Dhaka Univ. J. Sci. 60(2): 239-245, 2012 (July)

Dynamic Modelling of Flexible Manipulator System Using Genetic Algorithm

M. S. Alam

Department of Applied Physics, Electronics and Communication Engineering, University of Dhaka, Dhaka-1000, Bangladesh

Email: msalam@univdhaka.edu

Received on 10. 09. 2011. Accepted for Publication on 12. 12. 2011.

Abstract

Flexible robotic manipulators pose various challenges in modelling, design, structural optimisation and control. This paper presents investigations into practical dynamic modelling of a flexible manipulator system using genetic algorithm (GA). Conventional genetic algorithms (GAs) often converge prematurely to a suboptimal region and fail to provide effective solutions due to lack of diversity in the population set as the algorithm proceeds. In order to improve and maintain diversity in the population set, a relatively new variant of GA, namely, fitness sharing based replacement genetic algorithm (FSR-GA[1]) is employed where some individuals are replaced periodically based on a fitness sharing method. The algorithm is utilised to extract dynamic model of 1-DOF (degree of freedom) motion of a flexible manipulator system. A comparative assessment between FSR-GA and conventional GA is presented in the same application to highlight the novelty of the used GA. Results show that the FSR-GA significantly improves the searching capability of the optimisation process compared to conventional GA. Time domain and frequency domain results clearly reveal the potential of the proposed method in modelling flexible manipulator systems.

Keywords: Dynamic modelling, fitness sharing, genetic algorithms, manipulator system.

I. Introduction

Flexible robot manipulators are widely applied in industrial practice. Dynamic modelling and identification of flexible manipulator systems are of considerable interest in many engineering and scientific applications[2,3,4,5]. The structural flexibility leads to a high degree of elastic vibration especially during high-velocity manoeuvre of the manipulator. Also, some nonlinear phenomena such as joint friction will play more important role in the dynamics of a lightweight manipulator. Furthermore, the dynamic equations of motion are nonlinear and of large dimensions. These problems aggravate the difficulty of the modelling and control of flexible manipulators[2].

Dynamic modelling is the model estimation process of capturing system dynamics using measured data[6]. Since soft computing algorithms are bio-mimetic-based strategies and can handle qualitative techniques with no mathematical model, they are easily applied to complex systems. After David Goldberg[7] gave a basic framework of GAs in his popular book "Genetic Algorithms in Search, Optimisation and Machine Learning", there has been growing interest among scientists and engineers in the use of GAs. Although a large volume of work has been reported in recent years in various application areas[8], little work has been reported in dynamic modelling of flexible manipulator systems using GA[5]. Assuming that only input–output measurements of unknown dynamic systems are available; this paper presents a dynamic modelling technique of flexible manipulator system using fitness sharing based replacement genetic algorithm (FSR-GA). A comparative assessment of FSR-GA with conventional GA in modelling context is also presented.

II. Genetic Algorithm with Fitness Sharing Based replacement policy

GAs, introduced by Holland[9], are global, parallel, search and optimisation methods, founded on the principles of natural selection and population genetics. One of the most important factors that determines the performance of the GA, especially in multimodal problems, is the diversity of the population[10]. In order to maintain higher diversity in the population set, a new variant of GA algorithm, FSR-GA is proposed[1].

Fitness sharing

Fitness sharing lowers the fitness of each element of the population by an amount nearly equal to the number of similar individuals in the population. Typically, the shared fitness f_i' of an individual i with fitness f_i is simply[10]

$$f_i' = \frac{f_i}{m_i} \tag{1}$$

where m_i is the niche count which measures the approximate number of individuals with whom the fitness f_i is shared. The niche count is calculated as[11]:

$$m_i = \sum_{j=1}^{N} sh\left(d_{ij}\right) \tag{2}$$

where N denotes the population size and d_{ij} represents the distance between individuals i and j. The sharing function (*sh*) measures the similarity level between two elements of the population and is calculated as[11]:

$$sh(d_{ij}) = \begin{cases} 1 - \left(\dfrac{d}{\sigma_s}\right)^{\alpha}, & if \quad d \le \sigma_s; \\ 0, & otherwise. \end{cases} \qquad (3)$$

where σ_s denotes the threshold of dissimilarity (the niche radius) and α is a constant parameter which regulates the shape of the sharing function. In most applications, an α =1 or 2 is used and σ_s must be set right to define the niche size[12]. For phenotypic sharing, the Euclidean distance d_{ij} between two variable vectors $X^{(i)}$ and $X^{(j)}$ can be calculated as[11]:

$$d_{ij} = \sqrt{\sum_{k=1}^{n}\left(x_k^{(i)} - x_k^{(j)}\right)^2} \qquad (4)$$

Using normalised distance values[11]:

$$d_{ij} = \sqrt{\sum_{k=1}^{n}\left[\left(x_k^{(i)} - x_k^{(j)}\right)/\left(x_k^{(U)} - x_k^{(L)}\right)\right]^2} \qquad (5)$$

where $x_k^{(U)}$ and $x_k^{(L)}$ are upper and lower limits of parameters.

Replacement policy and period

In fitness sharing, individuals in the crowded region reduce fitness values of one another and thus shared fitness, f_i', reduces significantly depending on the value of niche radius, σ_s. As a result individuals with lower shared fitness values indicate that they belong to crowded region in the solution space and larger shared fitness values indicate that the individuals remain in less crowded regions. A certain percentage of the population, say N_{rep}, residing in the most crowded region are identified based on the lower shared fitness value. Then these individuals are removed and the same number of new individuals is introduced in the population. These newly introduced individuals (solutions) are evaluated in the problem domain and corresponding objective functions are calculated and the whole objective space of the population is updated. This process is repeated after every predefined number of generations, say GEN_{rep}. In FSR- GA, the percentage of total population to be replaced, N_{rep}, and the of period of generation for replacement, GEN_{rep} should be chosen by trading-off between performance and computational cost.

Algorithm pseudo-code: Algorithm pseudo-code for FSR-GA is as follows:

```
1) Initialisation: Number of individuals = $N$ (population size), Maximum number of generation = $MAXGEN$ and initialise generation counter, $iter = 1$, Number of variables = $NVAR$, Number of binary bits to represent each parameter = $NBIT$, Generation gap = $GGAP$, Probability of mutation $p_m$, Probability of crossover, $p_c$
    Initialise $N_{rep}$ and $GEN_{rep}$
2) Generate a random binary population, $CHROM$ of size $N \times NVAR \times NBIT$
3) Decode $CHROM$ to real value within specified
    range of $\lfloor X^L, X^U \rfloor$ and thus create real valued
    population, $PHEN$
4) WHILE $(iter \le MAXGEN)$ DO
a) Evaluate objective function for every individual (each row of $PHEN$ )
b) Select $GGAP$ % of fit individuals based on stochastic uniform sampling method
c) Reproduce individuals using crossover (binary)
d) Mutation (binary)
e) Evaluate children
f) Reinsertion
g) Update population $CHROM$
h) IF $(iter\ MOD\ GEN_{rep} = 0)$ DO
(i) Decode $CHROM$ to real value within $\lfloor X^L, X^U \rfloor$ to
    form $PHEN$
(ii) Calculate the shared fitness of each individual using
    equations (1) – (4)
(iii) Save the best individual (solution), found so far
(iv) Sort individuals based on their shared fitness values
    in the ascending order.
(v) Identify the first ( $N_{rep}$% of $N$ ) individuals
    according to shared fitness values.
(vi) Generate a random binary population, $CHROM_{rep}$
    of size $(N_{rep}\% N) \times NVAR \times NBIT$
(vii) Decode $CHROM_{rep}$ to real value within
    $\lfloor X^L, X^U \rfloor$ to form $PHEN_{rep}$
(viii) Evaluate objective function of each individual of
     $PHEN_{rep}$
(ix) Update population $CHROM$ by replacing
     individuals as identified in (v) with $CHROM_{rep}$;
     END
    i) Increment generation counter $iter := iter + 1$;
    END WHILE
```

III. Genetic Algorithms in Modelling

The basic schematic diagram is shown in Figure 1. Generally a dynamic modelling or system identification problem is formulated as an optimization task where the objective is to find a model and a set of parameters that minimize the prediction error between system output $y(t)$, i.e. the measured data, and the model output $\hat{y}(t,\hat{\theta})$ at each time step t. The process consists of two subtasks; a) structural identification of the equations in the model M, and b) identification of parameters $\hat{\theta}$ of the model. The sum of squared error (SSE) is a commonly used measure of the prediction error.

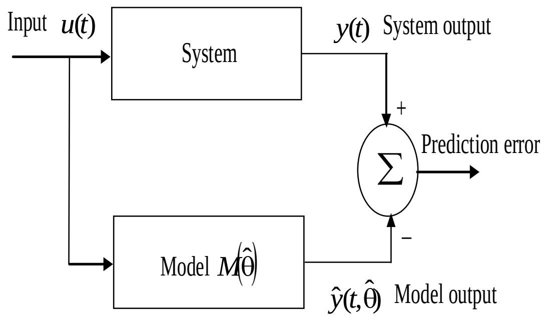

Fig. 1. Basic schematic diagram for modelling

Dynamic modelling of a single-link flexible manipulator

The experimental rig, as shown in Figure 2, is equipped with a U9M4AT type printed circuit motor driving the flexible manipulator[13]. This motor drive amplifier (current amplifier) delivers a current proportional to the input voltage. It serves as a velocity/position controller as well as a motor driver. The measuring devices used to record the various responses of the manipulator are shaft encoder, tachometer and accelerometer along the arm. The shaft encoder is used for measuring the hub-angle of the manipulator. A precision interface circuit PCL 818G is used to interface the flexible manipulator system with a computer.

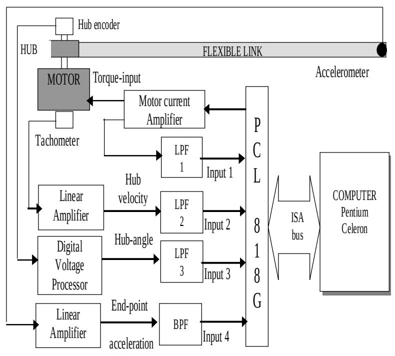

Fig. 2. Block schematic diagram of the experimental rig

The tachometer is used for measurement of the hub angular velocity of the manipulator. The accelerometer is located at the end-point of the flexible arm measuring the end-point acceleration. In this study, an aluminium type flexible manipulator of dimensions 900×19.008×3.2004mm^3, E=71×10^{9}N/m^2, I=5.253×10^{-11}m^4, ρ=2710kg/m^3, and I_h=5.8598×10^{-4}kgm^2 is considered[13].

Preliminary experiment

The flexible manipulator was excited with a sequence of psudo-random binary signal (PRBS), within ±0.1 volts and bandwidth (0-100Hz) so as to ensure that all system resonance modes within this range of frequencies are captured[3]. The system was run for 15s. and 1500 input-output data points were recorded at a sampling rate of 0.01s. The input and corresponding hub angle response are shown in Figures 3 and 4 respectively. Out of 1500 data points 300 are used for modelling and the next 200 for validating the model.

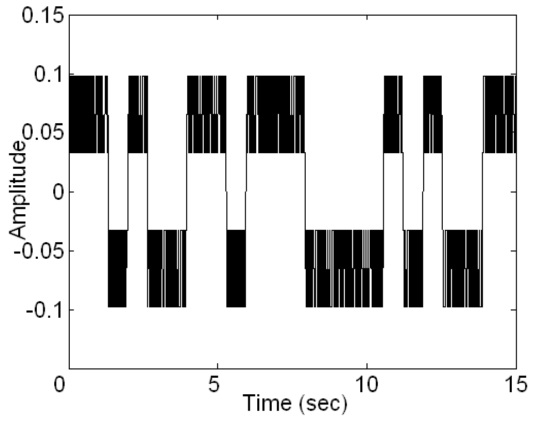

Fig. 3. PRBS input (time domain)

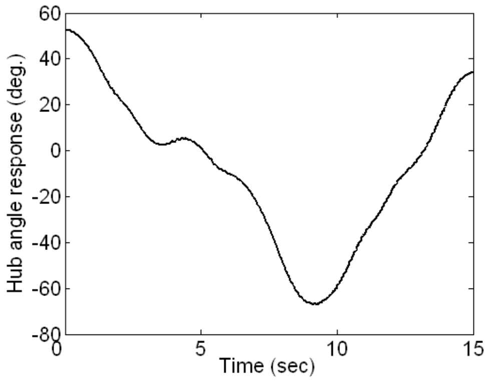

Fig. 4. Hub angle response

Structure formulation

Considering performance and computation costs, an autoregressive moving average (ARMA) structure is chosen to model the flexible manipulator from input to hub angle response. This is expressed as[6]

$$\hat{y}(k) = -\sum_{i=1}^{N} a_i \times y(k-i) + \sum_{j=0}^{M} b_j \times u(k-j) + \eta(k) \quad \ldots\ldots (6)$$

where a_i, b_j are denominator and numerator coefficients, N and M are number of coefficients in the denominator and numerator, y, u, $\hat{y}$, and η are measured output, input, predicted output and noise respectively. The order of the transfer function depends on N. Taking the values of N and M as 4 and 3 and neglecting the noise term η, equation (6) can be simplified as:

$$\hat{y}(k) = -a_1 y(k-1) \ldots - a_4 y(k-4) + b_0 u(k-1) \ldots + b_3 u(k-4) \quad \ldots.. (7)$$

In matrix form, the above equation can be written as:

$$\hat{y}(k) = -[a_1, a_2, a_3, a_4][y(k-1), y(k-2), y(k-3), y(k-4)]^T$$
$$+ [b_0, b_1, b_2, b_3][u(k-1), u(k-2), u(k-3), u(k-4)]^T \quad \ldots. (8)$$

where T represents transposition. This can be further simplified as:

$$\hat{y}(k) = \hat{\theta} \times \varphi(k) \quad \ldots (9)$$

where $\hat{\theta}$ is a row vector that contains the estimated parameters of the model as indicated in Figure 1 and expressed as $\hat{\theta} = [a_1, a_2, a_3, a_4, b_0, b_1, b_2, b_3]$ whereas $\varphi(k)$ is a column vector that contains previous output and input experimental data points as

$\varphi(k)$ *=[-y(k-1, -y(k-2), -y(k-3), -y(k-4), u(k-1), u(k-2), u(k-3), u(k-4)]*^{T}

Parameter Optimization

The GA optimisation process begins with a randomly generated initial population called chromosomes. Randomly generated binary codes of dimension 50×8×16 are created where the number of individuals and parameters in each individual are 50 and 8 respectively. Each parameter is encoded as 16 bit binary code which is logarithmically mapped into real numbers as specified within ranges of the real numbers are defined as -1 to +1, i.e., the randomly generated 16 bit binary codes for each parameter falls within -1 to +1 when converted into real numbers. Each individual or row represents a solution where the first four elements are assigned to $b_0, \ldots, b_3$ and the next four to $a_1, \ldots, a_4$ as indicated in equation (8). The predicted output $\hat{y}$, at any sample instant, is calculated based on equation (8) and taking the elements of first chromosome, actual input and output data. Subsequent predicted outputs are calculated in the same way with the same parameters while taking consecutive input and output data. The difference between the predicted and actual output is recorded as error $e(k) = y(k) - \hat{y}(k)$, which in turn is used to form the objective function $f(x)$ of the optimization process. In this work, sum of absolute error is chosen as the objective function. This is given as:

$$f(x) = \sum_{k=1}^{n} \left| y(k) - \hat{y}(k) \right| \quad (10)$$

where $n = 300$. After evaluating all individuals in the objective domain, fit individuals are selected based on stochastic universal sampling selection technique to form the mating pool[12]. The number of individuals in the mating pool depends on the generation gap $GGAP$; for example, if $GGAP$ = 80% and the number of individuals in the population set is 50, then 80% of the total individuals, i.e. 40 are selected for mating. Genetic operators such as crossover, mutation and reinsertion are applied to form the new population for the next generation. For selection, the stochastic universal sampling technique is used whereas shuffle crossover with reduced surrogate technique is used for crossover[14]. The probability of crossover, p_c, is set at 80%. Binary uniform mutation is used where the probability of mutation, p_m, is set at 0.001%. In order to maintain diversity in the population, shared fitness of all individuals of the current population is calculated after every GEN_{rep} generations. The shared fitness values become lower for densely populated individuals whereas solutions that are widely separated from each other have higher values. Then solutions are sorted based on their shared fitness value and the first N_{rep}% of solutions are identified with lower values. These solutions are replaced with newly generated random solutions as defined by the initial field descriptor[14]. These solutions are evaluated and genetic operators are applied as usual to continue the GA optimization process. It is mentioned that the whole modelling process using GA optimization is encoded and implemented in Matlab[15] and GA toolbox[14].

IV. Results: Effect of Niche Radius

The performance of fitness sharing based GA depends on suitable selection of the niche radius (σ_s)[10,12]. The optimal value of σ_s is selected heuristically. Figure 5 shows the convergence of FSR-GA algorithm with different σ_s. In order to compare the performance of FSR-GA with conventional GA, the convergence of conventional GA in the same problem domain is shown in the same figure. It is clearly evident from Figure 5 that, convergence of FSR-GA varies with σ_s and gives better performance with niche radius of 0.9.

Selection of N_{rep} and GEN_{rep}

In FSR-GA, the percentage of total population to be replaced, N_{rep}, and the period of generation for replacement, GEN_{rep} are two important parameters. The optimum values of N_{rep} and GEN_{rep} are selected based on a trade-off curve drawn in a two-objective domain, namely, best objective function obtained and central processing unit (CPU) execution time of the optimisation process for a specific number of generations. In this process, FSR-GA was run several times with the same GA parameters, such as, number of individuals, objective function, total number of generations, crossover, mutation, niche radius (σ_s = 0.9) etc but different values of N_{rep}, GEN_{rep} and two performance measures; best objective function and CPU execution time were recorded. FSR-GA was run for a maximum generation of 500. In order to select optimum values of N_{rep} and GEN_{rep}, a trade-off curve is introduced (see Figure 6) where all runs of FSR-GA are presented in a two-dimensional space where CPU execution time is plotted along the horizontal axis and best objective function is plotted along the vertical axis. The circles in Figure 6 represent the location of solution and associated numbers indicate the run number. It is clearly evident that, solutions for run-5, run-15, run-12 and run-16 dominate other solutions in the 2-dimensional performance space. From Figure 6, it is easier to select a particular solution under a specific trade-off condition between the two performance measures. In this work, the values of N_{rep} and GEN_{rep} correspond to run-5 are selected and for run-5, both the values of N_{rep} and GEN_{rep} are 10, i.e. 10% of the total individuals in the population set to be replaced after every 10 generations in FSR-GA.

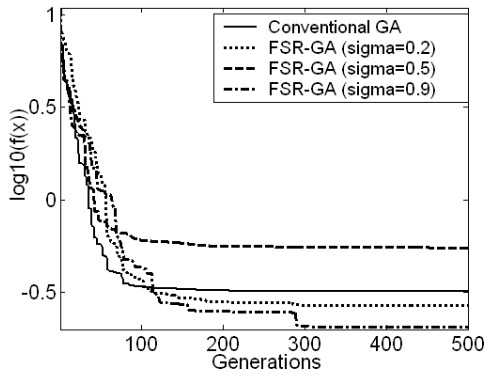

Fig. 5. Convergence of convention GA and FSR-GA for different values of σ_s

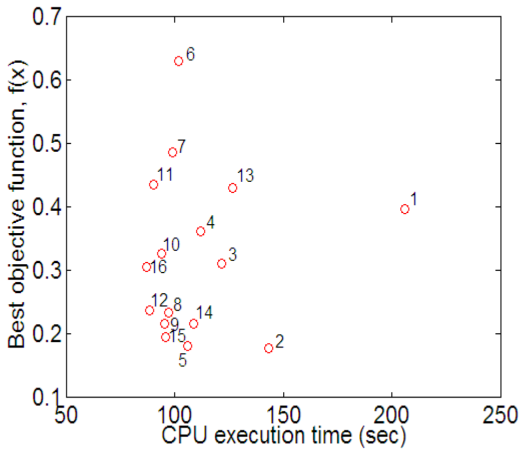

Fig. 6. Trade-off curve of different solutions of FSR-GA

V. Model Formulation and Validation of Flexible Manipulator

The FSR-GA was run for 500 generations with 50 individuals. Considering the convergence to minimum objective function, parameters of GA optimisation process for niche radius of 0.9 has been selected to formulate the model hub angle response of the flexible manipulator. To improve and maintain higher density in the population set, a replacement policy based on fitness sharing technique is invoked, as discussed, where N_{rep} and GEN_{rep} are set at 10, i.e., 1/10th of the total solutions (population set) is replaced after every 10 generations. Using eight parameter values, $b_0,\ldots,b_3$ and $a_1,\ldots,a_4$ obtained at the end of maximum generations and employing relevant Matlab[15] functions the transfer function is formed. The discrete transfer function for input to hub angle output of the flexible manipulator at a sampling time of 0.01sec is as follows:

$$H(z) = \frac{-0.0011z^3 - 0.002933z^2 + 0.002933z + 0.006112}{z^4 - 0.7063z^3 - 0.6015z^2 + 0.2101z + 0.09774} \quad (11)$$

The derived model was validated with a separate input-output data set, and the actual and one-step-ahead predicted outputs are shown in Figure 7. Time domain tracking reveals that the predicted output follows the actual output very well. The frequency domain plots (Figure 8) of the predicted and actual outputs indicate that the model has successfully captured the system dynamics, especially the first three main dominance modes. The pole-zero diagram (Figure 9) shows that all poles lie inside the unit circle while some zeros remain outside. This indicates that the model is stable and non-minimum phase.

Comparative Assessment

A comparative assessment is presented in this section where performance of the FSR-GA, is compared with conventional GA. The performance measure considered here is based on the best objective function at the end of certain number of generations and convergence of the algorithm. Both algorithms were run several times with same parameters. Firstly, each algorithm with a population of 50 individuals was run 10 times, each time with a maximum of 500 generations. In each run, the best objective function and CPU execution time at the end of generation were recorded. Figure 10 shows performance measures of different GAs at generation 500. It is observed that the values of best objective function values obtained with conventional GA were quite higher compared to FSR-GA. It is noted that, in all runs the best objective function values obtained with FSR-GA were lower than conventional GA.

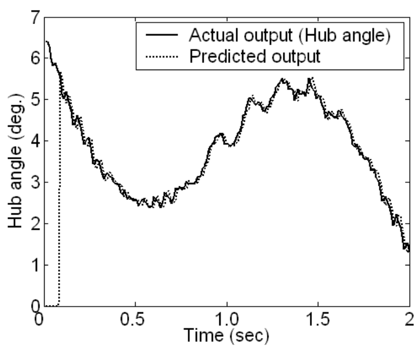

Fig. 7. Actual and predicted output of hub angle response (time domain)

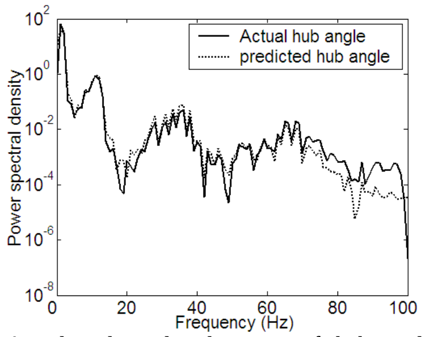

Fig. 8. Actual and predicted output of hub angle response (frequency domain)

The convergence curves for the two algorithms at different runs are shown in Figure 11. It is observed that the conventional GA converged slowly to higher value in the objective domain. It is noted that FSR-GA converged faster and to lower value compared to conventional GA. The improvement in convergence compared to conventional GA was much higher and this is clearly noted in Figure 11. This better convergence of FSR-GA may be attributed to the additional diversity introduced due to fitness sharing based replacement policy.

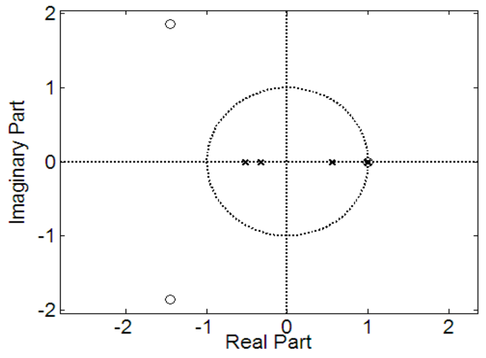

Fig. 9. Pole-zero diagram

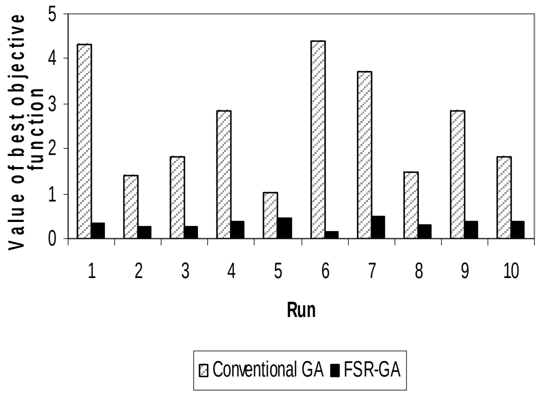

Fig. 10. Comparative assessment

VI. Conclusion

A relatively new variant of GA, namely, FSR-GA has been employed and tested in dynamic modelling of flexible manipulator system. In FSR-GA, a replacement policy based on fitness sharing technique is incorporated with conventional GA operators. The percentage of total population to be replaced and the period of generation for replacement are selected heuristically generating trade-off curve between two conflicting objectives; best objective function obtained and CPU execution time. Results showed that the FSR-GA yielded good result when both parameters were set at 10. i.e. 10% of total population to be replaced after every 10 generations.

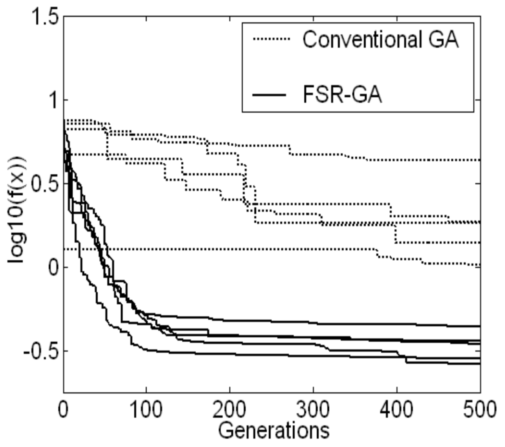

Fig. 11. Convergence of different GAs for different runs at generation 500

In this work, modelling of motion has been formulated as minimization problems with 8-dimentional searching space where FSR-GA has been used to estimate parameters of the model so as to minimize the prediction error between system output, i.e., the measured data, and the model output at each time step. From modelling results, it is evident that the algorithm, with same parameters, can extract stable and satisfactory model for hub angle response of a single-link flexible manipulator. The time-domain and frequency-domain results of modelling have clearly revealed the effectiveness of the modelling approach and the FSR-GA in characterising flexible manipulator system. The performance of FSR-GA has been assessed in comparison with conventional GA. It has been observed that FSR-GA can significantly improve the search ability in terms the objective function and the convergence in the search space compared to conventional GA.

……………………

1. Alam, M.S. and M.O. Tokhi, 2007, Design of a command shaper for vibration control of flexible system: a genetic algorithm optimisation approach, *Low Frequency Noise, Vibration and Active Control*, **26(4)**, 295-310.
2. *Flexible robot manipulators: modelling, simulation and control*, 2008, Edited by M.O. Tokhi and A.K.M. Azad, Control Engineering Series 68, The Institute of Engineering and Technology, UK.
3. Alam, M.S. and M.O. Tokhi, 2007, Dynamic modelling of a single-link flexible manipulator system: a particle swarm optimisation approach, *Low Frequency Noise, Vibration and Active Control*, **26(1)**, 57-72.
4. Zebin, T. and M.S. Alam, 2010, Dynamic Modelling and Fuzzy Logic Control of a Two-link Flexible Manipulator using Genetic Optimization Techniques, *The 13th International Conference on Computer and Information Technology (ICCIT 2010)*, Dhaka, Bangladesh, Dec. 23-25.
5. Shaheed, M.H., M.O. Tokhi, A.J. Chipperfield, and A.K.M. Azad, 2001, Modelling and open-loop control of a single-link flexible manipulator with genetic algorithms. *Journal of Low Frequency Noise, Vibration and Active Control*, **20(1)**, 39-55.
6. Ljung, L., 1999, *System Identification: Theory for the User*, Upper Saddle River, NJ: Prentice-Hall.
7. Goldberg, D.E., 1989, *Genetic algorithms in search, optimisation and machine learning*, Addison Wesley Longman, Publishing Co. Inc., New York.
8. Tokhi, M.O., M.Z.M. Zain, M.S. Alam, F.M. Aldebrez, S.Z.M. Hashim and I.Z.M. Darus, 2011, Genetic algorithm optimisation and control system design of flexible structures, *Journal of Intelligent System*, **17**, Issue Supplement, ISSN: 0334-1860, 133-168.
9. Holland, J. H., 1975, Adaptation in natural and artificial systems, University of Michigan Press, Ann Arbor MI, University of Michigan Press.
10. Goldberg, D.E., and J. Richardson, 1987, Genetic algorithms with sharing for multimodal function optimization, *J. Grefenstette, (Ed.), Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, NJ: Lawrence Erlbaum Associates, 41-49.
11. Deb K., 2001, *Multi-objective optimization using evolutionary algorithms*, New York; Chichester: Wiley.
12. Deb, K. and D.E. Goldberg, 1989, An investigation of niche and species formation in genetic function optimization, *Proceedings of the Third International Conference on Genetic Algorithms*, 42-50, Morgan Kauffman, San Mateo, CA.
13. Tokhi, M. O. and A.K.M. Azad, 1997, Design and development of an experimental flexible manipulator system. *Robotica*, **15**(Part 3), 283-292.
14. Chipperfield, A.J., P.J. Fleming, H. Pohlheim, and C. Fonseca, 1994, Genetic algorithms toolbox user's guide, *Research Report 512*, Dept. Automatic Control and Systems Engineering, University of Sheffield Sheffield, UK.
15. MATLAB Reference Guide, The Math Works, Inc., 2010.