 \sin\theta_2 - (h_1\delta_1 + h_2\delta_2)\cos\theta_2]$$
$$- 2(\dot{\theta}_1 + \dot{\theta}_2)(h_1\dot{\delta}_1 + h_2\dot{\delta}_2)\sin\theta_2]$$
$$c_2 = \dot{\theta}_1^{\,2}[h_3 \sin\theta_2 + (h_1\delta_1 + h_2\delta_2)\cos\theta_2]$$
$$c_3 = \dot{\theta}_1^{\,2} h_1 \sin\theta_2$$
$$c_4 = \dot{\theta}_1^{\,2} h_2 \sin\theta_2 \tag{5.3.8}$$

The input matrix takes the form,

$$G = \begin{bmatrix} I \\ G_\delta \end{bmatrix}, G_\delta = \begin{bmatrix} 0 & \phi'_{10} \\ 0 & \phi'_{20} \end{bmatrix} \tag{5.3.9}$$

while the elasticity matrix becomes,

$$K = \begin{bmatrix} 0 & 0 \\ 0 & K_\delta \end{bmatrix}, K_\delta = \begin{bmatrix} \omega_1^2 & 0 \\ 0 & \omega_2^2 \end{bmatrix} \tag{5.3.10}$$

Also, modal damping is included by specifying,

$$D = \begin{bmatrix} 0 & 0 \\ 0 & D_\delta \end{bmatrix}, D_\delta = \begin{bmatrix} 2\zeta_1\omega_1 & 0 \\ 0 & 2\zeta_2\omega_2 \end{bmatrix} \tag{5.3.11}$$

where the first zeros on the diagonal of D are due to the fact that the low friction at the joints is neglected. The above explicit expressions can be generalized to the case of n>2 modes in a straightforward way. And the tip deflection of the forearm is written as,

$$y_{tip} = \left(\frac{\phi_{1e}}{l_2} - \phi'_{10}\right)\delta_1 + \left(\frac{\phi_{2e}}{l_2} - \phi'_{20}\right)\delta_2 \tag{5.3.12}$$

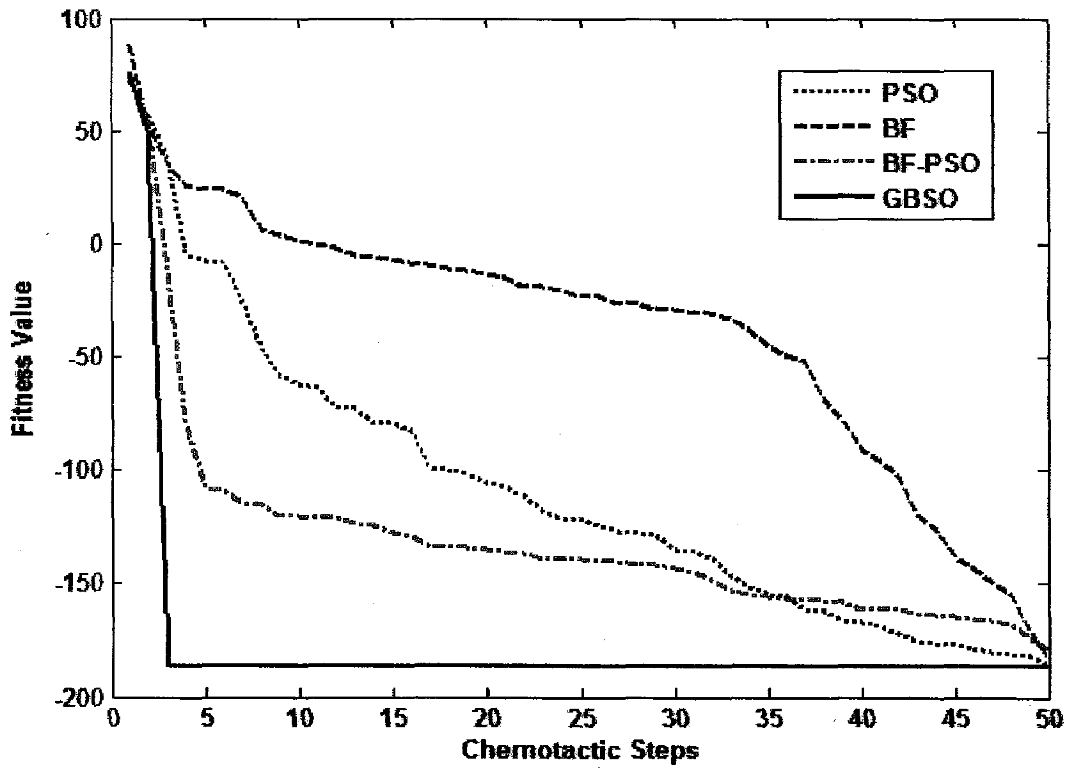


Figure-6.1.3 Performance of GBSO in comparison to others optimization techniques

Table-6.1.1 Performance of GBSO in comparison to other algorithms

	θ_1	θ_2	Optimal objective function	Mean (Std. Deviation)
PSO	-1.4172	-0.6480	-186.4206	-177.0016 (15.7282)
BF	-1.4371	-0.8213	-185.4273	-182.6566 (2.9025)
BF-PSO	-1.4164	-0.8354	-184.0926	-183.1964 (1.8568)
GBSO	-1.4251	-0.8003	-186.7309	-186.7309 (0)

Figure-6.1.3 shows the path of objective function achieving the optimal solution of all algorithms. The convergence rate as shown in Figure-6.1.3 is very high using GBSO in comparison to other optimization techniques. Twenty independent runs of the four algorithms were carried out on each problem and average of the best-of-run solutions and standard deviations were noted. The performance of all algorithms is tabulated in Table-6.1.1. The two variables are adjusted iteratively to achieve the optimal objective value with minimum standard deviation.

Figure-6.1.4 illustrates the Rosenbrock function behavior given by Eqs.-6.1.3 followed by the comparison of GBSO with other techniques in log scale in Figure-6.1.5.

$$F_{Rosenbrock} = 100(\theta_1^2 - \theta_2)^2 + (1-\theta_1)^2 \qquad (6.1.3)$$

The fitness value reaches the optimum in very less chemotactic steps whereas the other techniques settle at sub-optimal region. The comparison is shown in Table-6.1.2.

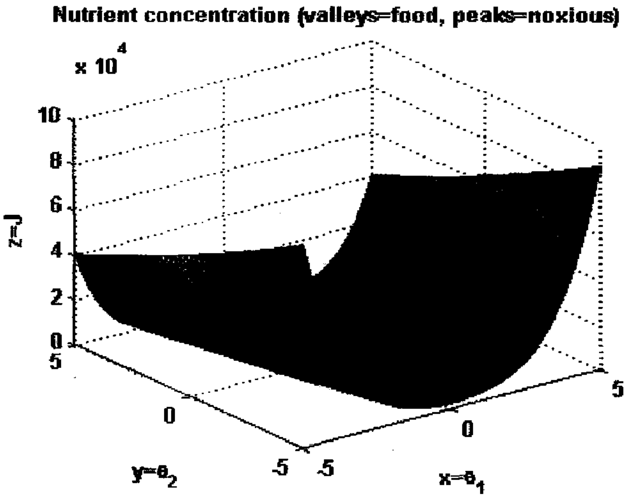


Figure-6.1.4 Rosenbrock function landscape

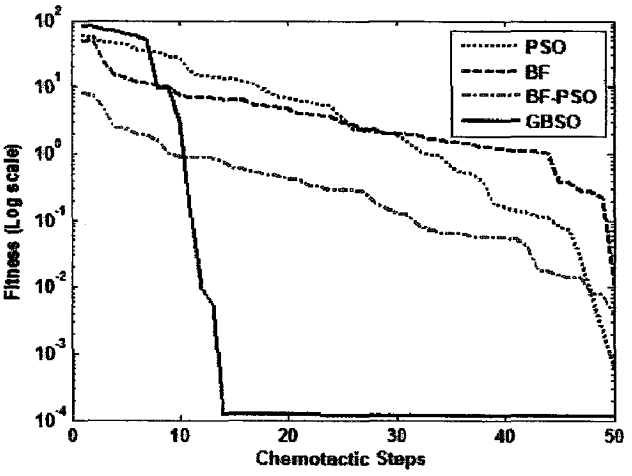


Figure-6.1.5. Performance of GBSO in comparison to others optimization techniques

Table-6.1.2. Performance of Rosenbrock function

	θ_1	θ_2	Optimal objective function	Mean (Std. Deviation)
PSO	0.9731	0.9519	0.0032	0.0182 (0.0176)
BF	0.9916	0.8841	0.0253	0.0065 (0.0047)
BF-PSO	1.0176	1.0408	0.0011	0.0027 (0.0025)
GBSO	1.0120	1.0242	1.4413e-004	1.947e-04 (3.188e-04)

Figure-6.1.6 illustrates the Rastrigin function behavior given by Eqs.-6.1.4 followed by the comparison of GBSO with other techniques in Figure-6.1.7. The comparison is shown in Table-6.1.3.

$$F_{Rastrigin} = \sum_{i=1}^{n} [\theta_i^2 - 10\cos(2\pi\theta_i) + 10] \qquad (6.1.4)$$

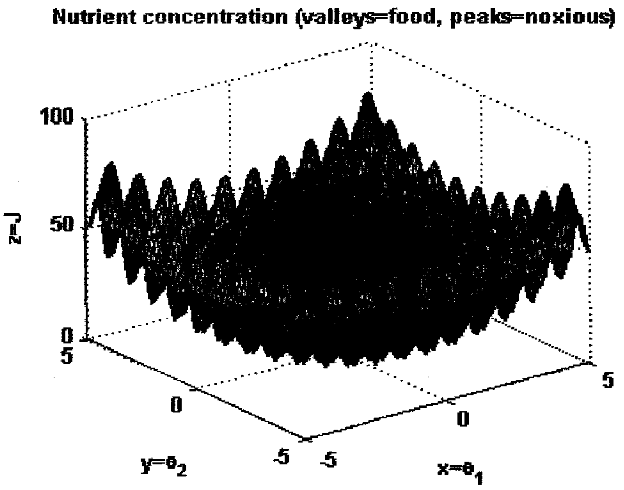


Figure-6.1.6 Rastrigin function landscape

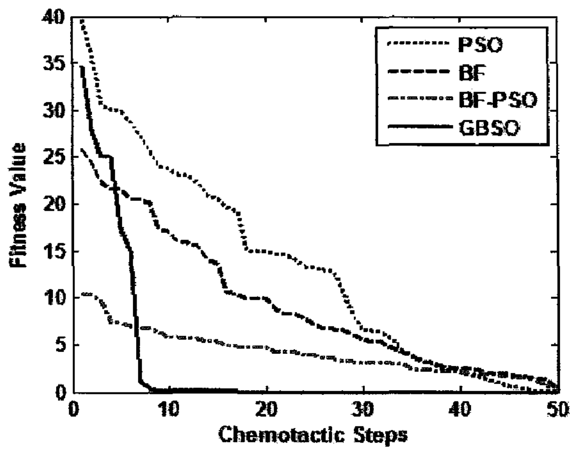


Figure-6.1.7 Performance of GBSO in comparison to others optimization techniques

Figure-6.1.8 illustrates the Griewank function behavior defined by Eqs.-6.1.5. The comparison between the optimization is shown in Figure-6.1.9. Statistical data is given in Table-6.1.4.

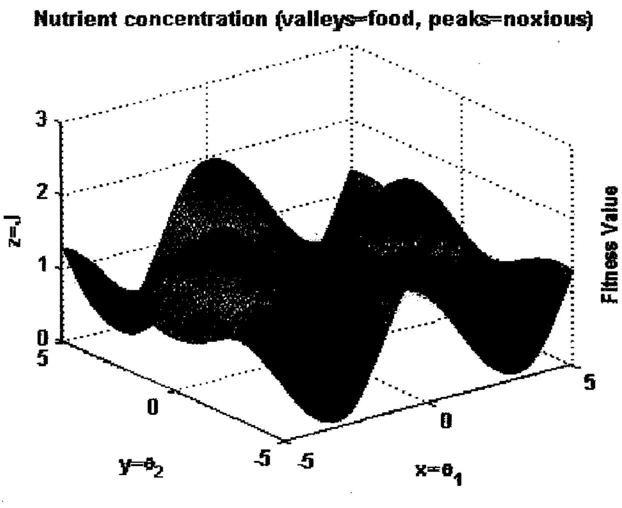


Figure-6.1.8 Griewank function landscape

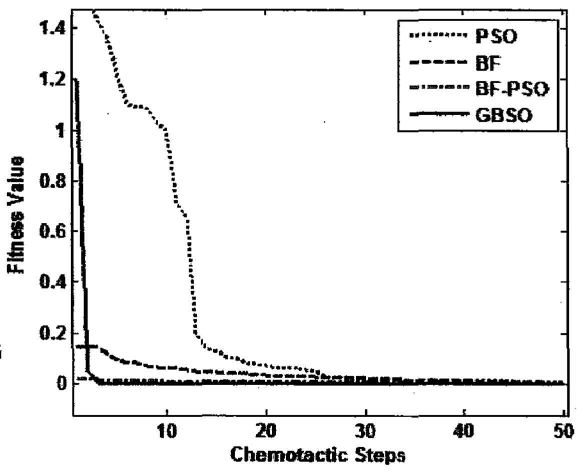


Figure-6.1.9 Performance of GBSO in comparison to others optimization techniques

$$F_{Griewank} = \frac{1}{4000}\sum_{i=1}^{n}\theta_i^2 - \prod_{i=1}^{n}\cos\left(\frac{\theta_i}{\sqrt{i}}\right) + 1 \tag{6.1.5}$$

Table-6.1.3 Performance of Rastrigin function

	θ_1	θ_2	Optimal objective function	Mean (Std. Deviation)
PSO	-0.0180	0.1016	0.0250	0.0563 (0.0935)
BF	-0.0286	0.0115	0.1879	0.3428 (0.2544)
BF-PSO	-0.0455	0.0207	0.0611	0.4809 (0.0966)
GBSO	-0.7620e-9	0.3871e-9	0	0 (0)

Table-6.1.4 Performance of Griewank function

	θ_1	θ_2	Optimal objective function	Mean (Std. Deviation)
PSO	3.1397	-4.4366	0.0074	0.0057 (0.0031)
BF	0.0156	0.0226	2.4842e-004	2.0809e-4 (2.1267e-4)
BF-PSO	0.0002	0.0505	6.3654e-004	6.4676e-5 (7.4995e-7)
GBSO	0	0	0	0 (0)

6.2 Inverted Pendulum

The open loop step response of a pendulum angle is shown in Figure-6.2.1. It can be seen that a small force or disturbance acting on the cart sets the pendulum angle at 90 degrees therefore, to improve the dynamics of the system; some controller has to be designed. The block diagram of PD-PI control system for Inverted Pendulum is shown in Figure-6.2.2.

The performance of the PD-PI control can be judged by the value of its parameters. Figure-6.2.2 shows the implementation of optimization algorithms which have been used to estimate the parameters. Empirical tuning methods cannot be applied to the system under consideration as the conditions (i.e. open loop stability, S-shaped response etc.) are not being satisfied. The performance with all the optimization algorithms has been analyzed on the basis of ITAE (Integral Time Absolute Error). The objective of the controller is to maintain the upward position of the pendulum for any external disturbance.

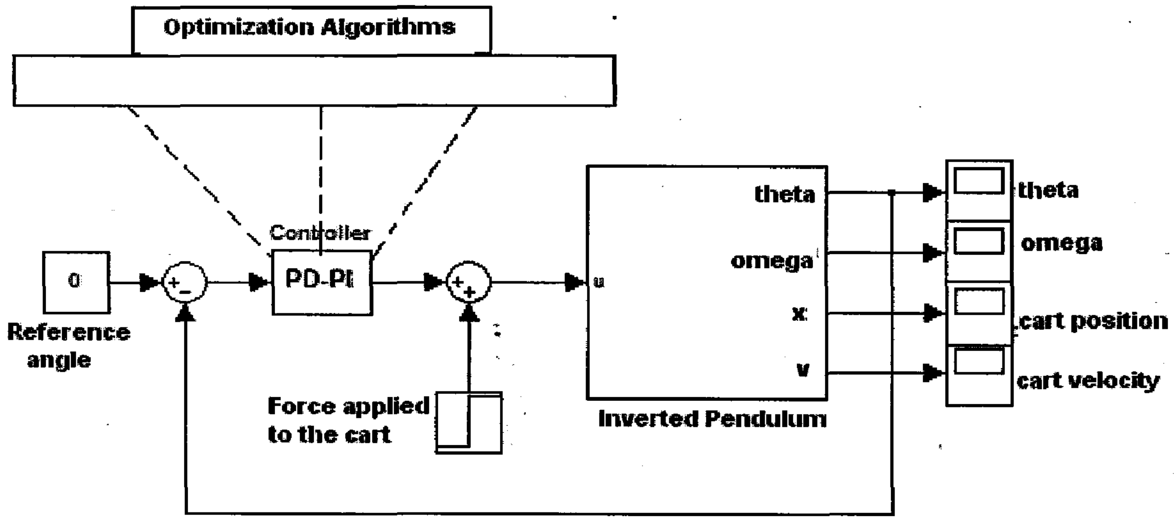


Figure-6.2.2 Block diagram of PD-PI controller and its implementation

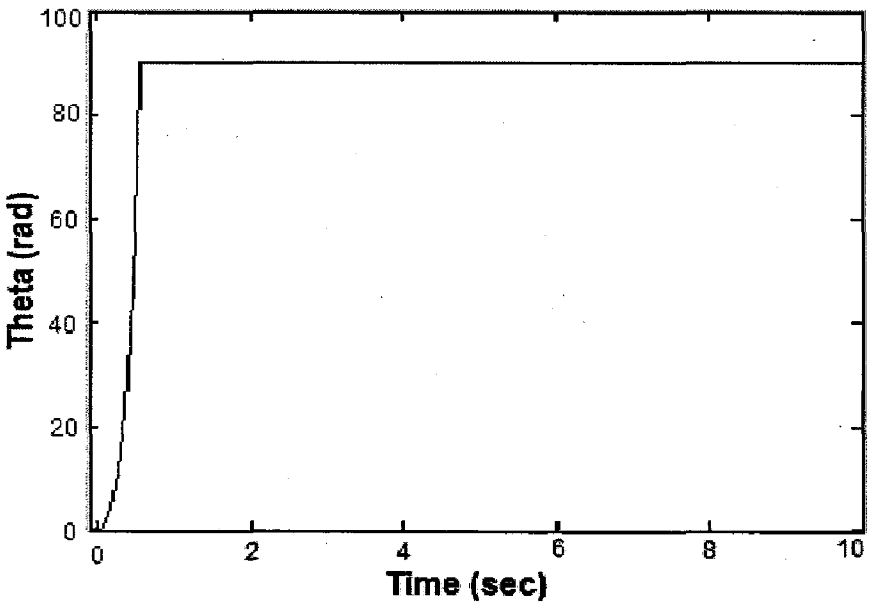


Figure-6.2.1 Open loop response of pendulum angle

Table 6.2.1 Closed loop data of Inverted Pendulum using Optimization Techniques

Optimization Techniques/Controller parameters	K_p	K_i	K_d	ITAE
GA	23.0673	37.5323	2.5116	0.01236
BF-GA	90.6809	104.7037	40.0440	0.00023
BF	25.4356	45.6982	2.6580	0.00745
BF-PSO	32.6711	50.9613	4.6441	0.00624
PSO	30.7611	49.1693	4.4641	0.00675
GBSO	150.5704	250.3077	70.4004	0.00011

Numerical values of constant parameters of the controller and performance index using different optimization techniques are tabulated in Table 6.2.1. It can be seen that ITAE reduces to an optimal value with the developed algorithms.

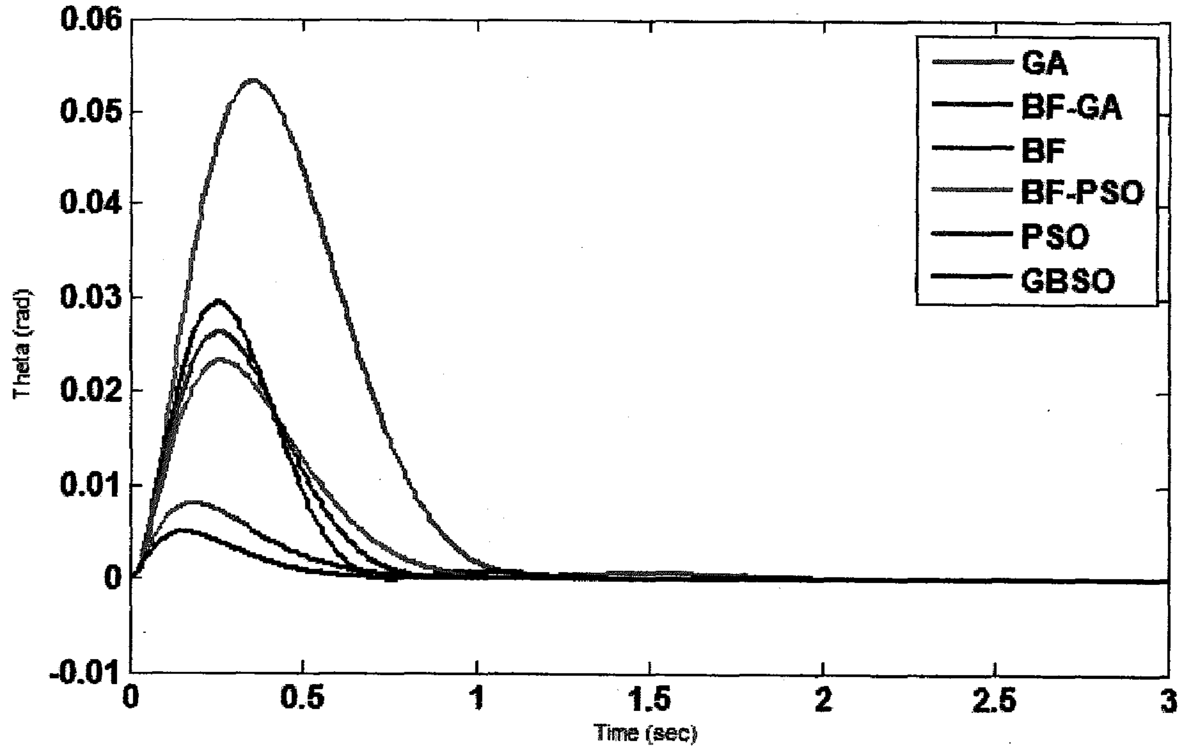


Figure-6.2.3 Closed loop response of controller with ITAE as performance index

For the system under consideration, the simulation results with GBSO techniques prove to be more effective than with other optimization algorithms. In GAs, the limits defined by the number of parameters gives the search region while in PSO, the search region is independent of the number of parameters, given by the distance between the randomly selected initial position and the position corresponding to optimal fitness value. The speed of computation is determined by the velocity initializing the PSO algorithm with which it reaches to the best solution. It is also observed that the speed of computation in PSO is very less in comparison to GAs and BG.

6.3 Ball and Beam System

Ball and Beam represents a Single Input Single Output (SISO) system where *X(s)* and θ(s) are the Laplace representation of the output (position of the ball on the beam) and input (beam angle) of Ball and Beam. The open loop transfer function of the system mechanics can be approximated by double integrator. The closed loop model is shown in Figure-6.3.1.

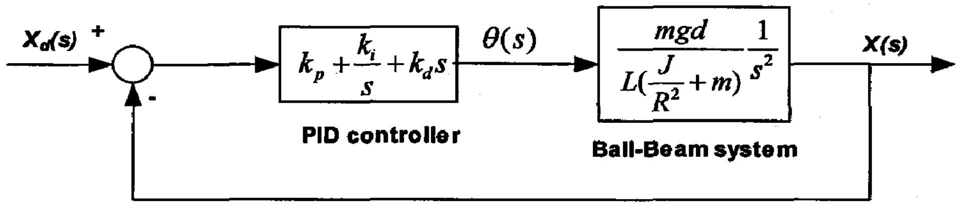

$X_d(s)$
+
-
$k_p + \frac{k_i}{s} + k_d s$
PID controller
$\theta(s)$
$\frac{mgd}{L(\frac{J}{R^2}+m)} \frac{1}{s^2}$
Ball-Beam system
$X(s)$

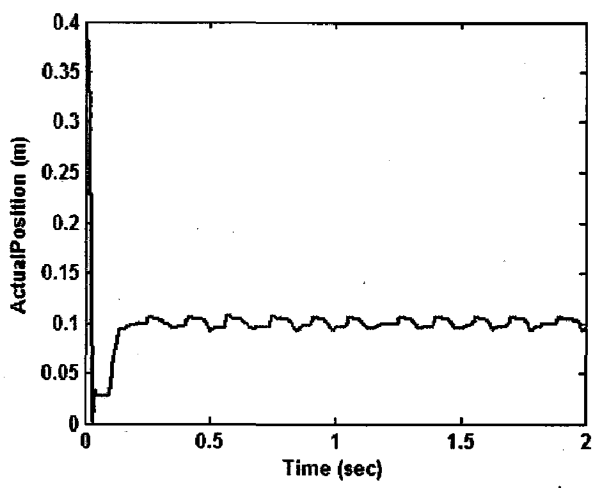
Figure-6.3.1 Block diagram of Ball-Beam system with a PID controller

Where

mass of ball, *m*	=	0.111
ball moment of inertia, *J*	=	2**m***R*^2/5
gravitational acceleration, *g*	=	-9.8
position of ball along the beam, *d*	=	0.04
radius of the ball, *R*	=	0.01

The real nonlinear system is approximated to double differentiator as shown in Figure-6.3.1. MATLAB simulations were carried first using Hybrid BF and GBSO algorithms to determine the optimal set of PID parameters. This experiment is mainly performed to determine the superiority of the previously developed hybrid BF-GA algorithm. The set of PID parameters are then implemented real time on experimental setup discussed above. Performance index for fitness evaluation constitutes summation of settling time and steady state error for a period of 10 sec.

The real time output of the ball-beam system with a step input is shown in Figure-6.3.2. Initially the ball is placed at one end (maximum length 40 cm) of the beam. For the desired position of the ball at 10 cm, the overshoot and settling time is shown in Figure-6.3.2a. For the desired position as 20 cm and 30 cm, the overshoot and settling time is shown in Figure-6.3.2b and Figure-6.3.2c respectively. It is evident from above figures that there is reduction in overshoot and settling time as the desired position approaches near the initial point of the ball (40 cm) for the same set of PID parameters. The response for a desired position having pulse input with duty cycle of 50% and amplitude varying between 20 cm and 0 cm is also studied as shown in Figure-6.3.3. The response has a large overshoot in comparison to step response with the same desired position.



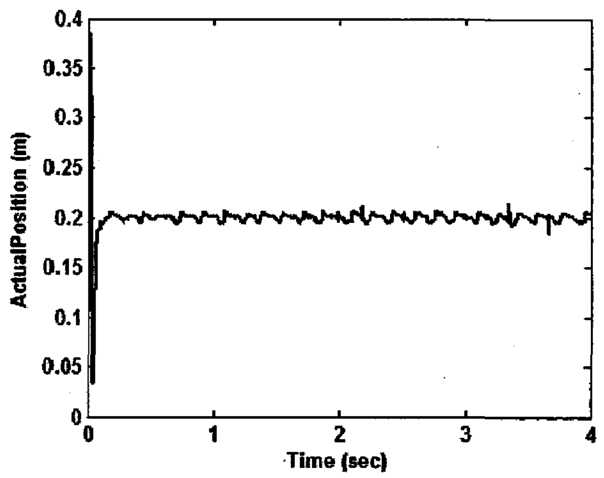
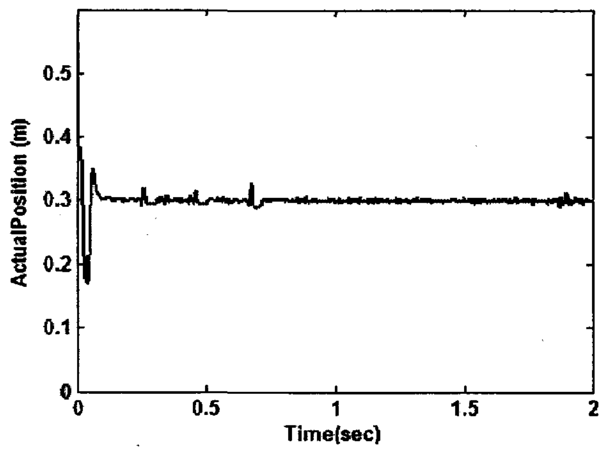


Figure-6.3.2a. Step Response with desired position 10 cm using GBSO algorithm GBSO

Figure-6.3.2b. Step response with desired position 20cm using



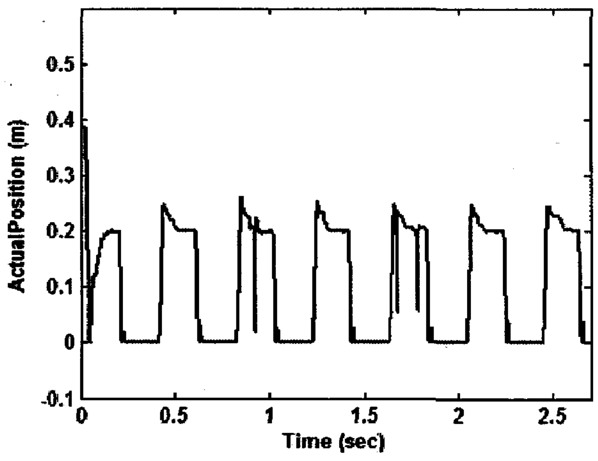


Figure-6.3.2c. Step response with desired with position 30 cm

Figure-6.3.3 Pulse input response duty cycle 50%

The fitness profile of ball and beam system using GBSO is shown in Figur-6.3.4. The comparison of performance using BF-GA and GBSO algorithms are shown in Table-6.3.1. For 80 numbers of decades, the tuning of three parameters towards optimal solution is illustrated in Figure-6.3.5.

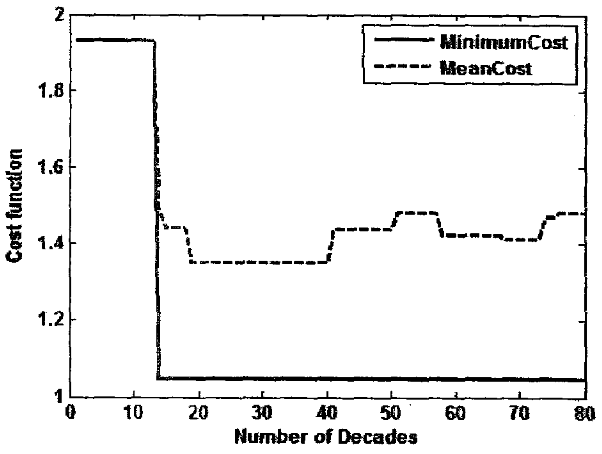


Figure-6.3.4 Fitness profile of Ball-Beam system

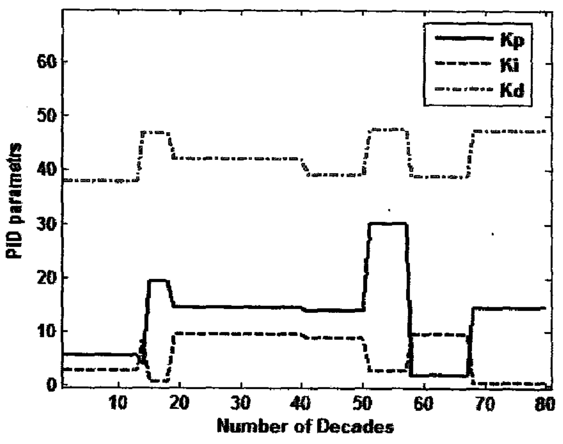


Figure-6.3.5 Tuning of parameters during 80 decades

Table-6.3.1 Performance comparison using different methods

Methods	k_p	k_i	k_d	$Mo(\%)$	Ess	t_s	t_r
BF-GA	9.6960	8.3156	48.3786	1.0030	0.023	0.1086	0.0644
GBSO	14.5911	0.5126	47.3933	0.8721	0	0.1080	0.0637

This section demonstrated the novel hybrid approach consisting BF-GA and GBSO. The improvement is shown in terms of convergence rate of the objective function towards the optimality in comparison to BF-GA for higher dimension. The proposed GBSO algorithm is implemented on a real time ball-beam system supplied by Googol Technology for tuning the PID controller. As evident from the graphical and empirical results, the suggested hybrid system performed well.

6.4 Two Link Rigid-Flexible Robot Manipulator

In this section, various optimization schemes have been used for the optimization of controller parameters shown in Figure-6.4.1. The integral square error (ISE) performance index is used as the objective function "*J*" for optimization

$$J = ISE = \int_0^t e^2(t)dt \text{ , where } e(t)=q_d(t)-q(t) \qquad (6.4.1)$$

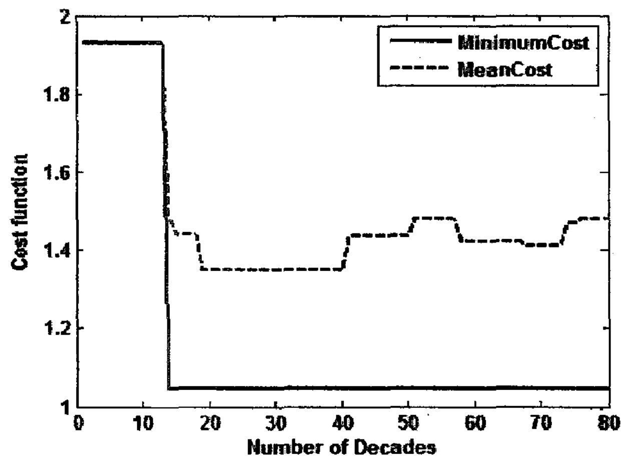


Figure-6.3.4 Fitness profile of Ball-Beam system

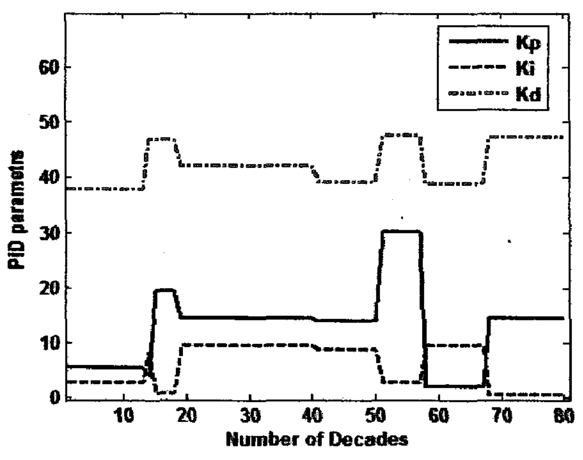


Figure-6.3.5 Tuning of parameters during 80 decades

Table-6.3.1 Performance comparison using different methods

Methods	k_p	k_i	k_d	*Mo(%)*	*Ess*	t_s	t_r
BF-GA	9.6960	8.3156	48.3786	1.0030	0.023	0.1086	0.0644
GBSO	14.5911	0.5126	47.3933	0.8721	0	0.1080	0.0637

This section demonstrated the novel hybrid approach consisting BF-GA and GBSO. The improvement is shown in terms of convergence rate of the objective function towards the optimality in comparison to BF-GA for higher dimension. The proposed GBSO algorithm is implemented on a real time ball-beam system supplied by Googol Technology for tuning the PID controller. As evident from the graphical and empirical results, the suggested hybrid system performed well.

6.4 Two Link Rigid-Flexible Robot Manipulator

In this section, various optimization schemes have been used for the optimization of controller parameters shown in Figure-6.4.1. The integral square error (ISE) performance index is used as the objective function "*J*" for optimization

$$J = ISE = \int_0^t e^2(t)dt \text{ , where } e(t)=q_d(t)-q(t) \qquad (6.4.1)$$

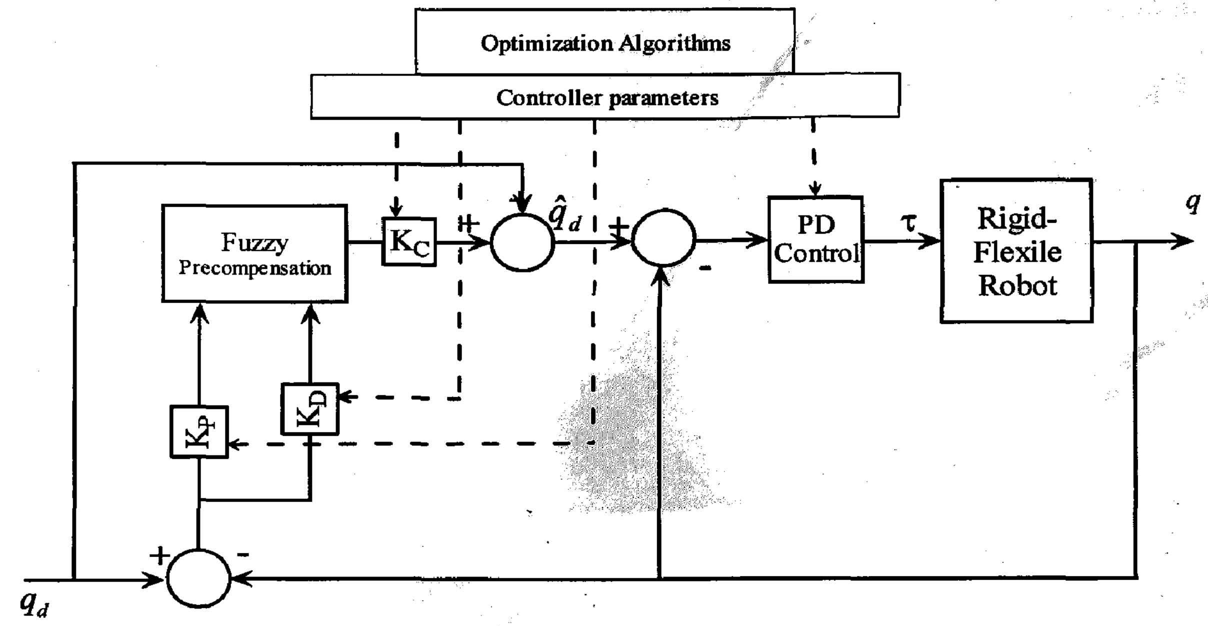


Figure-6.4.1 Fuzzy Precompensated PD Control

Here in simulation for tuning the fuzzy precompensated PD controller, dimension of search space is 7 with 10 numbers of bacteria, 4 number of chemotactic steps, length of the swim and number of reproduction steps as 4, and 2 number of elimination dispersal events is considered. The simulation was run under MATLAB 7.01 with Fuzzy Logic Toolbox 2.2 (R14SP1). Simulink block diagram of two link rigid-flexible manipulator is shown in Figure-6.4.2. To demonstrate the effectiveness of the proposed Fuzzy precompensated PD controller, the dynamic model of two link rigid flexible arm is considered. The fuzzification block converts crisp inputs to appropriate fuzzy sets using the membership funetions as shown in Figure-6.4.3. The knowledge base provides the membership functions and the linguistic control rules. The fuzzy inference engine performs fuzzy reasoning, based on the linguistic control rules, using Zadeh's compositional rule of inference. The defuzzification block generates a crisp control output u (t) by utilizing the centre of gravity method,

$$u(t) = \frac{\sum \mu_i(u)u_i}{\sum \mu_i(u)} \qquad (6.4.2)$$

The rules for our fuzzy precompensator are given in Table-6.4.1. In this case, we used 27 rules. Rules were derived by using a combination of experience, "trial and

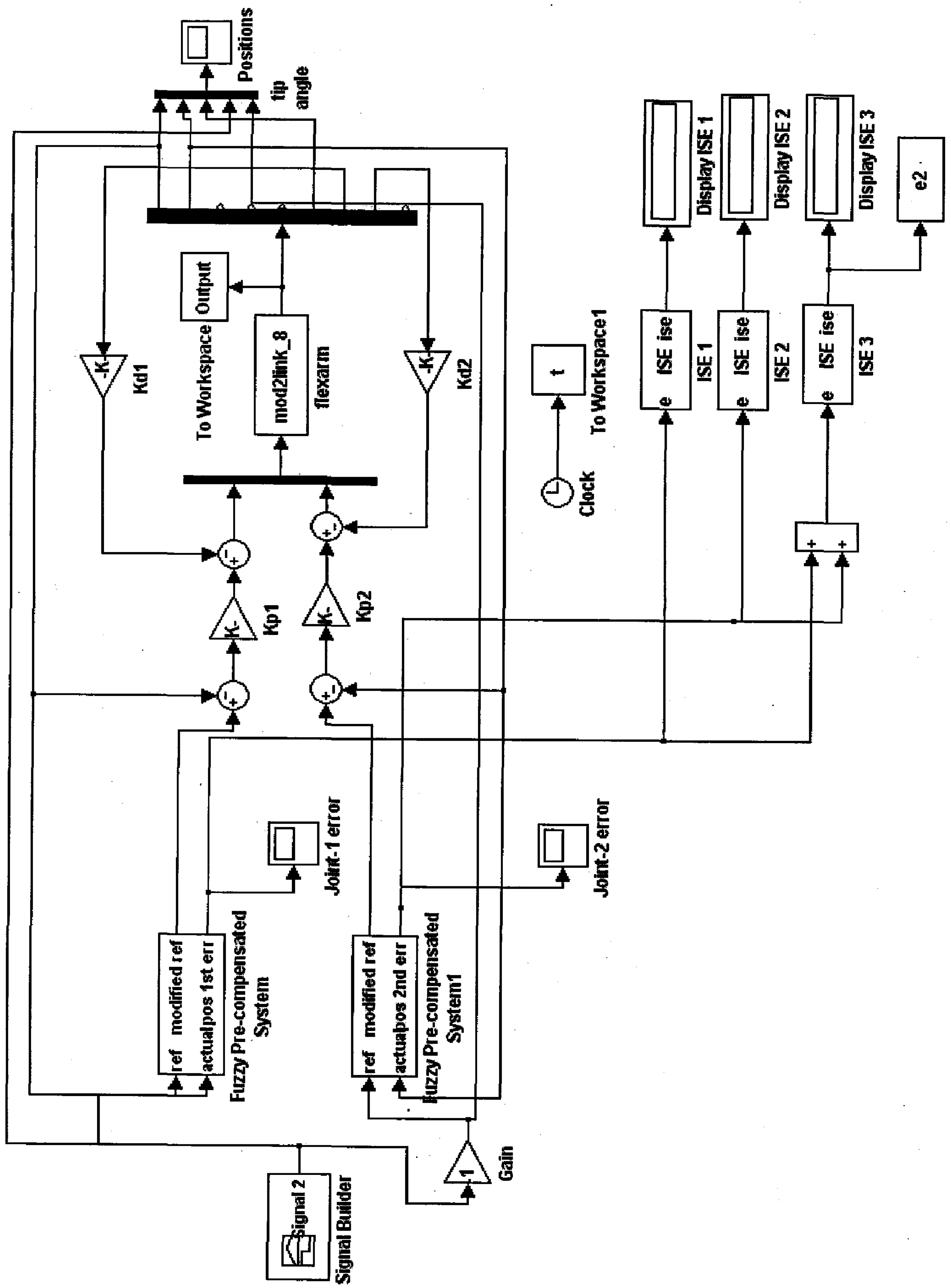


Figure- 6.4.2 Simulink diagram of two link rigid flexible manipulator error", and our knowledge of the response of the system. These are common approaches to the design of fuzzy logic rules, as described in [23]. Figure-6.4.4 represents the output surface of the fuzzy precompensated controller. The flexible robot arm considered is characterized by following data;

$l_1 = 0.3$ m	$h_1 = 0.336$ kg m^2	$\omega_2 = 14.395.2\pi$ rad/sec^{-1}
$l_2 = 0.7$ m	$h_2 = 0.126$ kg m^2	$\zeta_1 = 0.07$
$J_{ITot} = 0.447$ kg m^2	$h_3 = 0.195$ kg m^2	$\zeta_2 = 0.03$
$J_{ITot} = 0.303$ kg m^2	$\phi'_{10} = 5.74$	$\phi_{1e} = -1.446$ m
$J_{02} = 6.35$ X10^{-4} kg m^2	$\phi'_{20} = 11.64$	$\phi_{2e} = 1.369$ m
$M_P = J_P = 0$	$\omega_1 = 4.16.2\pi$ rad/sec^{-1}	

Figure-6.4.5 shows the desired joint position profiles for the simulation to evaluate the effectiveness of the proposed approach. Figure-6.4.6, 6.4.7, 6.4.8 shows the modified or compensated desired position $\hat{q}_{di}(t)$ for $i=1, 2$ using fuzzy precompensation. Figure-6.4.10, 6.4.11, 6.4.12 shows the Joint position error profiles for the two joints using the proposed bacterial foraging optimized fuzzy precompensated PD controller (BFFPPDC), particle swarm optimized fuzzy precompensated PD controller (PSOFPPDC) and hybrid particle swarm and bacterial foraging optimized FPPDC. The same controllers are compared by optimizing with genetic algorithm optimization and are shown in Figure-6.4.9. Table-6.4.2 constitutes the integral square errors for the two joints.

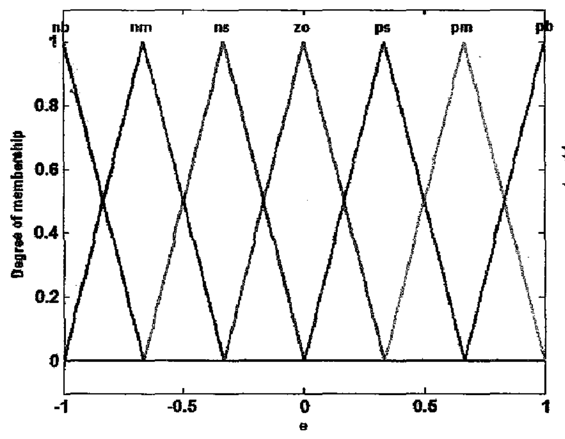


Figure-6.4.3 Fuzzy input-output membership function

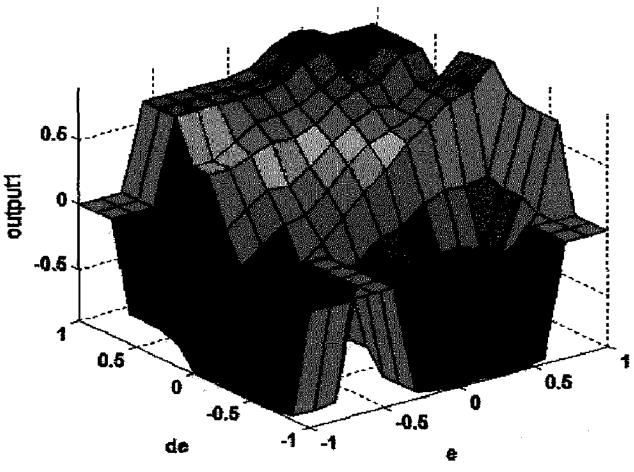


Figure-6.4.4 Output surface of the fuzzy precompensator controller

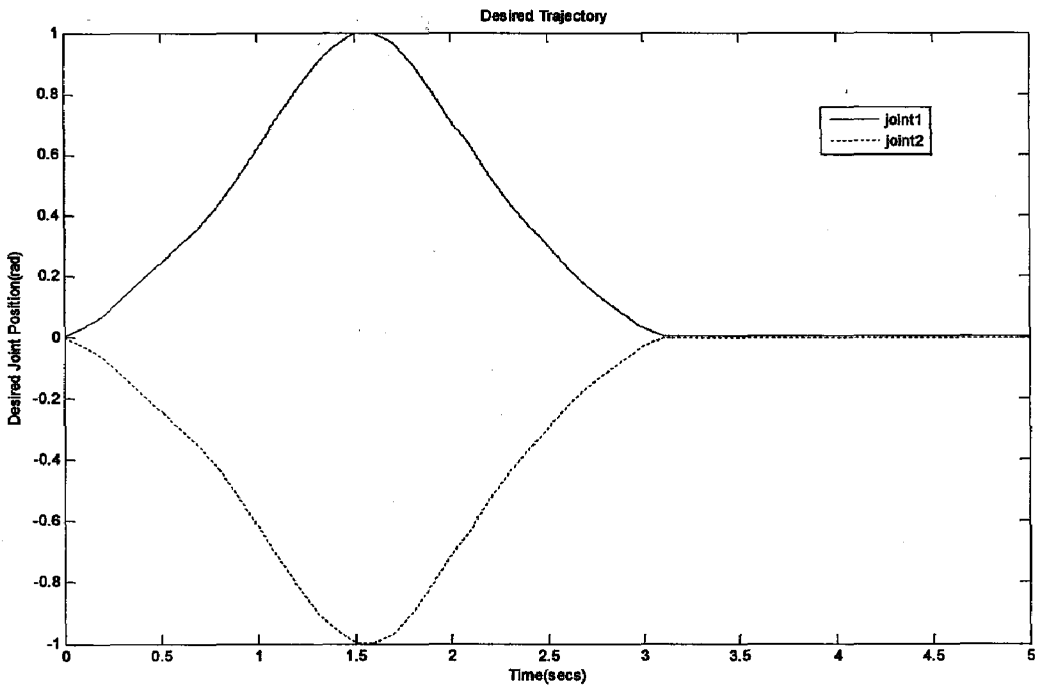


Figure-6.4.5 Desired Position profile

Table-6.4.1 Fuzzy Precompensated PD Controller rules

$\dot{e}$ / e	NB	NM	NS	Z	PS	PM	PB
NB		NB	NB	NB	NM		
NM				NM			
NS				NS	PS		PM
Z	NB	NB	NM	Z	PS	PM	PM
PB	NB	NB	NM	PS	PM	PB	PB
PM			NM	PM		PB	
PS			PM	PB			

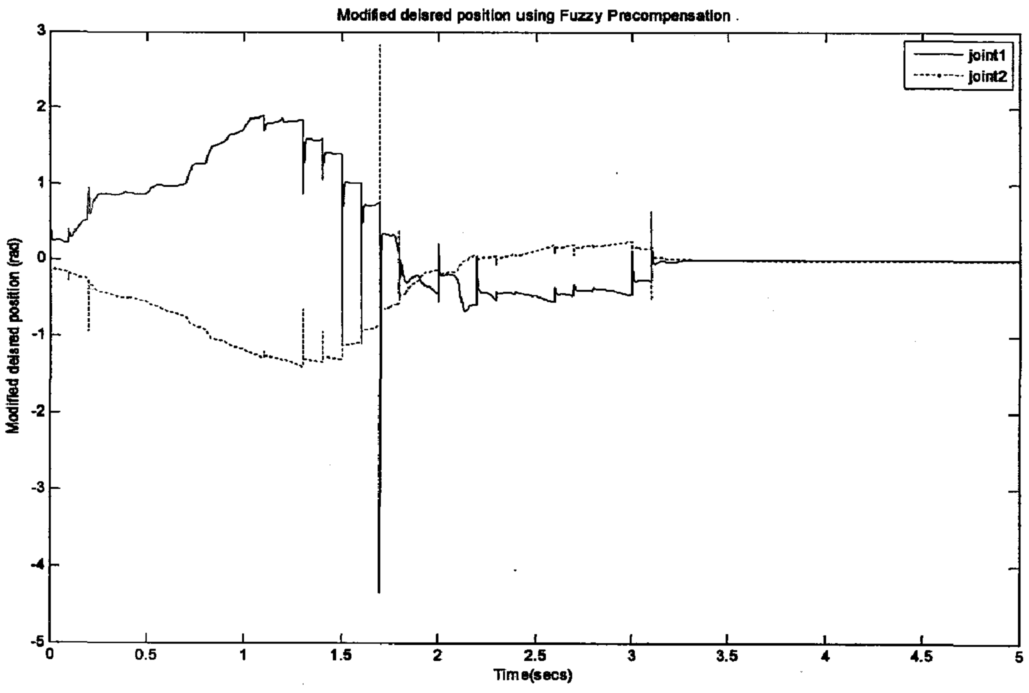


Figure-6.4.6 Modified desired position using fuzzy precompensation using BFFPPDC

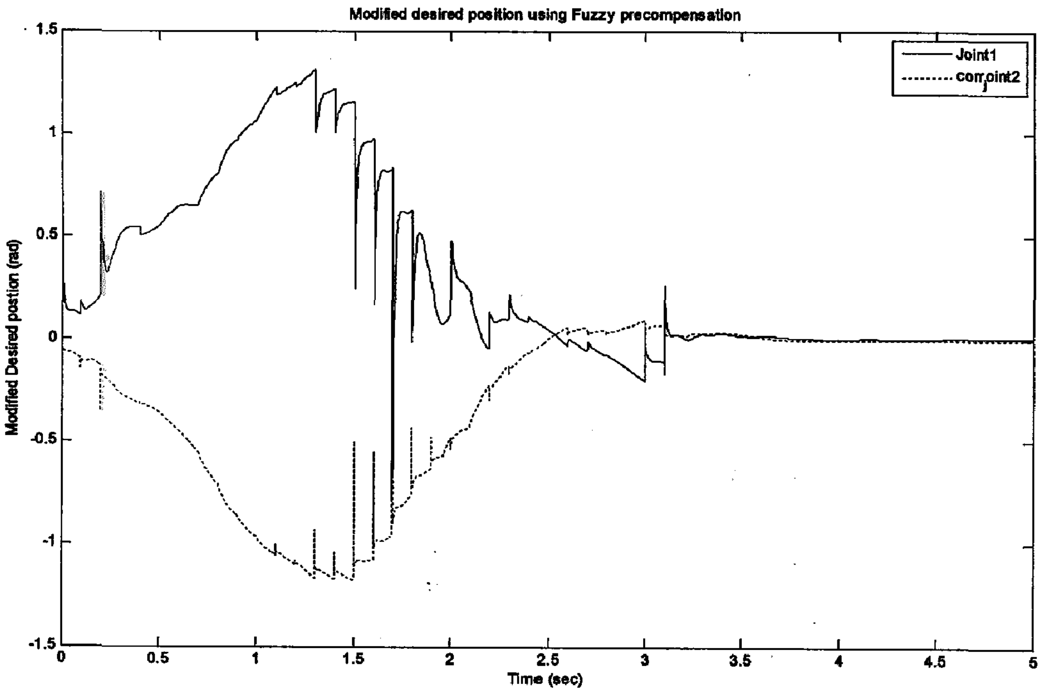


Figure-6.4.7 Modified desired position using fuzzy precompensation using PSOFPPDC

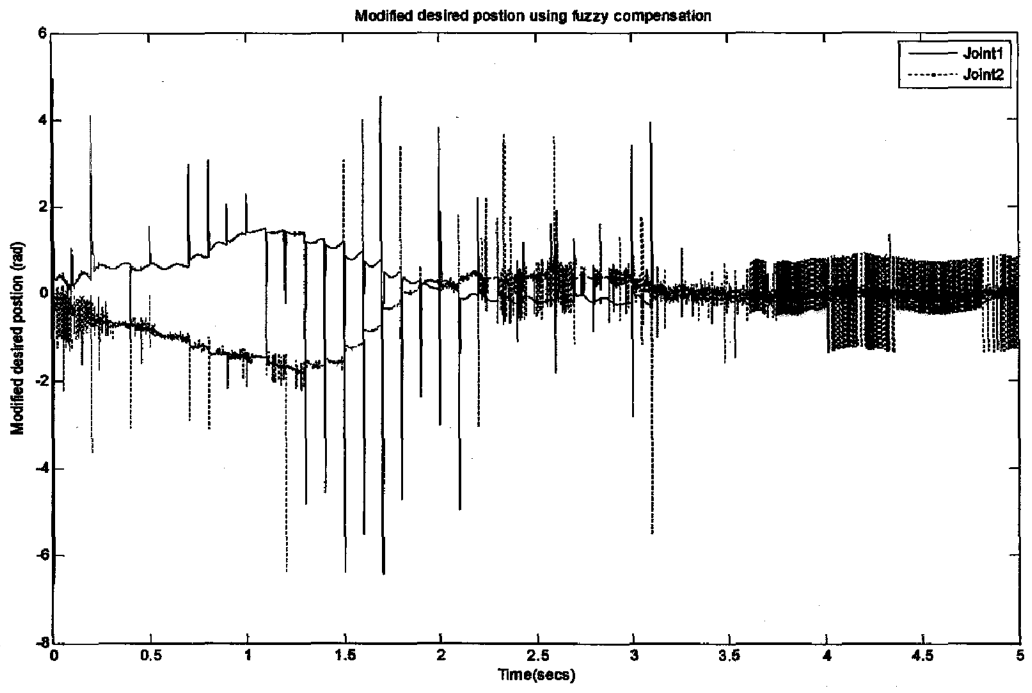


Figure-6.4.8 Modified desired position using fuzzy precompensation using BF-PSOFPPDC

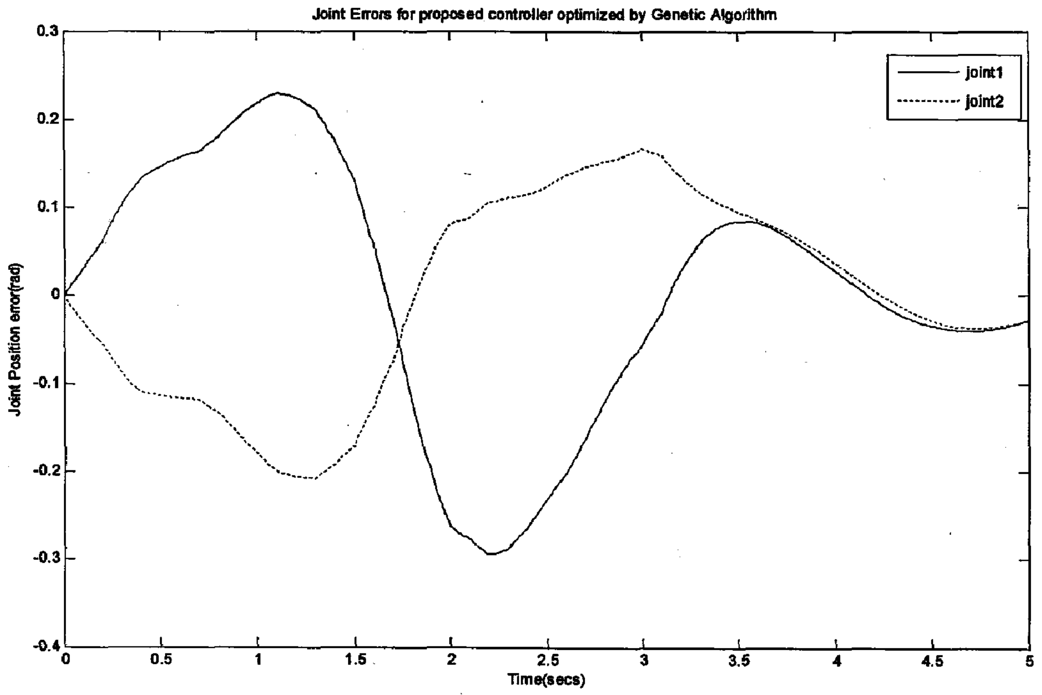


Figure-6.4.9 Error profiles for the two joints using GAFPPDC.

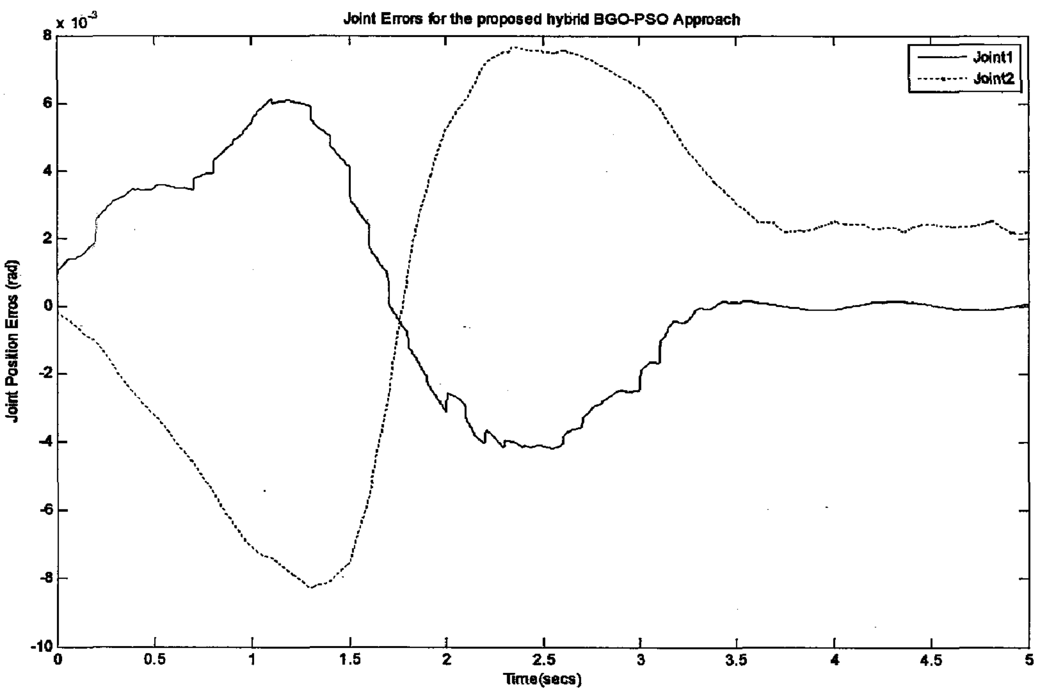


Figure-6.4.12 Error profiles for the two joints using BFO-PSO

Table-6.4.2 Controller constants and integral square errors for various joints

Controller	GAFPPDC	BFFPPDC	PSOFPPDC	BFO-PSO
K_p Fuzzy Precompensator input-1	15.0431	16.1823	8.0092	9.9119
K_d Fuzzy Precompensator input-2	5.2987	1.8166	1.4251	2.0370
K_c Fuzzy Precompensator output	-0.0266	9.6140	4.7827	8.2721
K_{p1}	13.8523	15.4089	10.5684	6.7586
K_{d1}	0.8591	15.7654	3.6909	2.4895
K_{p2}	6.7532	13.3856	6.6176	4.7579
K_{p2}	0.9215	7.5116	1.6961	3.9908
Joint-1 ISE	0.1059	0.0017	0.0063	4.227e-05
Joint-2 ISE	0.0643	0.0009	0.0040	0.0001

In second phase of designing the fuzzy precompensated PD control of two link rigid-flexible manipulator FLC rule base is optimized using characteristic parameters. Here the results are presented using GBSO algorithm. To apply a GBSO algorithm to the design of fuzzy logic controller, an evaluation function is defined to calculate the fitness of a set of parameters. The parameters are passed to the evaluation function, which processes them and returns a value corresponding to how well the parameter performed the task. This function firstly extracts the relevant parameters from the individuals passed in. After performing some error checking, the parameters are used to create a Fuzzy Inference System (FIS) and set the appropriate scaling factors.

A SIMULINK model is then called as shown in Figure-6.4.2 from which a record of the error in the joint position throughout the duration of the simulation is returned. The desired trajectory is shown in Figure-6.4.5. The error in the joint position and the change in of error are scaled by the appropriate gains (these parameters are also set by the optimization algorithm) and the result is clipped so that it lies in the range -1 to 1. These inputs are fed into the FLC and the FLC's output is then scaled by another gain.

To run GBSO, a suitable encoding for each of the parameters and bounds for each of them needs to be decided. For this task the parameters given in Table 6.4.3 are used with the shown ranges and precisions.

Table-6.4.3 Parameters used for encoding

Parameter	**Range**	**Precision**
Number of Membership Functions	3 – 9	2
Membership Function Spacing	0.1 – 1.0	0.01
MF Spacing (Power to be Raised by)	-1 – 1	2
Rule Base Scaling	0.1 – 1.0	0.01
Rule-Base Spacing (Power to be Raised by)	-1 – 1	2
Input Scaling	-50 – 50	0.1
Output Scaling	-50 – 50	0.1
Rule-Base Angle	$0 - 2\pi$	$\pi/512$

The numbers of membership functions are limited to the odd integers inclusive between three and nine. For the spacing parameters, two separate parameters are used.

The first, with the range [0.1 – 1.0], determines the magnitude and the second, which takes only the values -1 or 1, is the power by which the magnitude is to be raised. This determines whether the membership functions compress in the centre or at the extremes. The scaling for the input and output variables is allowed to vary in the range [-50 - 50].

The comparison is presented between two algorithms namely, GA and GBSO. In first part, the rule base is designed using experts' rule base and same input-output membership function. Here the two algorithms optimize the membership functions using characteristic parameters. Figure 6.4.13 shows the rule base optimized using two algorithms.

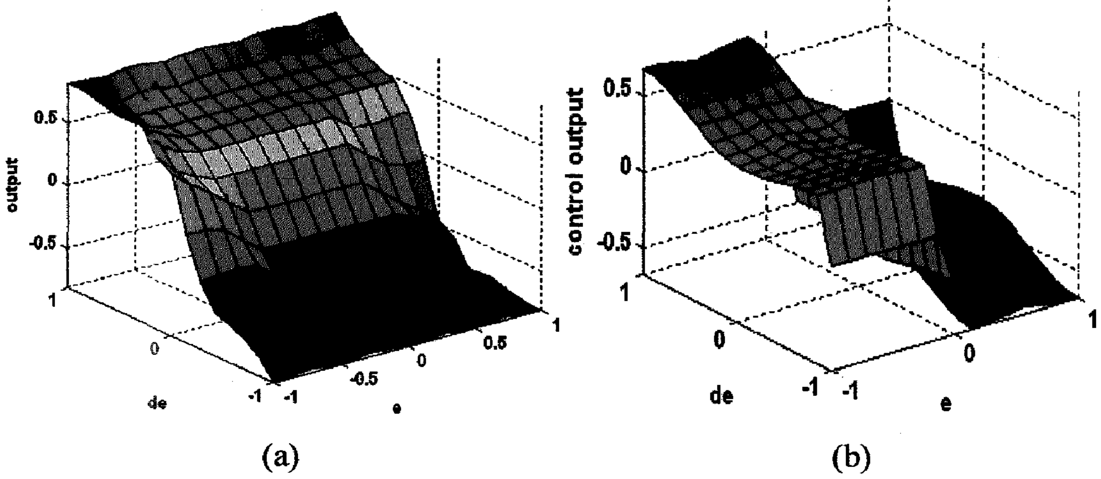


Figure 6.4.13 Output surface of the fuzzy precompensator controller (a) using GA (b) using GBSO

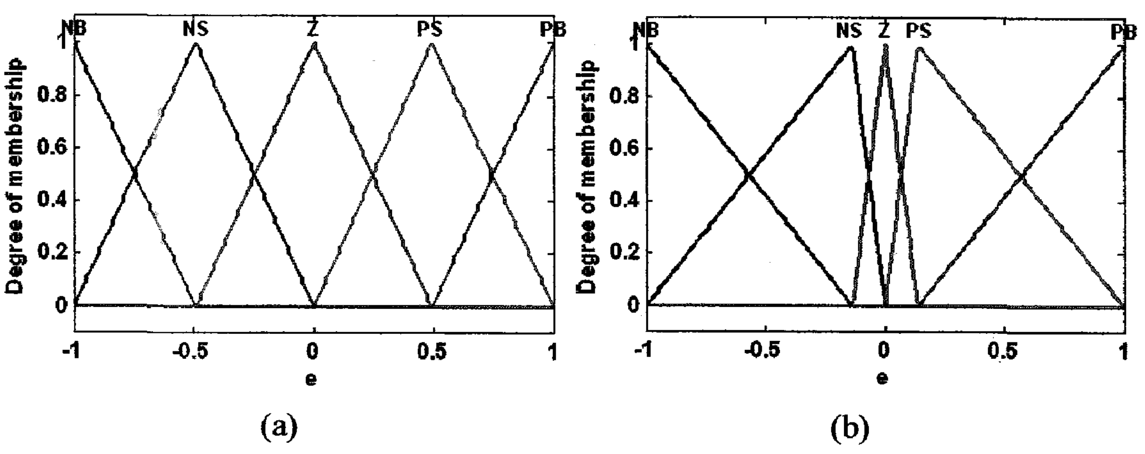
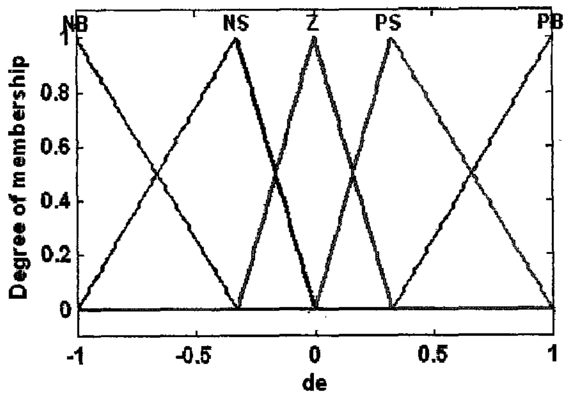
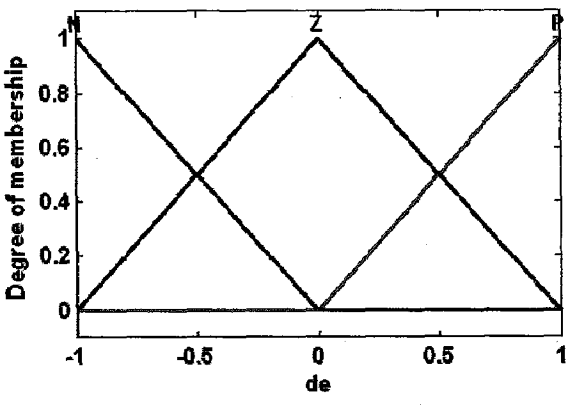


Figure 6.4.14 Input-1 membership function of the fuzzy precompensator controller (a) using GA (b) using GBSO



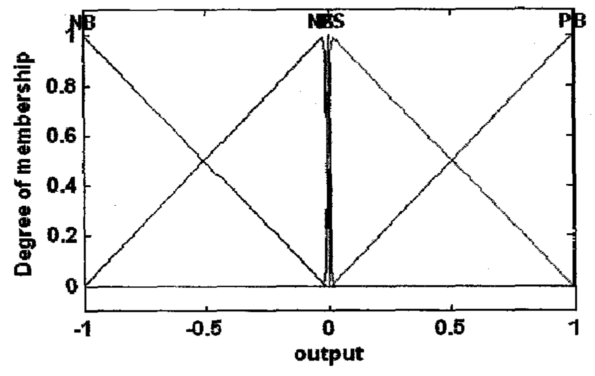
(a)



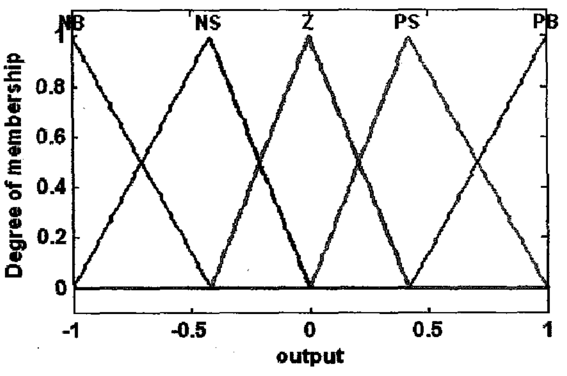
(b)

Figure 6.4.15 Input-2 membership function of the fuzzy precompensator controller (a) using GA (b) using GBSO

Figure 6.4.14, 6.4.15, 6.4.16 shows the input output membership functions. Finally the joint error profile is illustrated in Figure 6.4.17 and 6.4.18. It can be easily verified that by optimizing the rule base using GA improves the joint error profile from the previous case where only the constants of the controller were optimized. The statistical comparison is tabulated in Table 6.4.5 and 6.4.6. The scaling factors and rule base actually decides the superiority of the GBSO algorithm over GA. The PD controller constants are kept as $Kp_1 = 8$, $Kd_1 = 1$, $Kp_2 = 8$, $Kd_2 = 1$.



(a)



(b)

Figure 6.4.16 Output membership function of the fuzzy precompensator controller (a) using GA (b) using GBSO

Table 6.4.5 Scaling factors and ISE of the fuzzy precompensated controller

	K_e	K_{de}	K_u	Joint-1 ISE	Joint-2 ISE
GA	2.1569	1.843	7.500	0.0001	4.264e-05
GBSO	9.6078	-0.0392	-8.5882	8.499e-08	2.577e-10

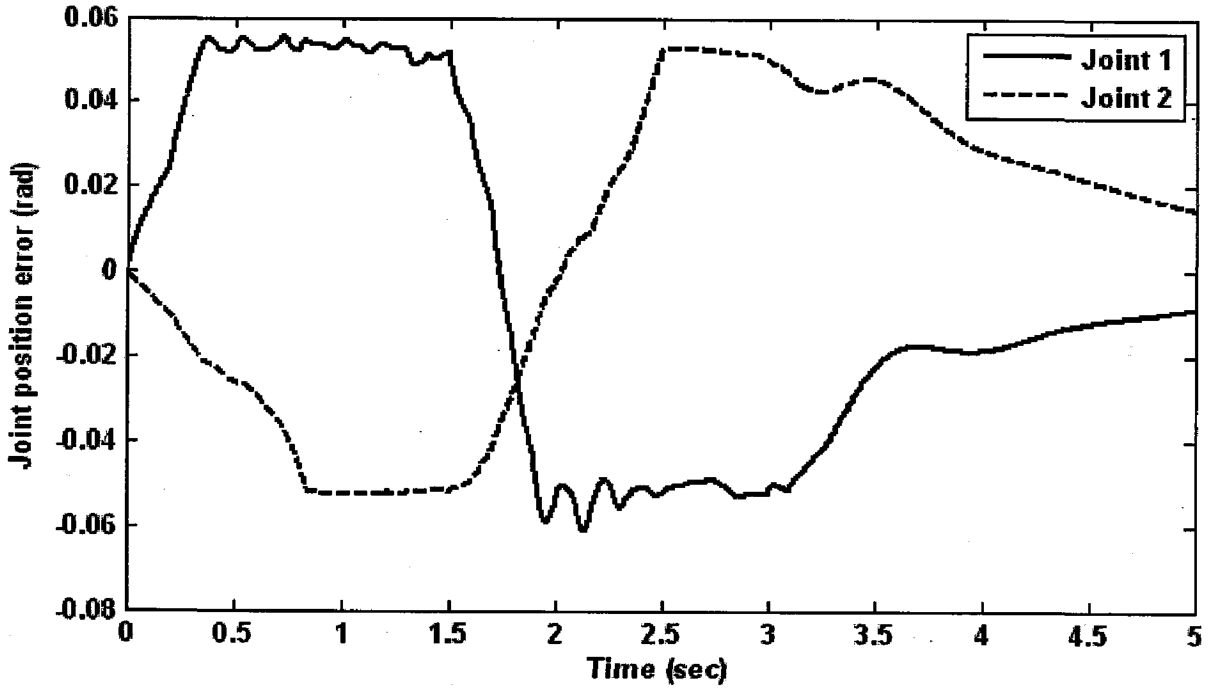


Figure 6.4.17 Joint error profile of hybrid FPPD using GA

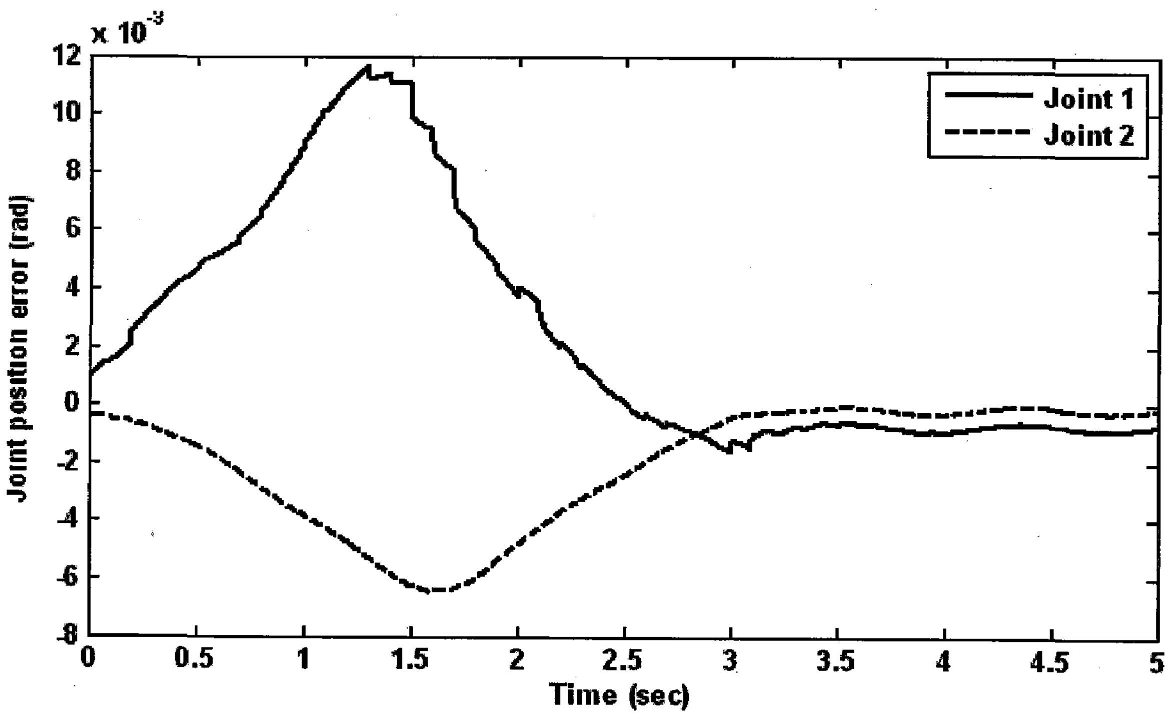


Figure 6.4.17 Joint error profile of hybrid FPPD using GBSO

7. CONCLUSION AND FUTURE SCOPE

In this dissertation, novel hybrid approach consisting genetic algorithm, bacterial foraging and particle swarm optimization and their performances are evaluated using various test functions. It has been established by comparison of the fitness function profile that the developed hybrid algorithms outperformed standard basic techniques. The improvement is shown in terms of convergence rate of the performance index in reaching the optimality over basic optimization algorithms. Also, the proposed algorithm is implemented on a practical ball and beam system supplied by Googol Technology for tuning the PID controller. As evident from the graphical and empirical results, the suggested hybrid system performed exceedingly well.

Fuzzy precompensated PD control also proves its effectiveness by minimizing the overshoot and modifying the required trajectory over the simple PD controller for two link rigid-flexible robot manipulator. The performance of the hybrid fuzzy precompensator is successfully achieved by first optimizing the PD controller constants and then optimizing the fuzzy rule base using characteristic parameters. An advantage of the present approach is that an existing PD controller can be easily modified into the suggested control structure by simply adding a fuzzy precompensator.

As the complexity of the highly uncoupled system increases, the iterative procedure takes a lot of time in reaching the optimal solution. Sometimes, the fitness function converges to the sub-optimal region too as analyzed in two link rigid-flexible manipulator. The future research would include the model complexity reduction using optimizing techniques. This will help in reducing the system order while keeping intact the behavior of original model. Also, the effects of initialization parameters on the convergence behavior of the hybrid algorithms may be worthy to undertake.

REFERENCES

[1] Chaiyaratana, N., Zalzala, A.M.S. "*Recent developments in evolutionary and genetic algorithms: theory and applications*", GALESIA '97, pp. 270-277.

[2] Potter, M. A., De Jong, K. A. "*A cooperative coevolutionary approach to function optimization*", International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature (pp. 249-257). Berlin, Germany: Springer-Verlag, 1994.

[3] Muhlenbein, H., Schlierkamp-Voosen, D, "*Predictive models for the breeder genetic algorithm: I. continuous parameter optimization*" Evolutionary Computation, 1(1), pp. 25-49, 1993.

[4] Francisco Herrera and Manuel Lozano, "*Gradual distributed real-coded Genetic Algorithms*", IEEE transactions on Evolutionary Computation, Vol.4, No.1, pp. 43-63, April 2000.

[5] Ming Chen, Zhengwei Yao, "*Classification Techniques of Neural Networks using improved Genetic Algorithms*", Second IEEE Conference on Genetic and Evolutionary Computing, 2008, pp. 115-119.

[6] Shengxiang Yang and Renato Tinos, "*Hyper-Selection in Dynamic Environments*", IEEE Congress on Evolutionary Computation, 2008, pp. 3185-3192.

[7] Jingjun Zhang, Kanghau Lou, Ruizhen Gao, Guanyuan liu, Yang sun, "*Application of Coarse-Grained Genetic Algorithm for the Optimal Design of the Flexibility Multi-body Model Vehicle Suspensions*", 3rd IEEE Conference on Industrial Electronics and Applications, 2008, pp. 1343-1347.

[8] M.N.H. Siddique, M.O. Tokhi, "*GA-based neuro-fuzzy controller for flexible-link manipulator*", Proceedings of IEEE Conference on Control Applications, pp. 471-476, 2002.

[9] A.L. Buczak, R.E. Uhrig, "*Hybrid fuzzy-genetic technique for multisensor fusion*", Information Sciences 93 (3-4), pp. 265-281, 1996

[10] P. Chootinan, A. Chen, "*Constraint handling in genetic algorithms using a gradient-based repair method*", Computers and Operations Research 33 (8), pp. 2263-2281, 2006

[11] R.C. Eberhart and Y. Shi, "*Comparison between genetic algorithms and particle swarm optimization*", in Proc. IEEE Int. Conf. Evol. Comput., Anchorage, AK, May 1998, pp. 611-616.

[12] Zne-Jung Lee, Chaou-Yaun Lee, "*A hybrid search algorithm with heuristics for resource allocation problem*", Information Sciences, Vol.173, pp. 155-167, 2005.

[13] Shengxiang Yang, Renato Tinos, "*A Hybrid Immigrants Scheme for Genetic Algorithms in Dynamic Environments*", International Journal of Automation and Computing, 2007, pp. 243-254.

[14] J. Kennedy, R. Eberhart, "*Particle Swarm Optimization*", Proceedings of IEEE Conference on Neural Networks, 1995, pp. 1942-1948.

[15] M. Clerc, J. Kennedy, "*The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space*", IEEE Transactions on Evolutionary Computation 6, pp. 58-73, 2002.

[16] K.E. Parsopoulos, M.N. Vrahatis, "*On the Computation of All Global Minimizers Through Particle Swarm Optimization*", IEEE Transactions on Evolutionary Computation, pp. 211-224, 2004.

[17] Y. Tan, Z.M. Xiao, "*Clonal Particle Swarm Optimization and its Application*", IEEE Congress on Evolutionary Computation, 2007, pp. 2303-2309.

[18] Teng-Bo Chen, Yin-Li Dong, Yong-Chang Jiao, and Fu-Shun Zhang, "*Crossed Particle Swarm Optimization Algorithm*", ICNC 2006- Springer-Verlag Berlin Heidelberg, 2006, pp. 935-938.

[19] Juan C. Fuentes Cabrera, Carlos A. Coello Coello, "*Handling Constraints in Particle Swarm Optimization Using a Small Population Size*", Lecture notes in Computer Science, Springer Berlin/Heidelberg, 2007, pp. 41-51.

[20] Ben Niu, Li Li, "*A Novel PSO-DE-Based Hybrid Algorithm for Global Optimization*", ICIC 2008, Springer-Verlag Berlin Heidelberg, pp. 156-163.

[21] Riccardo Poli, William B.L. and Owen Holland, "*Extending Particle Swarm Optimization via Genetic Programming*" Proceedings of 8th European Conference EuroGP, 2005, pp. 291-300.

[22] Chia-Feng Juang, "*A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design*", IEEE Transactions on Systems, Man, and Cybernetics, Vol.34, No.2, pp. 997-1006, 2004.

[23] D.H. Kim, A. Abraham, K. Hirota, "*Hybrid Genetic: Particle Swarm Optimization Algorithm*", Studies in Computational Intelligence: Springer-Verlag Berlin Heidelberg, 2007, pp. 147-170.

[24] K.M. Passino, "*Biomimicry of Bacterial Foraging for Distributed Optimization*", University Press, Princeton, New Jersey, 2001.

[25] K.M. Passino, "*Biomimicry of Bacterial foraging for distributed optimization and control*", IEEE Control Systems Magazine, 2002, pp. 52-67.

[26] D.H. Kim, A. Abraham, J.H. Cho, "*A hybrid genetic algorithm and bacterial foraging approach for global optimization*", Information Sciences, Vol.177 (18), pp. 3918-3937, 2007.

[27] D.H. Kim, J.H. Cho, "*Intelligent Control of AVR system using GA-BF*", in: Rajiv Khosla, Robert J. Howlett, Lakhmi C. Jain (Eds.), Proceedings of KES 2005, Melbourne, Australia, Lecture Notes in Computer Science, Vol.3684/2005, 2005, pp. 854-860.

[28] Arijit Biswas, Sambarta Dasgupta, Swagatam Das, Ajith Abraham, "*Synergy of PSO and Bacterial Foraging Optimization*", Innovations in Hybrid Intelligent Systems, Springer-Verlag Berlin Heidelberg, 2007, pp. 255-263.

[29] Esmaeil Atashpaz Gagari, Farzad Hashemzadeh, Ramin Rajabioun, Caro Lucas ,"*Colonial competitive algorithm: A novel approach for PID controller design in MIMO distillation column process*", International Journal of Intelligent Computing and Cybernetics, pp. 337-355, 2008.

[30] A.H. Mantawy, Youssef L. Abdel-Magid, M.A. Abido, "*A Simulated Annealing Algorithm for Fuzzy Unit Commitment Problem*", IEEE Conference on Transmission and Distribution, pp. 142-147, 1999.

[31] John J. Grefenstette, "*Optimization of Control Parameters for Genetic Algorithms*", IEEE Transactions on Systems, Man, and Cybernetics, p.p. 122-128, 1986.

[32] Crina Grosan, Ajith Abraham, Hisao Ishibuchi, "*Hybrid Evolutionary Algorithms*" Springer-Verlag Berlin Heidelberg, 2007

[33] Tai-Chen Chen, Pei-Wei Tsai, Shu-Chuan Chu, and Jeng-Shyang Pan, "*A Novel Optimization Approach: Bacterial-GA Foraging*" Innovative computing, innovation and control conference, p.p. 391-394, 2007.

[34] Dong Hwa Kim, Jae Hoon Cho, "*A Biologically Inspired Intelligent PID Controller Tuning for AVR Systems*" International Journal of Control, Automation and Systems, Vol.4, No.5, pp. 624-636, 2006.

[35] Andries P. Engelbrecht, "*Computational Intelligence: An Introduction*" John Wiley and Sons, Ltd, Second Edition, 2007.

[36] Kevin M. Passino, Stephen Yurkovich, "*Fuzzy Control*" Addison Werley Longman, First Edition, 1998.

[37] *"Fuzzy Logic Toolbox"* Matlab. http://www.mathworks.com

[38] Young Jun Park, Hyung Suck Cho, Dong Hyuk Cha, "*Genetic Algorithm Based Optimization of Fuzzy Logic Controller Using Characteristic Parameters*" Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 831-836, 1995.

[39] France Cheong, Richard Lai, "*Constraining the Optimization of a Fuzzy Logic Controller Using an Enhanced Genetic Algorithm*" IEEE Transactions on Systems, Man, and Cybernetics, pp. 31-46, 2000.

[40] Elmer P. Dadios, David J. Williams, "*A Fuzzy-Genetic Controller for the Flexible Pole-Cart Balancing Problem*" Proceedings of IEEE Conference on Evolutionary Computation, pp. 223-228, 1996.

[41] F. Bcllezza, L. Lanari, G. Ulivi, "*Exact modeling of the slewing flexible link*" Proceedings of IEEE Conference on Robotics and Automation, Cincinnati, 0H, May 13 18, pp. 734-739, 1990.

Research paper published by the author:

1. **Tushar Jain** and M.J. Nigam, "*Optimization of PD-PI Controller Using Swarm Intelligence*", International Journal of Computational Cognition, Vol.6, No.4, p.p. 55-59, December, 2008.
2. **Tushar Jain**, Vishwanath Patel, M.J. Nigam, "*Implementation of PID Controlled SIMO Process on FPGA Using Bacterial Foraging for Optimal Performance*", International Journal of Computers and Electrical Engineers, Vol.1, No.2, p.p. 109-112, June 2009
3. Srinivasan Alavandar, **Tushar Jain** and Madhav Ji Nigam, "*Bacterial Foraging Optimized Hybrid Fuzzy Precompensated PD Control of Two Link Rigid-Flexible Manipulator*", International Journal of Computational Intelligence Systems, Vol.2, No.1, p.p. 51-59, March 2009
4. Srinivasan Alavandar, **Tushar Jain** and Madhav Ji Nigam, "*Particle Swarm Optimized Hybrid Fuzzy Precompensated Trajectory Control of Rigid-Flexible Manipulator*", International Journal of Knowledge-Based and Intelligent Engineering System (*in press*)
5. Srinivasan Alavandar, **Tushar Jain** and Madhav Ji Nigam, "*Hybrid Bacterial Foraging and Particle Swarm Optimization for Fuzzy Precompensated Control of Flexible Manipulator*", International Journal of Automation and Control. (*in press*)
6. **Tushar Jain**, Srinivasan Alavandar, M.J. Nigam, "*A Hybrid Genetically-Bacterial Foraging Algorithm converged by Particle Swarm Optimization for global optimization*", International Journal of Bio-Inspired Computation-Inderscience Publishers(*in press*)
7. **Tushar Jain**, M.J. Nigam, "*Tuning of Type-1 Servo System Using Swarm Intelligence for SIMO Process*", Journal of Engineering Science and Technology. (*under review*)
8. **Tushar Jain**, M.J. Nigam "*Optimization of PID Controller Using Evolutionary Algorithms*", Proceedings of National Conference on Mechanism Science and Technologies, NIT-Hamirpur, p.p. 44-53 November, 2008.
9. Srinivasan Alavandar, **Tushar Jain**, M.J. Nigam "Synthesis of genetically bacterial swarm algorithm for optimization of fuzzy control of flexible manipulator" (*under writing*)

10. **Tushar Jain**, Srinivasan Alavandar, M J Nigam "Optimization of Modified fuzzy PID control of flexible manipulator using genetically bacterial swarm algorithm" (*under writing*)
11. **Tushar Jain**, Srinivasan Alavandar, M J Nigam "Cooperative Approaches of hybrid genetically bacterial swarm algorithm" (*under writing*)

2. INTELLIGENT COMPUTATIONAL TECHNIQUES

This chapter briefly describes the basic techniques for intelligent computation. The components of genetic algorithm based optimization are discussed in section 1. Behavior of bird flocking is analyzed in section 2 which leads to particle swarm optimization. Section 3 discusses the foraging of behavior of bacteria known as bacterial foraging optimization.

2.1. Genetic Algorithm

Genetic algorithms (GA) are stochastic global search methods inspired by the process of natural evolution. The genetic algorithm starts with no knowledge of the correct solution and depends entirely on responses from its environment and evolution operators (reproduction, crossover and mutation) to arrive at the best solution [31]. By starting at several independent points and searching in parallel, the algorithm avoids local minima and converging to sub optimal solutions.

A genetic algorithm is typically initialized with a random population consisting of between 20-100 individuals. This population is usually represented by a real-valued number or a binary string called a chromosome. How well an individual performs a task is measured by the objective function. The objective function assigns each individual a corresponding number called its fitness value. The fitness of each chromosome is assessed and a survival of the fittest strategy is applied.

2.1.1. Basic Construction of GA's

The basic construction of GA's can be simply described as follows:

1. ***Define the string of chromosome:*** The string of searching parameters for the optimization problem should be defined first. These parameters are genes in a chromosome, which can be binary coded or real coded and termed "chromosome". Different chromosome represents different possible solutions.
2. ***Define the Fitness Function:*** The fitness function is the performance index of GA's to resolve the viability of each chromosome. According to the performance

requirements of the problem, the fitness function can be obtained, e.g., convergence value, error, rise time, etc.

3. ***Generate an Initial Population:*** *N* sets of chromosomes should be randomly generated before using GA's operation. These chromosomes are called the initial population. The size of the population, *N,* is chosen according to the sophistication of the optimization problem. Generally speaking, the larger the value of *N* requires fewer generations to come to a convergent solution. However, the total computation effect depends on *N* times the generation numbers.

4. ***Generate the Next Generation or Stop:*** GA's use the operations of reproduction, crossover, and mutation to generate the next generation. From generation to generation, the maximum value of the fitness value is achieved.

a. ***Reproduction:*** Reproduction is the operator carrying old strings through into a new population, depending on the fitness value. Strings with high fitness values obtain a larger number of copies in the next generation. An example of such an operation is shown in Table-2.1.1

Table-2.1.1 An example of the reproduction of GA's

Old Chromosome	Fitness value	New Chromosome
[101010]	0.3	
[010101]	0.5	
[110110]	0.1	
[011011]	0.9	[011011]

b. ***Crossover:*** Crossover is a recombination operator incorporated with reproduction. It is an effective way of exchanging information segments from high-fitness individuals. The crossover procedure is to randomly select a pair of strings from the mating pool, then randomly determine the crossover position. An example of the operation is shown in Table-2.1.2

c. ***Mutation:*** The mutation operator is used to avoid the possibility of mistaking a local optimum for a global one. It is an occasional random change at some

2.2. Particle Swarm Optimization

In 1995, Kennedy and Eberhart first introduced the particle swarm optimization (PSO) method. It is one of the optimization techniques and a kind of evolutionary computation technique. The particle swarm optimization (PSO) algorithm is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. In PSO, individual, referred to as particles, are "flown" through hyper dimensional search space. Changes to the position of particles within the search space are based on the social-psychological tendency of individuals to emulate the success of other individuals. PSO is therefore a kind of symbiotic cooperative algorithm. A PSO algorithm maintains a swarm of particles, where each particle represents a potential solution. The method has been found to be robust in solving problems featuring nonlinearity and nondifferentiability, multiple optima, and high dimensionality through adaptation, which is derived from the social-psychological theory. The features of the method are as follows [35]:

- The method is developed from research on swarm such as fish schooling and bird flocking.
- It can be easily implemented, and has stable convergence characteristic with good computational efficiency.

Instead of using evolutionary operators to manipulate the particle (individual), like in there evolutionary computational algorithms, each particle in PSO flies in the search space with velocity which is dynamically adjusted according to its own flying experience and its companions' fling experience. Each particle is treated as a volume less particle n *g*-dimensional search space.

Each particle keeps track of its coordinates in the problem space, which are associated with the best solution (evaluating value) it has achieved so far. This value is called pbest. Another best value that is tracked by the global version of the particle swarm optimizer is the overall best value, and its location, obtained so far by any particle in the group, is called gbest. The PSO concept consists of, at each time step, changing the velocity of each particle toward its pbest and gbest location. Acceleration is weighted by the BF and the GA into one algorithm. The BF optimization algorithm is known for its 'excellent local search' capabilities but it does have obvious limitations in its global search approach. This presents a scenario, which is the converse for the GA: it has excellent global search capabilities but is rather limited in its local search procedure. Merging the two algorithms, through selective combination of certain favorable functions of the BF and GA could potentially yield an algorithm that has excellent local and global search capabilities. Every other process within the HBF is exactly the same as those of the BF optimization algorithm apart from the Chemotactic and Reproduction process. The modifications that are implemented through the combination with GA are the reasons for the differences. These constituent processes were modified because they largely determine the effectiveness of the HBF and sensible modifications to such processes can bring about significant improvements in the performance of any algorithm.

3.1.1 Hybrid Chemotaxis

The Hybrid chemotactic process includes the process of tumble and run/swim. It also includes the GA reproductive process which is the modification that aims to improve the normal chemotactic process of the BF. After every bacterium has performed chemotaxis, the new bacteria position and the corresponding nutrient concentration values achieved are modified using the reproductive process adopted from the GA. The idea is to create a new set of bacteria positions from the initial set, which is derived from the tumbles (and swims). The new set obtained through the GA modifications will then be used in next chemotactic step.

In deriving the new set of bacteria positions, first, the initial set of bacteria positions are ranked. The ranking is based on their nutrient concentration values. The smaller values are ranked higher and vice versa. After the ranking process, a certain (user-specified) number of bacteria positions, which are the most highly ranked are passed directly into the new set of bacteria positions. The remaining members of the new set are formed from the initial set by randomly simulating crossover to produce new bacteria positions and then carrying out the mutation function on randomly selected bacteria positions. These GA reproduction operators are described in section 2.1. The new set of bacteria positions are used as the initial positions for the next chemotactic step.

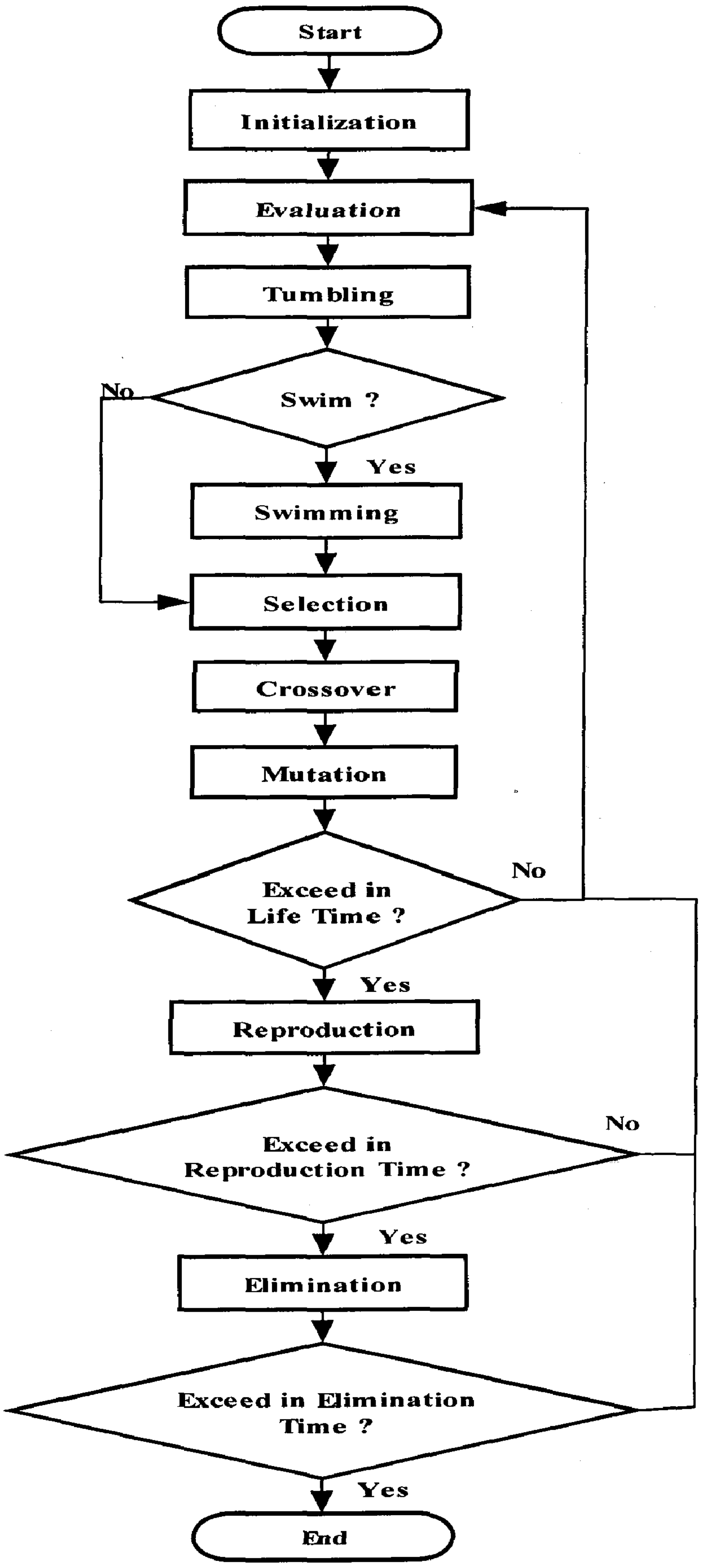


Figure-3.1.1 Flowchart of the hybrid bacterial foraging algorithm [33]

3.1.2 Hybrid Reproduction

A similar modification described for the Hybrid chemotaxis has been implemented for the reproduction phase of the BF to yield the hybrid reproduction of the HBF. The reproduction phase described in section 2.1.1 involves initially assigning fitness values to each member of the bacteria population, ranking each member of the population according to its respective fitness value, killing-off of the bottom-half of the ranked population and finally duplicating each member of the top half of the population. The GA modification alters this reproduction process by using the operators of crossover and mutation to produce a new population, rather than merely duplicating the top half of the ranked bacterial population.

The bacterial population is ranked according to their fitness values of each bacterium. The fitness value is derived by considering only a single value, which corresponds to the best (minimum) nutrient concentration value the bacterium experienced all through the chemotactic process. The lower the fitness value of bacterium, the better is rank and vice versa. Having achieved a ranked bacterial population, a number of highly ranking bacterium is passed unaltered to the new population of bacteria. The remaining members of the new population are obtained by applying the functions of crossover and mutation randomly on the remaining bacteria within the ranked population. The modified approach to reproduction enables a better chance of convergence of the bacterial population to the positions that correspond to the best structure and parameters for the controller being designed. At the same time, the mutation function enables the algorithm to search wider areas within the sample space thus enhancing the global nature of the search procedure. The flowchart of hybrid bacterial foraging is shown in Figure-3.1.1.

3.2. Hybrid Bacterial Foraging-Particle Swarm Optimization

The hybrid bacterial foraging-particle swarm optimization (BF-PSO) was proposed by Arijit et. al. [28]. In his approach, after undergoing a chemo-tactic step, each bacterium also gets mutated by a PSO operator. In this phase, the bacterium is stochastically attracted towards the globally best position found so far in the entire population at current time and also towards its previous heading direction. The PSO

3.3. Genetically-Bacterial Swarm Optimization

The main goal of GBSO algorithm is to find the minimum of a function $J(\theta)$, $\theta \in R^p$ which is not in the gradient $\nabla J(\theta)$. Here, θ is the position of the bacterium, and $J(\theta)$ is an attractant-repellant profile. That is, where nutrients and noxious substances are located, $J < 0$, $J = 0$, and $J > 0$ represents the presence of nutrients. A neutral medium, and the presence of noxious substances, respectively can be defined by

$$P(j,k,l) = \{\theta^i(j,k,l) \mid i = 1,2,\ldots,S\} \tag{3.3.1}$$

Eqs. (3.3.1) represents the position of each member in the population of S bacteria at the jth chemotactic step, kth reproduction step, and lth elimination-dispersal event. The co-ordinates of the bacterium here represent an individual solution of the optimization problem. The approach to GBSO technique is considered in two phases: first, the genetic selection using stochastic universal sampling method, crossover using extended intermediate recombination and mutation as used in BGA [3] are included in the chemotaxis loop which forces bacteria to exchange the information they carried to the others via switching the information on parts of the dimension. These operators extract common features from different bacteria in order to achieve even better solutions. Secondly, the search direction vector is iterated using PSO algorithm. The randomly initialized direction vector in basic BF algorithm remains same throughout the algorithm which can result delay in reaching the optimal solution and at times it can converge into some sub-optimal region. This delay is handled using PSO velocity equation. Furthermore, the elimination course is marginally different from the process in BF. If there is a probability of elimination-dispersal event to occur then instead of generating another population via the initialization process as considered in basic BF algorithm, the whole new individuals are generated via mutating all the dimensions from the eliminated one.

In this work, the GA implementation presents the following characteristics:

3.3.1 Ranking and Selection in GBSO

In a minimization problem of function $J(\theta)$, a 'ranking' operation [32] is performed where individuals are sorted in decreasing $J(\theta)$ value first, and then, $J(\theta)$ is replaced by its position. Each individual has a new cost function value $J'(\theta)$.

Selection is made by the operator known as *Stochastic Universal Sampling (SUS)* [31]. If N_{ind} is the number of individuals, then the survival probability of an individual $P(\theta_i)$ is guaranteed to be:

$$P(\vec{\theta}_i) = \frac{J'(\vec{\theta}_i)}{\sum_{j=1}^{Nind} J'(\vec{\theta}_j)} \tag{3.3.2}$$

3.3.2 Crossover operation in GBSO

An extended intermediate recombination [30] is used for the GBSO algorithm as:

$$z_i = x_i + \alpha_i (y_i - x_i), i = 1,....,n \tag{3.3.3}$$

where $x = (x_1,...x_n)$, $y = (y_1,...y_n)$ are the parents and $z = (z_1,...z_n)$ is the successor, α_i is a random number generator. The operation, achieved with a probability P_c can be performed on each bacteria, separately.

3.3.3 Mutation operation in GBSO

The mutation operation is performed for each bacterium with a probability $\underline{P_m}$. Then, a random value is added for each individual – generated with a normal distribution and a standard deviation set to 20% of the search space range. If necessary, the mutated individual is kept in the search space by truncation.

3.3.4. Particle Swarm Optimization

PSO [10] is a stochastic optimization technique that draws inspiration from the behavior of flock of birds or the collective intelligence of a group of social insects with limited individual capabilities. The *E coli* algorithm depends on random search directions which may lead to delay in reaching the global solution. The velocity update equation of PSO algorithm is used for optimizing the search direction in *E coli* algorithm which decreases the delay and hence enhances the convergence rate. They move iteratively through the d-dimension problem space to search the new solutions. Each particle has a position represented by a position vector X_k^i where (i is the index of the particle), and a velocity represented by a velocity-vector V_k^i. Each particle remembers its own best position P_{Lbest}^i. The best position vector among the swarm then stored in a vector P_{Global}^i. During the iteration time k, the update of the velocity from the previous velocity to the new velocity is determined by

$$V_{k+1}^i = \omega V_k^i + C_1 R_1 (P_{Lbest}^i - X_k^i) + C_2 R_2 (P_{Global}^i - X_k^i) \qquad (3.3.4)$$

The new position is then determined by the sum of the previous position and the new velocity.

$$X_{k+1}^i = X_k^i + V_{k+1}^i \qquad (3.3.5)$$

where R_1, R_2 are random numbers and C_1, C_2 are learning factors.

3.3.5 The GBSO algorithm

The GBSO algorithm to search optimal values of parameters is described as follows:

[Step 1] Initialize parameters $p,\ S,\ N_c,\ N_s,\ N_{re},\ N_{ed},\ P_{ed},\ C(i)(i=1,2,\ldots,S),\ \theta^i$

Where,

p,	Dimension of the search space
S,	Number of bacteria in the population
N_c,	No. of Chemotactic steps
N_s,	Swim length

Certain characteristics of the rule-base are assumed in using the proposed construction method:

- Extreme outputs more usually occur when the inputs have extreme values while mid-range outputs generally are generated when the input values are mid-range.
- Similar combinations of input linguistic values lead to similar output values

Using these assumptions the output space is partitioned into different regions corresponding to different output linguistic values. How the space is partitioned is determined by the characteristic spacing parameters and the characteristic angle. The angle determines the slope of a line[1] through the origin on which seed points are placed. The positioning of the seed points is determined by a similar spacing method as was used to determine the centre of the membership functions.

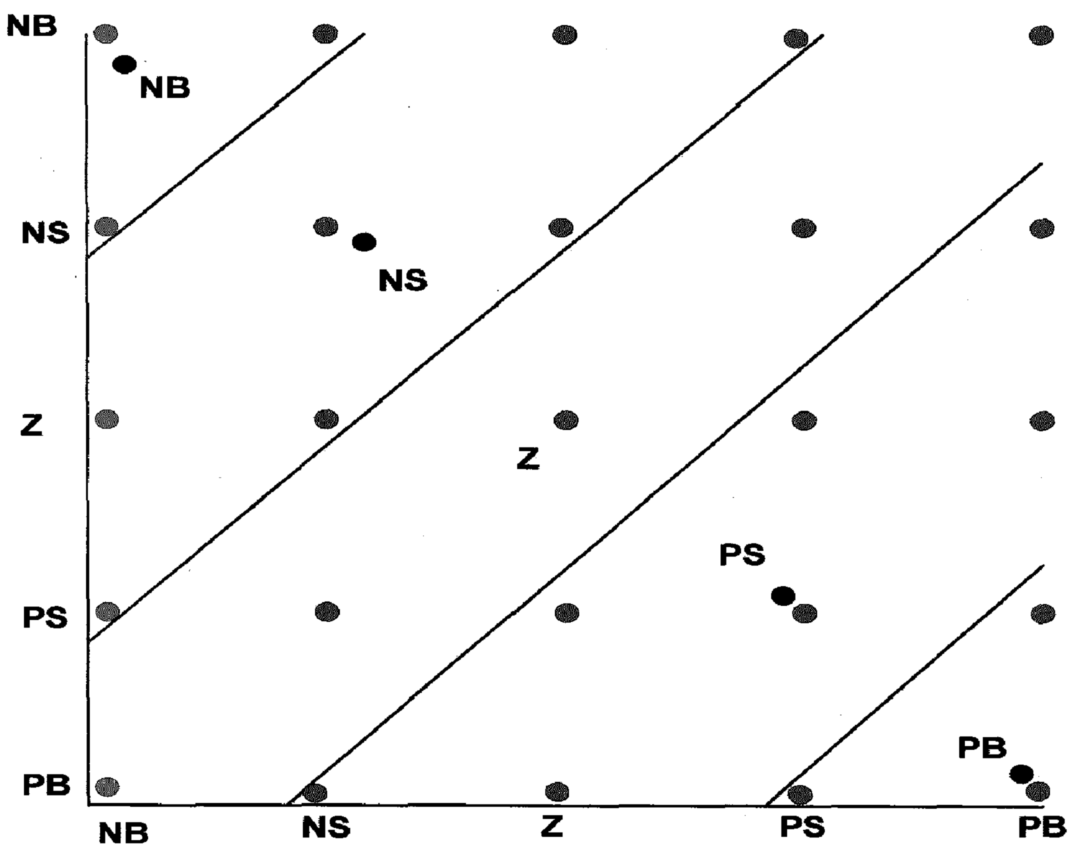


Figure-4.2.2 Seed Points and Grid Points for rule-base construction