Intelligent Control and Automation, **2010, 1, 82-89**
doi:10.4236/ica.2010.12009 Published Online November 2010 (http://www.SciRP.org/journal/ica)

A Novel Stochastic Algorithm Using Pythagorean Means for Minimization

A. Mona Subramaniam, A. Manju, Madhav J. Nigam
Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee, India
E-mail: {*monasubramaniama, manju.senthil*}*@gmail.com, mkndnfec@iitr.ernet.in*
Received July 1, 2010; *revised September* 1, 2010; *accepted November* 3, 2010

Abstract

In this paper, A Novel Stochastic Algorithm using Pythagorean means for minimization of the objective function is described. The algorithm is initially tested with Rastrigin's function and compared with Genetic algorithm results for the function with the same initial conditions. After this, it is used in tuning the gains of fuzzy PD + I controller for trajectory control of PUMA 560 robot manipulator. The results are again verified with the results of genetic algorithm.

Keywords: Stochastic Algorithm, Pythagorean Means, Gain Tuning, Fuzzy Controller, Genetic Algorithm, PUMA560 Trajectory Control, Robotics

1. Introduction

PUMA 560 is a six Degree of Freedom industrial robot. It has a wide variety of applications including material handling, welding, assembling, painting, grinding and other industrial applications [1]. Fuzzy PD+I controllers are the most general use fuzzy controller as it has the following advantages of being Simple, having Less overshoot, Removes steady state error, and smoothens control signal [2]. The most popular technique in evolutionary computation research has been the genetic algorithm which can be applied to any problem that can be formulated as function optimization problem [3]. By tuning the gains of the Fuzzy PID controller using genetic algorithm and other search algorithms better results are obtained [1,4]. Some of the basics of any stochastic algorithms are discussed in the work of *Pierre and Jean* [5].

The main motivation for writing this paper is to propose a stochastic algorithm that will minimize the objective function and converge faster than the Genetic Algorithm. As part of preliminary study, this algorithm will be tested with a standard test function, namely Rastrigin's function, and later on used for tuning the gains of fuzzy PD+I controller for trajectory control of PUMA 560 robot manipulator. The convergence of both the standard test function and gain tuning by proposed algorithm are compared with genetic algorithm convergence. An inference is drawn as to which algorithm is better with reference to convergence plot.

This paper is organized as follows. Section 2 introduces PUMA560 robot arm and its dynamics. Section 3 provides discussion on Fuzzy PD + I controller scheme. Section 4 describes Genetic Algorithm. Section 5 introduces a Novel Stochastic Algorithm for optimization. Section 6 results and discussions are provided, and in Section 7 conclusions and future scope are presented.

2. Dynamics of PUMA 560

Dynamics of a serial n-link rigid robot can be written as:

$$M(q)\ddot{q} + c(q,\dot{q}) + g(q) = \tau \qquad (1)$$

where q is the n × 1 vector of joint displacements, $\dot{q}$ is the n × 1 vector of joint velocities, τ is the n × 1 vector of actuators applied torques, $M(q)$ is the n × n symmetric positive definite manipulator inertia matrix, $c(q,\dot{q})$ is the n × 1 vector of centripetal and Coriolis torques and $g(q)$ is the n × 1 vector of gravitational torques due to gravity. We assume that the robot joints are joined together with revolute joints.

Let the desired joint position q_d be a twice differentiable vector function. We define a control problem to determine the actuator torques in such a way that the following control aim be achieved:

$$\lim_{t->\inf} q(t) = q_d(t) \tag{2}$$

A six DOF PUMA 560 robot is considered for the simulation, the Kinematical and dynamical parameters of the arm and the torque limitations are adopted from the work of *Srinivasan et al, and references therein* [6].

3. Fuzzy PD + I Controller

In this structure, the fuzzy system is applied only to the proportional and derivative signal of the linear PID controller [7]. The integral signal uses conventional linear method. The major roll of the integral signal is to eliminate the steady state error. The transient response is affected mostly by the proportional signal and the derivative signal. For the enhancement of the transient response, the varying gains are implemented on the proportional and derivative parts using two-input fuzzy system. The nonlinearities that make the varying gains possible are added by the fuzzy control rules and the membership functions.

Figure 1 shows the structure of the Fuzzy PD + I controller, where GCE, GE, GIE and GU are the gains of Fuzzy PD + I controller and more often called scaling factors which can be varied to tune the controller. Genetic Algorithm and proposed stochastic algorithm are used in this paper to tune these. **Figure 2(a)** shows the surface view of the fuzzy PD controller and **Figure 2(b)** shows the membership function and **Figure 2(c)** shows the rulebase of the Fuzzy PD controller used.

4. Genetic Algorithm

Genetic Algorithms are general purpose search algorithms which use principles inspired by natural genetics to evolve solutions to problems. The basic idea is to maintain a population of *chromosomes* which represents candidate solutions to the concrete problem being solved, that evolves over time through a process of competition and controlled variation. Each chromosome in the population has an associated *fitness* to determine (*selection*) which chromosomes are used to form new ones in the competition process. The new ones are created using genetic operators such as crossover and mutation.

A GA starts off with a population of randomly generated chromosomes, and advances toward better chromosomes by applying genetic operators modeled on genetic processes occurring in nature. The population undergoes evolution in a form of natural selection. During successive iterations, called *generation*, chromosomes in the population are rated for their adaptation as solutions, and on the basis of these evaluations, a new population of chromosome is formed using selection mechanism and specific genetic operators such as *crossover* and *mutation*. An evaluation or fitness function(*f*) must be devised for each problem to be solved. Given a particular chromosome, a possible solution, the fitness function returns a single numerical fitness, which is supposed to be proportional to the utility or adaptation of the solution represented by that chromosome.

Although there are many possible variants of the basic GA, the fundamental underlying mechanism consists of three operations:

1) Evaluations of individual fitness,

2) Formation of a gene pool (intermediate population) through selection mechanism, and,

3) Recombination through crossover and mutation operators.

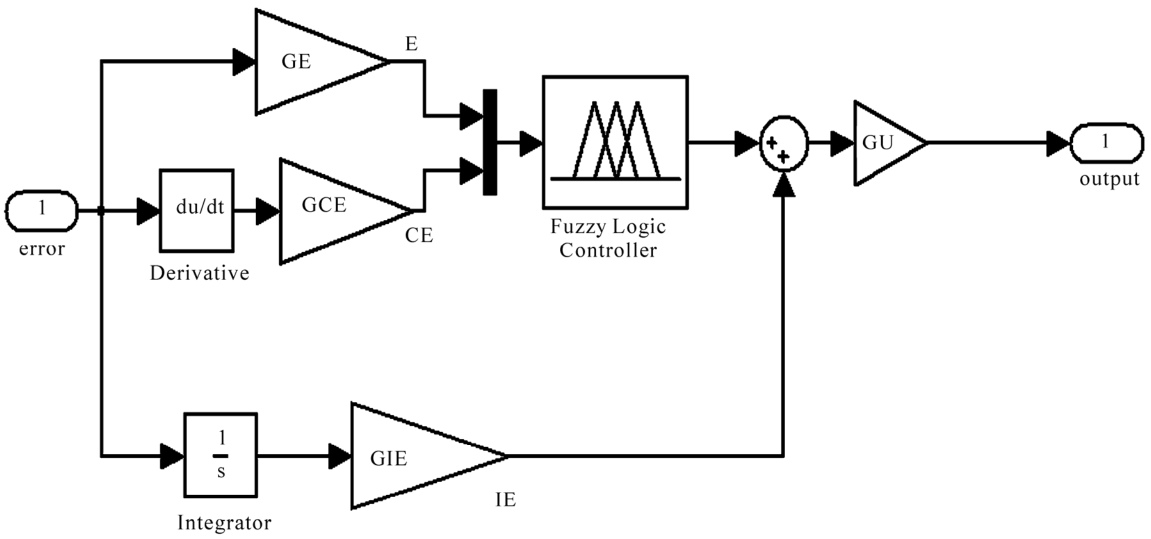
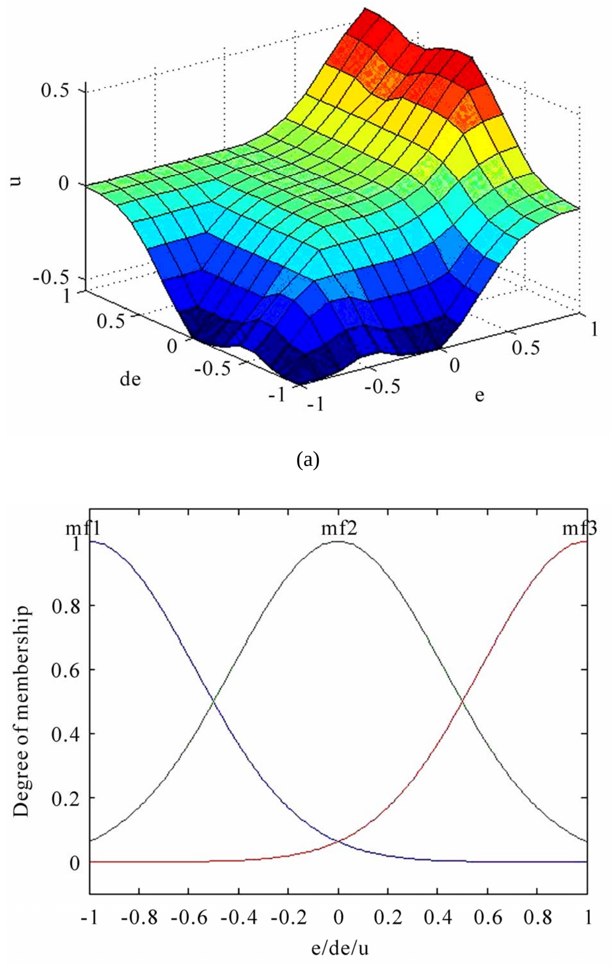
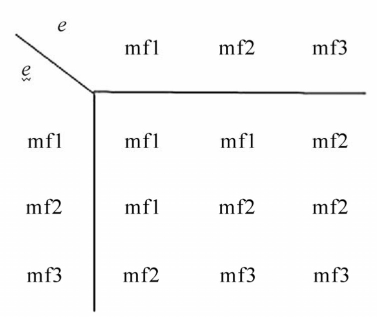


Figure 1. Structure of the fuzzy PD + I controller.



(a)

(b)



e \ $\dot{e}$	mf1	mf2	mf3
mf1	mf1	mf1	mf2
mf2	mf1	mf2	mf2
mf3	mf2	mf3	mf3

(c)

Figure 2. (a) Surface view, (b) Membership function (Mamdani type) and (c) Rulebase of the fuzzy PD controller.

The below pseudo-code shows the structure of a Basic GA [8], where *P*(*t*) denotes the population at generation *t*.

```
Procedure Genetic Algorithm
Begin(1)
    $t$ = 0;
    initialize $P(t)$;
    evaluate $P(t)$;
    While (Not termination-condition) do
      Begin(2)
    t = t + 1;
    select $P(t)$ from $P(t-1)$;
    recombine $P(t)$;
    evaluate $P(t)$;
        End (2)
End(1)
```

5. A Novel Stochastic Algorithm Using Pythagorean Means for Optimization

Given a space Ω of individual solutions $\omega \in R^n$ and an objective function *f*, $f(\omega) \to R$, optimizing *f* is the process of finding the solution ω^* which minimizes (maximizes) *f*.

Random search consists of picking up random potential solutions and evaluating them. The best solution over a number of samples is the result of random search. Stochastic algorithm is nothing other than a random search, with hints by a chosen heuristics (or meta-heuristics) to guide the next potential solution to evaluate [5]. Stochastic optimization algorithms were designed to deal with highly complex optimization problems. There are a number of stochastic search algorithms present, Genetic algorithm, simulated annealing, ant colony optimization, etc to name a few. In this paper the proposed algorithm is compared with the genetic algorithm, the scope of work is restricted to comparison with genetic algorithm only. The proposed stochastic search algorithm is initially tested on Rastrigin's function and then later is used to minimize the ISE of the joint for trajectory tracking control of the PUMA560.

Before describing the algorithm, a small discussion about Pythagorean means is given. The three classical Pythagorean means are the arithmetic mean (A), the geometric mean (G), and the harmonic mean (H) [9]. They are defined by:

$$A(x_1,...,x_n) = \frac{1}{n}(x_1 + ... + x_n) \tag{3}$$

$$G(x_1,...,x_n) = \sqrt[n]{x_1 \cdots x_n} \tag{4}$$

$$H(x_1,...,x_n) = \frac{n}{\frac{1}{x_1} + \cdots + \frac{1}{x_n}} \tag{5}$$

Each of these means satisfies the properties:

$$M(x,x,...,x) = x \tag{6}$$

$$M(bx_1,...,bx_n) = bM(x_1,...,x_n) \tag{7}$$

There is an ordering to these means (if all of the x_i are positive):

$$A\left(x_1,\ldots,x_n\right)\geq G\left(x_1,\ldots,x_n\right)\geq H\left(x_1,\ldots,x_n\right) \quad (8)$$

with equality holding if and only if the x_i are all equal.

The search algorithm is divided into three phases. In the first phase the search space is searched thoroughly and if the solution exists outside this search space then the space is extended, this can be considered as a global search. In the second phase the search space restricted but still able to change its neighborhood considerably, this can be considered as a moderately local search. And in the third phase the search is completely restricted to a very small space in the neighborhood of the solution, this can be considered as a highly restricted local search phase.

Before starting the algorithm, as part of initialization a random number vector '*x*' is generated within the range specified as the search space, this is done just once. With the start of the algorithm, value of *x* is used as the centre and absolute value of *x* is used as spread of the normal distribution in the first iteration. The spread should always be positive. The random numbers are generated by a Gaussian function, in other words they are normally distribution. After the first iteration the number generated by the normal distribution is taken as '*x*'. The algorithm is discussed phase by phase. Let y be the function to be minimized.

Phase I: This is the phase of global search, so the neighborhood to be searched from the present solution has to be very large. This is ensured by making the spread of the normal distribution equal to the arithmetic mean of the best solution of *y* so far and the absolute value of random number *x* generated in the previous iteration by the normal distribution. By using the number generated in the previous iteration in calculating the spread, it is ensured that there is a certain degree of randomization or perturbation in search neighborhood, which will ensure that unknown better solutions can be explored. The mean of the normal distribution will be the values of *x* corresponding to the solution of best *y*. If a better solution is obtained the best x and best y are updated. When best x is updated the centre of the distribution also moves to this location. This way it is ensured that the search space is made variable, moving towards better solution. The reason for using arithmetic mean for spread in this stage is that the value spread will be large and half way between the x and y. For example let us consider that y is a function of x where x is a vector of size 1. Now if for x = 10, best y = 0.2, then the spread that will be used for next search will be (10 + 0.2)/2 which is a value 5.1. This is a pretty large value of spread considering the value of x.

Phase II: This is the phase of moderate local search, so the neighborhood to be searched from the present solution has to be moderate. This is ensured by making the spread of the normal distribution equal to the geometric mean of the best solution of *y* so far and the absolute value of random number *x* generated in the previous iteration by the normal distribution. The mean of the normal distribution will be the values of *x* corresponding to the solution of best *y*. If a better solution is obtained the best x and best y are updated. When best x is updated the centre of the distribution also moves to this location. The reason for using geometric mean for spread in this stage is that the value of spread will be close to the minimum value of x and y. For example let us consider that y is a function of x where x is a vector of size 1. Now if for x = 10, best y = 0.2, then the spread that will be used for next search will be $\sqrt{10*0.2}$ which is 1.141. This is a moderate value of spread considering the value of x.

Phase III: This is the phase of extreme local search, so the neighborhood to be searched from the present solution has to be very small. This is ensured by making the spread of the normal distribution equal to the harmonic mean of the best solution of *y* so far and the absolute value of random number *x* generated in the previous iteration by the normal distribution. The mean of the normal distribution will be the values of *x* corresponding to the solution of best *y*. If a better solution is obtained the best x and best y are updated. When best x is updated the centre of the distribution also moves to this location. The reason for using harmonic mean for spread in this stage is that the value of spread will be very close to the minimum value of x and y. For example let us consider that y is a function of x where x is a vector of size 1. Now if for x = 10, best y = 0.2, then the spread that will be used for next search will be $\frac{2*(10*0.2)}{(10+0.2)}$ which is 0.39. This is a small value of spread considering the value of x.

The value of the x corresponding to the best y is returned as a result of possible solution. Flow chart of the proposed algorithm is shown in **Figure 3**.

This algorithm is initially used to find the minimum of Rastrigin's function. A comparison of Rastrigin's function results by the proposed algorithm and Genetic algorithm is made. Based on the success it is implemented for tuning the gains of the Fuzzy PD + I controlled PUMA 560 arm for pseudo-random joint trajectories. A comparison of results by the proposed algorithm and Genetic algorithm is made.

The objective functions considered here for gain tuning is based on the error criterion. In this paper, performance of gain tuning is evaluated in terms of Integral square Error (ISE) error criteria. The error criterion is given as a measure of performance index. The ISEs of individual joints are added together to obtain an overall

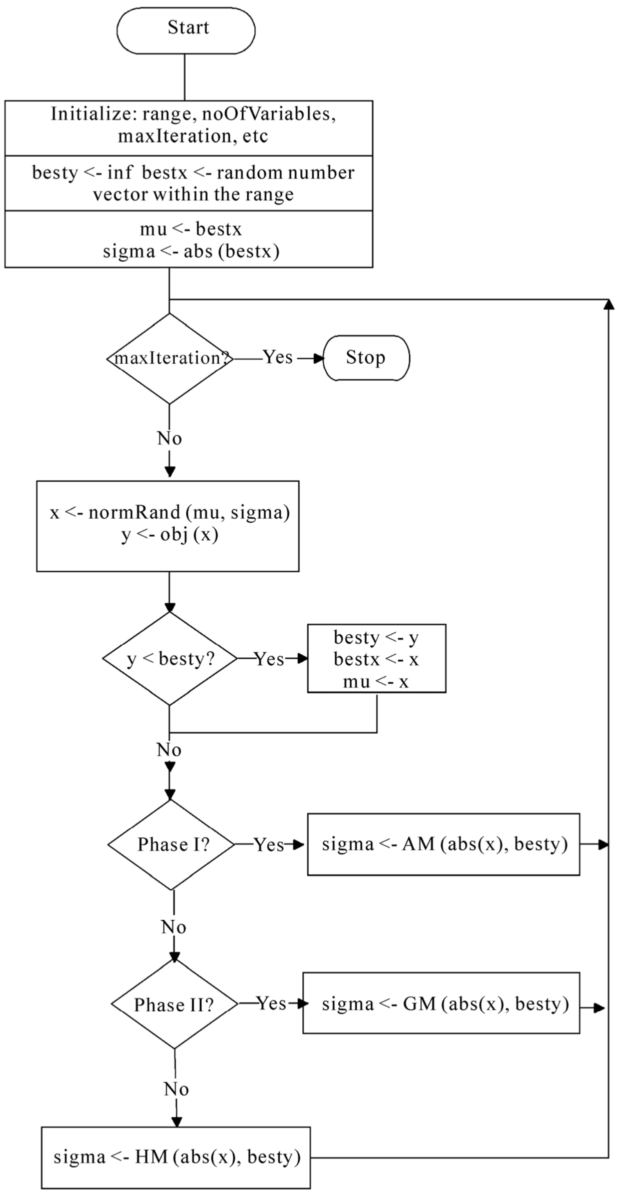

Start
Initialize: range, noOfVariables, maxIteration, etc
besty <- inf bestx <- random number vector within the range
mu <- bestx
sigma <- abs (bestx)
maxIteration?
Yes
Stop
No
x <- normRand (mu, sigma)
y <- obj (x)
y < besty?
Yes
besty <- y
bestx <- x
mu <- x
No
Phase I?
Yes
sigma <- AM (abs(x), besty)
No
Phase II?
Yes
sigma <- GM (abs(x), besty)
No
sigma <- HM (abs(x), besty)

Figure 3. Flow chart of the proposed stochastic optimization algorithm.

ISE. This is done to simplify the task of minimization algorithm. The objective of the minimization algorithm is to minimize this overall ISE. The overall ISE is given by Equation (9).

$$ISE = \sum_{i=1}^{6} \int e_i^2(t)\,dt \quad (9)$$

where $e_i(t)$ is the error signal for the ith joint. Here i can take values from 1 to 6 corresponding to 6 joints.

6. Results and Discussions

Simulations are carried out using MATLAB Version 7.0.1.24704 (R14 with Genetic algorithm & Direct search toolbox and Fuzzy Logic Toolbox), 2 GHz computer with 2 GB RAM. Population size used is 20. To verify the functioning of the proposed algorithm, it is first tested on Rastrigin's function. Rastrigin's function is often used to test the algorithm, because it has many local minima. Once the algorithm works on Rastrigin's function, it can be tested on other functions. For two independent variables, Rastrigin's function is defined as in Equation (10)

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2) \quad (10)$$

Rastrigin's function has many local minima but the function has just one global minimum [10], which occurs at the point [0,0] in the x-y plane, where the value of the function is 0. At any local minimum other than [0,0], the value of Rastrigin's function is greater than 0. The farther the local minimum is from the origin, the larger the value of the function is at that point. **Figure 4** show the plot of Rastrigin's function.

Table 1 shows the tabulated results for Rastrigin's function by Genetic algorithm and proposed algorithm. The initial range of search space was [9,10].

Figure 5 shows the comparative plot for objective function value of Rastrigin's function by Genetic algorithm and proposed algorithm with the x axis denoting the objective function value of Rastrigin's function and y axis denoting the number of iteration. Since we are using

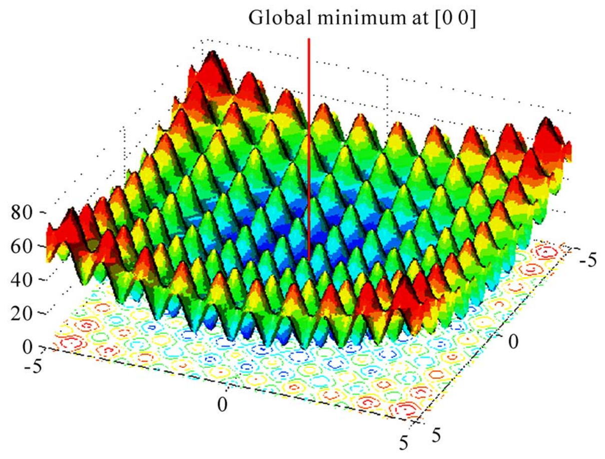


Figure 4. Plot of the Rastrigin's function.

Table 1. Results for Rastrigin's function by genetic algorithm and proposed algorithm.

Algorithm used	x1	x2	y
Genetic Algorithm	0.0011	0.99877	0.99808
Proposed algorithm	2.06E-07	-2.21E-07	1.81E-11

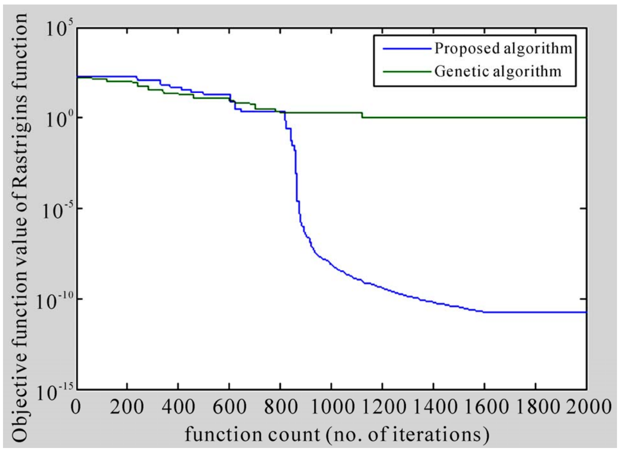


Figure 5. Comparative plot for objective function value of Rastrigin's function by Genetic algorithm and proposed algorithm.

genetic algorithm with population size of 20, one generation corresponds to 20 iterations. Genetic algorithm is run for 100 generations, in other words 2000 iterations.

With the encouraging Rastrigin's function result, the algorithm is used for tuning the gains of the Fuzzy PD + I controller of PUMA 560. A Pseudo-random joint angle trajectory is used as input to robot arm as shown in **Figure 6**. Pseudo random joint trajectory generation is discussed in Appendix A.

The objective function used is as described in Equation (9). **Figure 7** shows the convergence of both Genetic algorithm and the proposed algorithm. **Figure 8** shows the joint error plot for gain tuning by Genetic algorithm and **Figure 9** shows the joint error plot for the gain tuning by the proposed algorithm.

The value of the performance index by the Equation (9) is 4.15E-03 for Genetic algorithm after 1000 iterations (50 generation) and 3.63E-03 for the proposed algorithm after 1000 iterations.

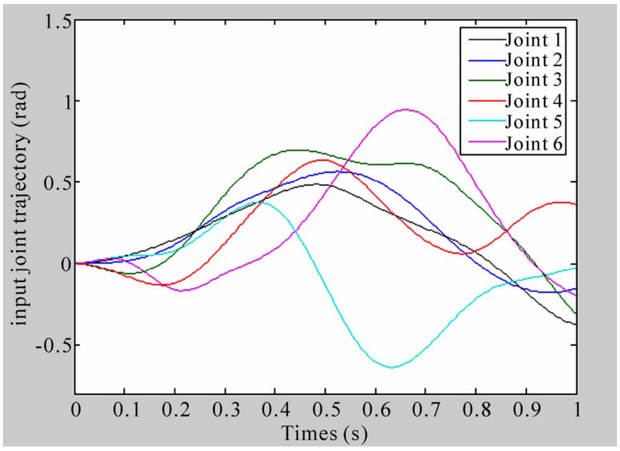


Figure 6. Pseudo-random input joint angle trajectories.

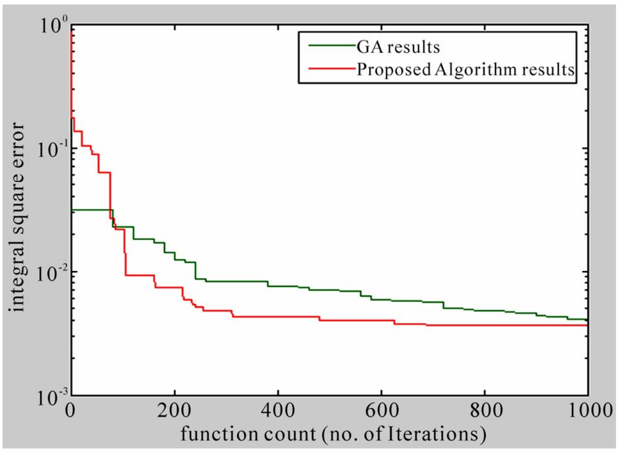


Figure 7. Convergence plot of Genetic algorithm and the proposed algorithm.

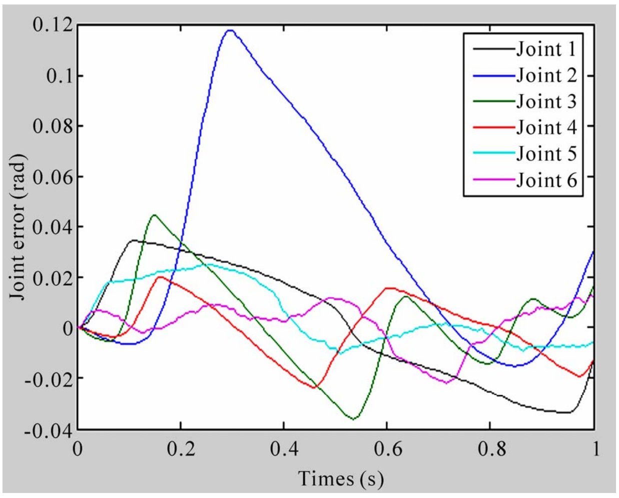


Figure 8. Joint errors plot for gain tuning by Genetic algorithm.

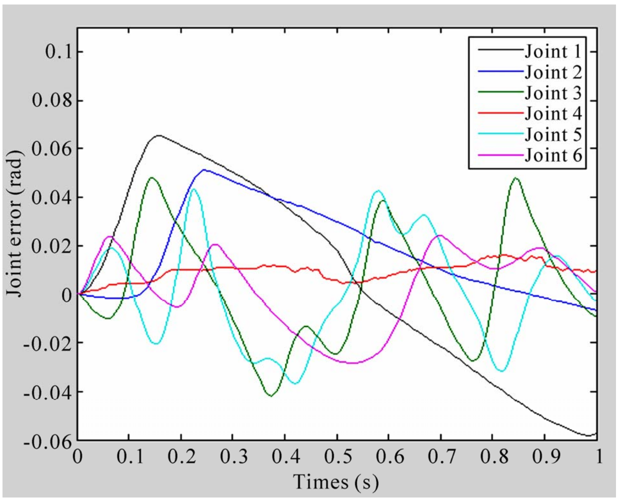


Figure 9. Joint errors plot for gain tuning by proposed algorithm.

7. Conclusion and Future Scope

The results obtained for the Rastrigin's function shown in **Figure 5** and **Table 1** is quite encouraging. The minimum value obtained for Rastrigin's function is very close to the value 0 when evaluated using the proposed algorithm. It proves its worth by being able to converge at a faster rate when compared to Genetic algorithm. It is also seen that gain tuning of Fuzzy PD + I controller of PUMA560 using the proposed algorithm results in a quicker convergence and better performance than genetic algorithm, performance based on the overall ISE. In future minimization and optimization of much more complicated problems using the proposed algorithm can be taken up.

8. References

[1] S. A. Mazhari and S. Kumar, "PUMA 560 Optimal Trajectory Control using Genetic Algorithm, Simulated Annealing and Generalized Pattern Search Techniques," *International Journal of Electrical, Computer and Systems Engineering*, Vol. 2, No. 1, 2008, pp. 71-80.

[2] J. Jantzen, "Foundations of Fuzzy Control," John Wiley & Sons Ltd, Hoboken**,** 2007.

[3] S. N. Sivanandam and S. N. Deepa, "Introduction to Genetic Algorithms," Springer-Verlag, Berlin, Heidelberg, 2008.

[4] S. A. Mazhari and S. Kumar, "Heuristic Search Algorithms for Tuning PUMA 560 Fuzzy PID Controller," *International Journal of Computer Science*, Vol. 3, No. 4, Fall 2008, pp. 277-286.

[5] P. Collet and J.-P. Rennard, "Stochastic Optimization Algorithms," Handbook of Research on Nature Inspired Computing for Economics and Management, 2006.

[6] S. Alavandar and M. J. Nigam, "Fuzzy PD + I Control of a Six DOF Robot Manipulator," *Industrial Robot*: *An International Journal*, Vol. 35, No. 2, 2008, pp. 125-132.

[7] B.-J. Kim and C.-C. Chung, "Design of Fuzzy PD + I Controller for Tracking Control," *Proceedings of the American Control Conference Anchorage*, Alaska, 8-1 May 2002, pp. 2124-2129.

[8] F. Herrera and L. Magdalena. "Genetic Fuzzy Systems: A Tutorial," In: R. Mesiar and B. Riecan, Eds., *Fuzzy Structures, Current Trends. Lecture Notes of the Tutorial: Genetic Fuzzy Systems. Seventh IFSA World Congress* (*IFSA*97), Tatra Mountains Mathematical Publications, Prage, Vol. 13, 1997, pp. 93-121.

[9] "Pythagorean Means". http://en.wikipedia.org/wiki/Pythagorean_means

[10] MATLAB Help file, MATLAB Version 7.0.1.24704 (R14 with Genetic algorithm & Direct search toolbox and Fuzzy Logic Toolbox), The MathWoks Inc, 2004.

Appendix

The method for generating pseudo random joint trajectories is described below:

A sequence of Pseudo random numbers is passed through a system with transfer function given chosen by trial and error. Transfer function chosen is $\frac{100}{s^2+s+100}$. When the pseudo random numbers are passed through this system a continuous pseudo-random signal is obtained which is given to the joint angles.

Time intervals at which the numbers in the sequence appears are
[0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1], that is at time = 0 the first element in the sequence appears, at time = 0.1 the second element and so on.

Joint 1 Pseudo-random sequence:
[0.1 0.1 0.2 0.3 0.4 0.2 0.4 0.2 –0.1 –0.2 0]

Joint 2 Pseudo-random sequence:
[0 0.1 0.3 0.2 0.5 0.4 0.3 0.2 0.1 0 0]

Joint 3 Pseudo-random sequence:
[–0.3 0.2 0.4 0.3 0.4 0.6 0.8 0.2 0.4 –0.2 0]

Joint 4 Pseudo-random sequence:
[–0.2 –0.1 0.3 0.2 0.4 0.1 0.4 0.3 0.5 0.08 0]

Joint 5 Pseudo-random sequence:
[0.2 –0.1 0.4 0.2 –0.4 –0.2 0.1 –0.3 –0.5 –0.08 0]

Joint 6 Pseudo-random sequence:
[0.3 –0.5 0.4 –0.2 0.4 0.6 0.5 0.3 0.5 0.2 0]