Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

COMPARISION BETWEEN GENETIC APPROACH AND ISLAND APPROACH PARALLEL GENETIC ALGORTHM TO THE POLE PLACEMENT IN LINEAR STATE SPACE SYSTEM

Abhishek Pratap Singh[1],Prof. Ashis Patra[2]

[1]*Electrical Engineering Dept., Madhav Institute of Technology and Science*
[2]*Electrical Engineering Dept., Madhav Institute of Technology and Science*

Abstract— This thesis describes an island approach for shaping the dynamic responses of linear state space systems through pole placement. This paper makes additional comparisons between this approach and genetic approach.

Both approaches generate a gain vector K. The vector K is employed in state feedback for modifying the poles of the system thus on meets step response needs like settling time and percent overshoot. To obtain the gain vector K by the proposed genetic approaches, a pair of ideal, desired poles is calculate first. Those poles function the idea by which an initial population is created. In the island approach, those poles function a basis for n populations, where n is the dimension of the necessary K vector.

Keywords—Genetic Algorithm,Pole Placement, State Space Models,Mutation Vector,Island Approach

I. INTRODUCTION

Pole placement by output feedback is separated into pole placement by state feedback and observer pole placement. Since both problems are dual, only the state feedback case is worked out in details. In the single-input case, the pole placement problem has a unique solution. The design of modeling a system is prediction. A well-defined model can predict a system's behavior given any motivation. A good prediction can be used during the design phase, to build a system which will behave in a desired fashion. Such a model, and the predictions made from it, are used well in advance and familiarize the design of a system.

The improvements discussed in this paper are in dividing up that search to multiple, specialized areas, while still maintaining a conversion of well answers in each area. It is through the consolidation of these techniques that the search ends faster, and the desired feedback to control the system is achieved in less time.

The aim of this thesis is to compare a basic genetic algorithm with an improved algorithm. Both algorithms are set to the same task, which is modifying the behavior of an existing system to a set of performance criteria. They both achieve this by searching for appropriate feedback to added feature the input. Many different systems, with many different performance criteria are tested.

II. STATE SPACE MODELS

A system is loosely outlined as a bunch of often interacting bodies. [2] One terribly basic means of observing a system is by perceptive its behavior with reference to time (t).

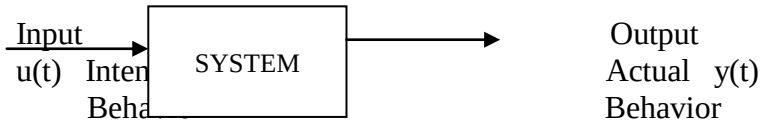

Figure 1: Block Diagram of simple system

Figure 1 shows the input to the system as u(t) and also the output of the system as y(t). The system block encompasses all the interacting bodies' middle. During this terribly basic model, nothing is thought apart from what goes in and what comes out.

Once the behavior of a system (given a predefined input) is thought, it may be controlled. Once a user plans the input of a system with the goal of manufacturing a selected output, they're aforesaid to require management of that system.

This paper is simply involved with control system systems, and is targeted at finding the quantity of internal feedback which will bring the particular performance of a system nearer to the supposed performance of a system. To attain this correct combination of feedback, it's necessary to model the system in what's referred to as state house. A state house model is additional advanced than the higher than diagrams, therein it details additional of the interior workings of a system. Not solely will a state house model monitor a number of the interior variables, these state house models incorporate a way of feedback into the system. During this means, additional info is gathered concerning the inner workings of the system.

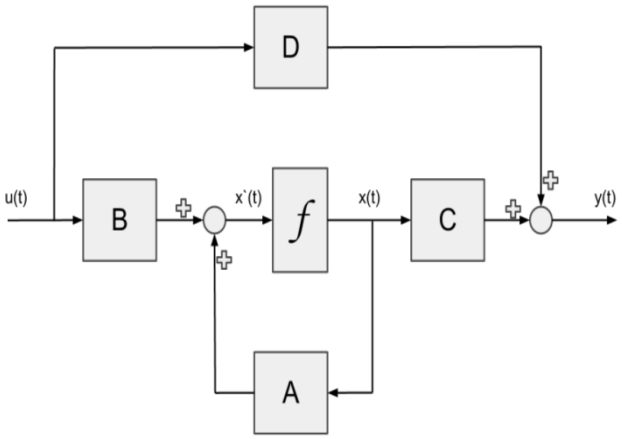

Figure 2: State Space Modal Diagram

A state area model could be a method of mathematically relating the system's input the inner elements of the system, and also the system's output. The instance system shown in Figure 3 has been de-escalated into 3 distinct functions.

- *u(t)* describes the input to the system
- *x(t)* describes the inner workings of the system
- *y(t)* describes the output from the system

This new element, x(t), is that the state vector of the model. This vector contains all of the state variables for the model. It's additionally shown within the figure higher than that some components of the inner workings, ẋ(t) depend on each the input u(t) and additionally x(t). ẋ(t) is the time rate of change of the state variables, or $\frac{dx(t)}{dt}$ In a simple RLC circuit, these variables could also be voltages or currents, that don't seem to be directly controlled by the user input, however some combination of the signaling and their current state.

A state space model is characterized by this state vector, and its relationships to the other components. In order to represent a system in a state space model, it is necessary to be able to define the derivatives of each of the elements in the state vector as a linear combination of the values of the other elements in the state vector and some input to the system.

The general form of which is:

$$\dot{x}(t) = Ax(t) + Bu(t) \qquad (1)$$

Where ẋ(t) is a vector containing all of the derivatives of all the state variables at some time t. The A matrix, then, would be a matrix of coefficients which are multiplied by the elements in the state vector x(t), and B is a vector relating the input u(t) to ẋ(t).
The system output is similarly composed of some part of the internal variables x(t) and some direct mapping from the input to the output.
Y(t) = Cx(t) + Du(t) (2)

III. POLE PLACEMENT METHOD

pole placement, could be a methodology utilized in feedback system theory to put the closed-loop poles of a plant in pre-determined locations within the s-plane.[4] Inserting poles is fascinating as a result of the situation of the poles corresponds on to the eigenvalues of the system, that management the characteristics of the response of the system. The system should be thought of manageable so as to implement this methodology.
The state area model illustration of the system could be a powerful tool as a result of abundant of what quantity data will be foreseen from the A matrix itself. Since the A matrix features a variety of rows up to the amount of parts within the x(t) vector, and additional variety of columns up to that very same number of parts, A is often a matrix. A system's behavior will be foreseen in giant half by the Manfred Eigen values of the A matrix. Eigen values are defined as the solution(s) to the equation:

$$\det(A - \lambda I) = 0 \quad (3)$$

Or, more graphically:

$$\det\left(A - \begin{bmatrix} \lambda & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda \end{bmatrix}\right) = 0 \quad (4)$$

The determinant of the ensuing matrix is often stated because the "characteristic equation" of the matrix. The solutions for λ square measure normally referred to as the "poles" of the system. If an equivalent system were instead drawn as a transfer function, the Eigen values of this A matrix would correspond to roots within the divisor of the transfer function.
Additional feedback channel is additional to the input of the equation, one that's some combination of the system's actual state alongside the user input, the A matrix are often effectively altered, and therefore the system's performance will thereby amendment, while not meddling with the prevailing system or model. This "full state feedback" or "pole placement" may be a classic methodology up to the mark theory, developed partially by Eduardo sontag. [4] The subsequent substitution is formed.
u(t) = r(t) – Kx(t) (5)
For simplicity's sake, the user input has been renamed from u(t) to r(t) in order that the combined input continues to be named u(t). The feedback vector K is increased by the state vector, and then subtracted from the user input to make the full input to the system.
The new equation governing the behavior of ẋ(t) in Figure 3 above is:
ẋ(t) = (A-Bk) x(t) + Bu (t) (6)
If the K vector is non-zero, the matrix relating the state variables to their derivatives can have modified. By dynamic the A matrix into (A-BK) the characteristic equation, the system's poles, and thereby its dynamic response have all been altered.

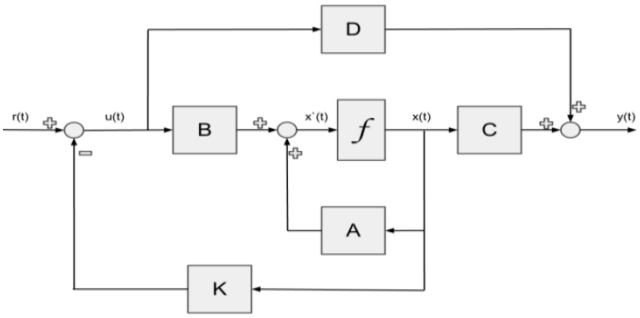

Fig 3: State Space Diagram with K Vector Feedback

It is worth mentioning at this point that so as for a system to be stable the poles of the system should exist within the left half the quantity plane: they need to be non-positive values. A stable system is one that may settle into an equilibrium state. To boot, the poles of those systems typically exist as advanced conjugate pairs, as real numbers, or some combination of each. [4, 6]

Simple algebra can follow on and realize a vector that transforms an existing matrix into one with the required poles (eigenvalues). If the new, desired poles are represented as λi's, then K vector may be found by setting the subsequent adequate the freshly desired characteristic equation.

$$\det ((\lambda I – (A - Bk)) \quad (7)$$

This direct technique of calculation will solely manufacture at the most one complicated conjugate try of poles. If the A matrix is of dimension larger than 2x2, then it'll have additional eigenvalues, more poles, and therefore the remainder of the required poles can need to be guessed at victimization heuristics. Often, it is sufficient to easily select the remainder of the required poles as giant negative real numbers. [7]

IV. GENETIC APPROACH

Genetic algorithms can trace their origins back to Alex Fraser, who explored the thought in 1957. [8] A genetic rule could be a random process; it's somewhat random, however this state isn't totally freelance of the previous states. There's no single way to implement a genetic algorithm, however most of those algorithms adhere to constant structure and embrace similar elements. [9, 8] The algorithms elaborated during this thesis square measure original work built from the essential principles that are long established. Consecutive few sections detail the customization and design method.

The aim of a genetic rule is to seek out a solution to a haul. Several potential solutions square measure generated supported previous solutions and conjointly on injected randomness, until one matches the exit criteria.

This requires that potential solutions (called genomes) be representable in a digital format. Further, a genetic algorithm requires a method of determining the suitability (called fitness) of each potential solution.

Fitness calculation is based on match method. The calculation is designed to find a K vector that will drive the state space model to within 5% of both user provided criteria. In order to match both criteria, a score is given to each (overshoot and settling time) and the maximum between those is given to the genome.

$$fitness = \max\left(\left|\frac{measured\ overshoot - desired\ overshoot}{desired\ overshoot * 0.05}\right|, \left|\frac{measured\ settling\ time - desired\ settling\ time}{desired\ settling\ time * 0.05}\right|\right) + 10 * measured\ undershoot$$

If fitness value is feasible, feedback gain matrix is added to population along with fitness value otherwise discarded. Once the initial population is created it is sorted best on fitness value in increasing order. If the lowest fitness value satisfy user criteria the process is stop otherwise new generation is created again [1].

A. Modifying a Genetic Algorithm to Pole Placement

The main steps in modifying a genetic algorithmic program to a problem area unit ordering choice, fitness calculation, breeding method selection, and mutation selection. The genomes are the attainable solutions to the matter, the fitness method may be a live of however well they solve the matter, the breeding method is how they're combined to create new solutions, and also the mutation method describes how and the way usually new information is injected. Every of those steps are going to be delineate here.

For a state space drawback, the population of genomes is chosen to be feedback gain vectors (referred to within the remainder of this paper as K vectors.) whereas it might have been attainable to create a genome instead that contained the desired poles of the system, instead of the K vector, that approach had many drawbacks.

This has several key implications:

- Each genome will be the same length (contain the same number of elements)
- Each element in the genome will be a real number
- The order of the elements within the genome has an effect on its fitness.

V. MATLAB© IMPLEMENTATION

The genetic algorithm was implemented as a function within MatLab. There were four input criteria, and 2 return values. The first input argument is a state-space model. The percentOS and settling Time arguments are the desired percent overshoot of the impulse response, and the desired 2% settling time respectively. The last input argument is an optional argument to alter the way fitness is calculated. This will be discussed in detail in the fitness section.

function [kVector, fitness Matrix] =GeneticK(systemA, percentOS, settling Time, mode Option)

At the completion of the algorithm, the function will provide the user with a kVector which has been selected to transform the system. That gain vector will drive the system to the performance criteria. GeneticK will also provide a fitness Matrix, which is a two dimensional arrangement of the evolution of the genome population.

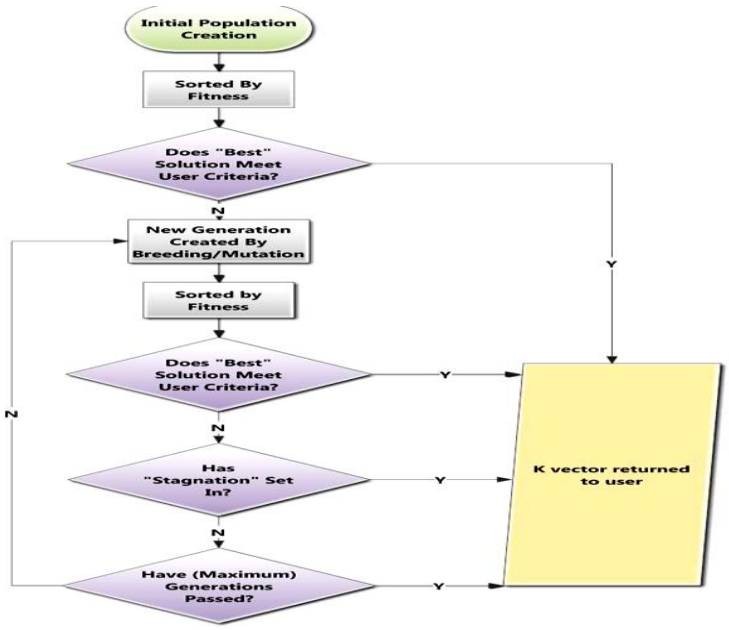

Figure 4: Block Diagram of Genetic Algorithm

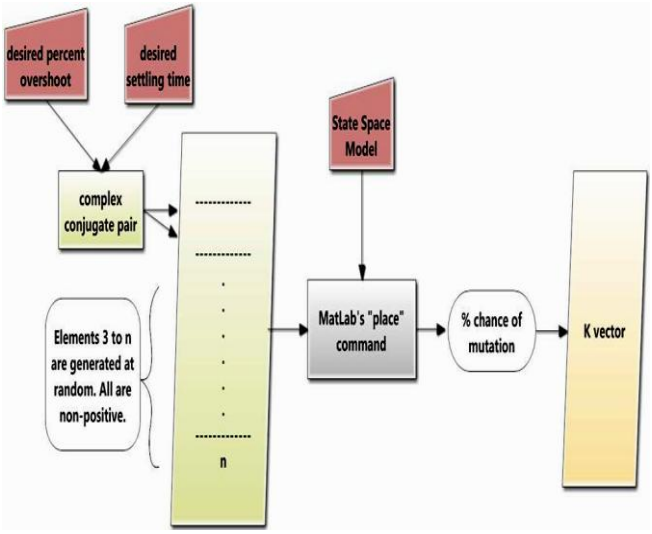

Figure 5: Initial Genome Creation

VI. ISLAND APPROACH

The island approach is mostly similar the approach described and enforced above. It is a separate MatLab© function with constant input arguments and also the same is largely data returned.

function [kVector, fitness Matrix] = Genetic KIsland(systemA, percentOS, settling Time, mode Option)

The purpose of implementing the island approach is to decrease the runtime of the algorithm. Breeding and fitness calculations are computationally intensive, and should be done a large number of times in every generation. In associate island approach, the genetic population is divided up into sub-populations, known as islands that breed locally.

If this task is distributed to multiple computers, every computer will manage all of the breeding, mutation, and fitness calculations of one island at constant time many alternative computers manage the other islands. Although the islands can maintain separate breeding pools, they are not fully isolated from each other. Between breeding cycles the most effective solutions from every island are forced to migrate to neighboring islands in order that sensible genomes still are favored.

Because every island breeds with solely a sub-set of the full population, that island's population expertise some specialization, breeding results that differ from other island populations more and more as every generation passes. Migrating the most effective solutions between islands can keep a targeted provide of the most effective solutions flowing, but for this drawback, a shot has been created to take advantage of every island's potential for specialization.

The initial genomes are subject to a random chance of mutation, unless the system is of size two. In that case each genome is subject to mandatory mutation to assure diversity within the genome population.

One island is formed for every component within the (K vector) genome. Whenever a mutation occurs, every island can handle that mutation process slightly differently. every island is numbered and corresponds to one component within the K vector. The numbered islands can mutate the corresponding component in the ordination to a far bigger degree than the remainder. The total population is divided evenly between each island. The creation process for each genome is the same as discussed earlier in the regular genetic algorithm.

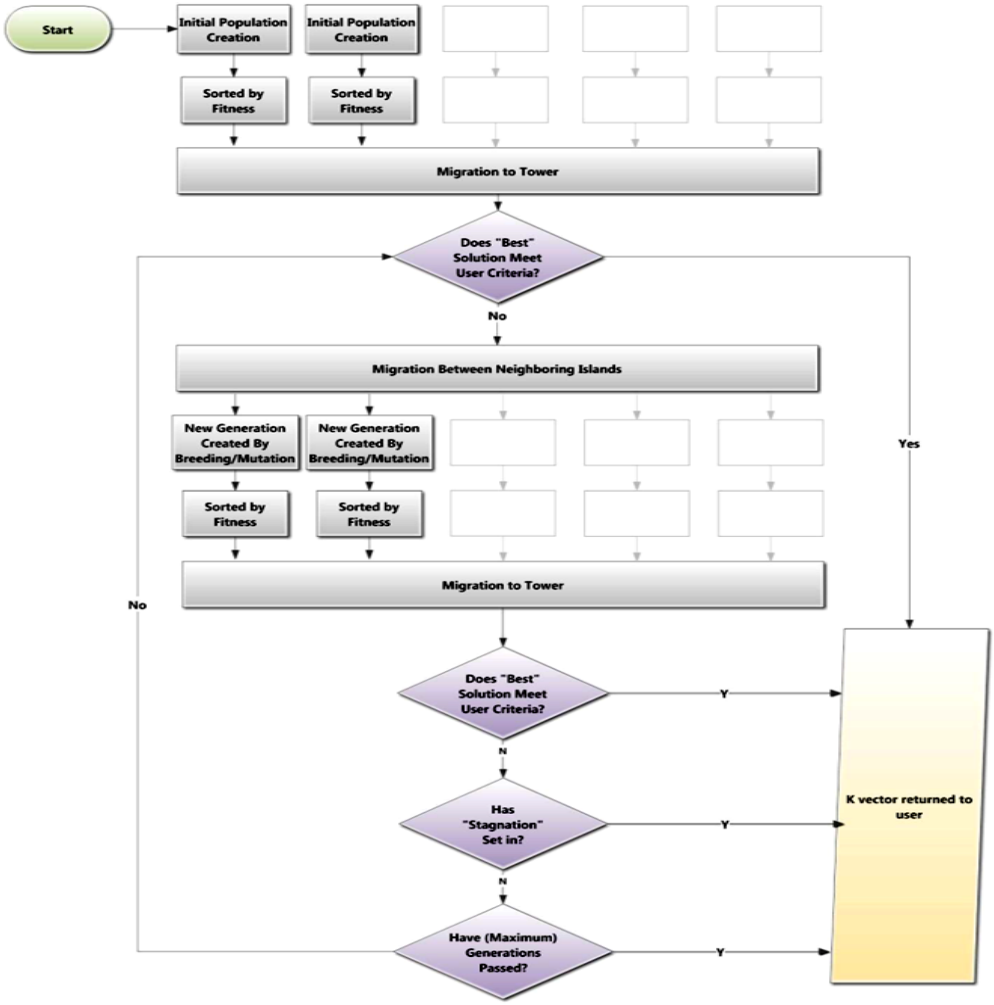

Figure 6: Island Approach Block Diagram

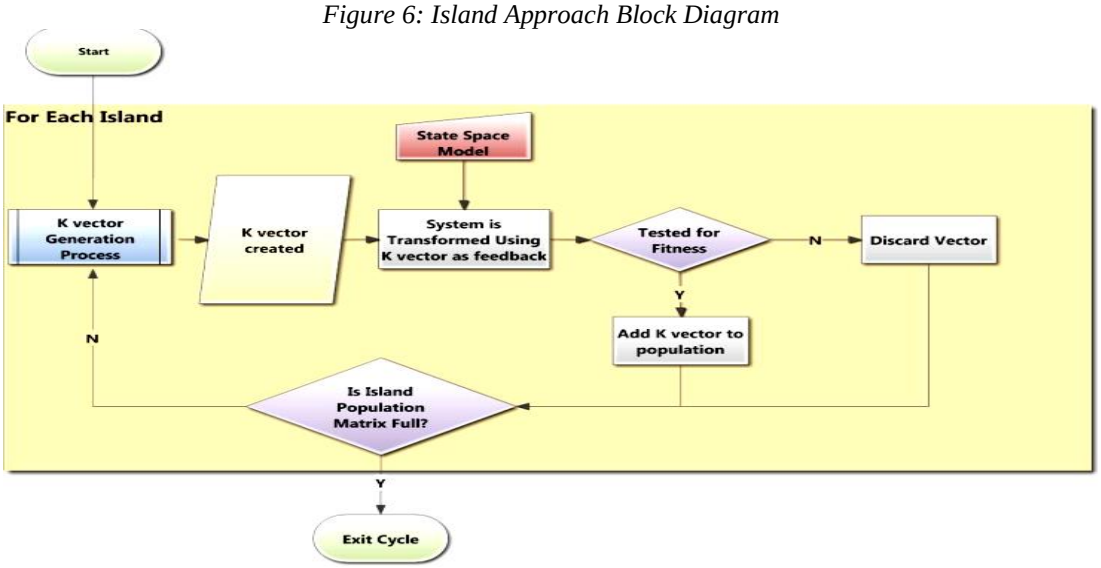

Figure 7: Island Population Creation

VII. RESULT AND ANALYSIS

The island approach, using mutation vectors, demonstrates higher performance than the regular genetic approach in an exceedingly type of conditions. The island approach generally found a solution in an

exceedingly smaller range of generations, and also the final fitness lots of the island approach were higher than those using the regular genetic approach. A comparison of total computation time is additionally created. Whereas the island approach concludes inside fewer generations, it's shown that those generations will take longer to conclude themselves under certain conditions.
Both approaches were able to find good results in most cases. Their performance in every fitness mode is shown here.

a) State Space Model for System 1

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -5 & -6 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$C = [12 \quad 0 \quad 0]$$

$$D = [0]$$

Transfer Function:-

$$H(s) = \frac{12}{s^3+6s^2+5s}$$

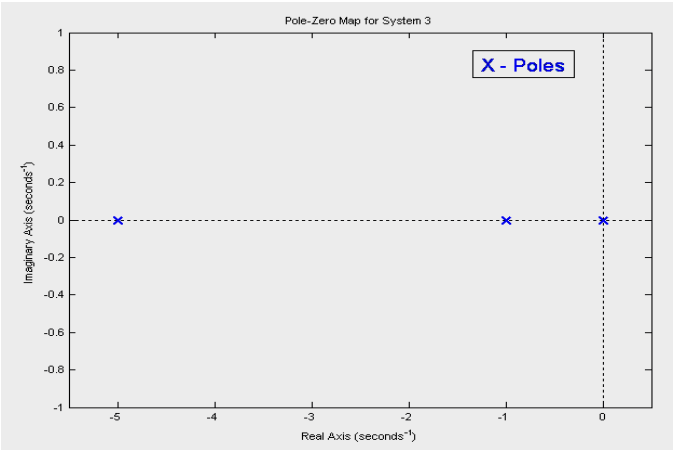

Figure 8: Pole Zero map for System 1

System 1 has three poles and no zeros. It is likely that two directly calculated complex conjugate poles will be enough to dominate, so long as the third pole is sufficiently far from the origin. The desired criteria for tests involving System 3 were Desired Overshoot 7% and Desired Settling Time 4.5s.

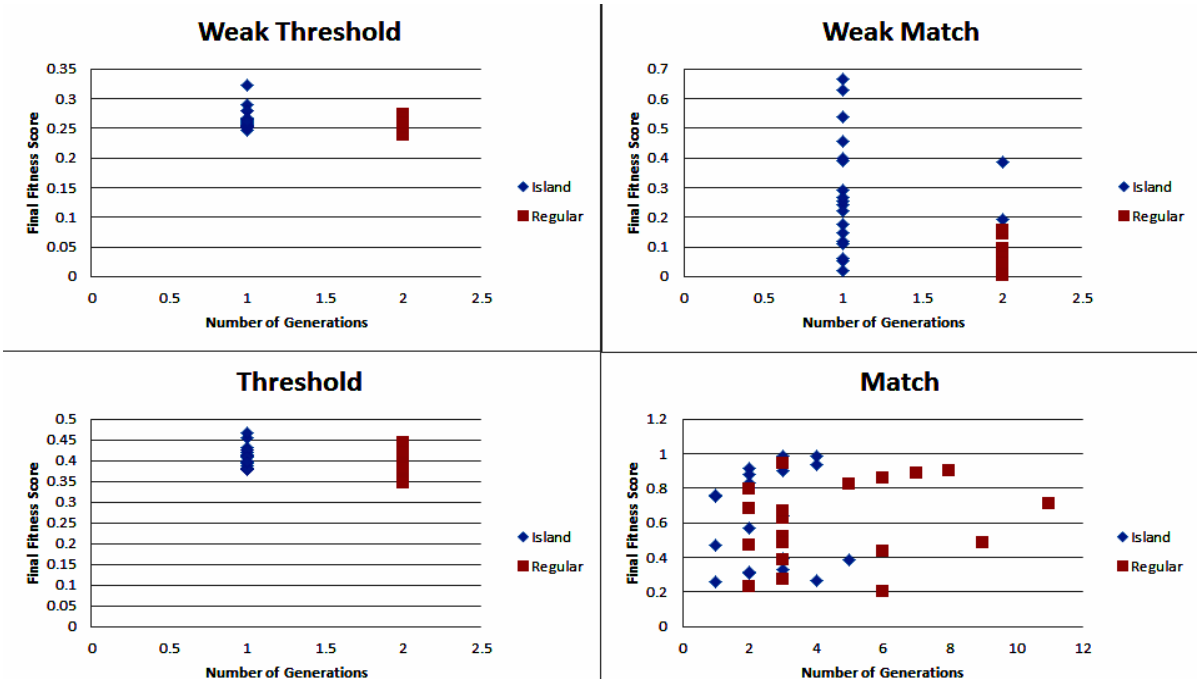

Figure 9: Result from system 1

System 1 failed to present much of a challenge to either system in Weak Threshold, Threshold, or Weak Match mode. Each approaches settled quickly and with remarkably match results. The largest distinction between the two implementations may be seen once more in Match mode. The island approach systematically settled in fewer generations; however it'd be difficult to assert that the island approach was strictly higher overall. There are many cases wherever the regular approach took up to 10 generations longer, however came back the same result because the island approach. Interesting comparisons also can be drawn when the final systems made by the different implementations are analyzed. The subsequent could be a comparison supported the Weak Match mode on System 1:

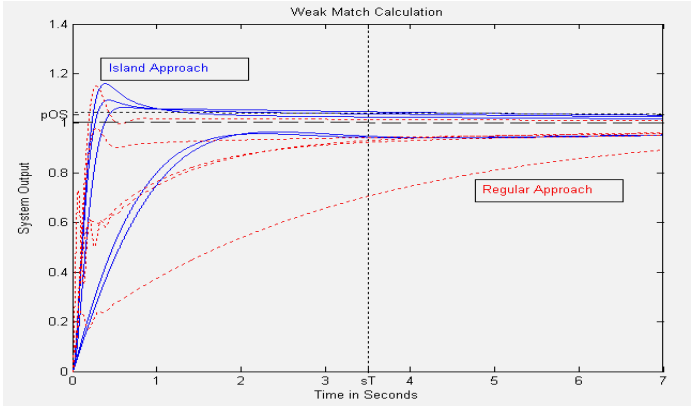

Figure 10: Sample of Island and Regular Implementation Solutions

Island approach has been demonstrated to provide results that are nearly as good, if not higher than the regular genetic implementation. Allow us to now turn to different measures of performance. The amount of your time taken to succeed in the final result was calculated for every instance above. The time taken overall was then divided by the amount of generations that the algorithmic rule searched. For the island

approach, the time was further divided by the amount of islands within the population, because the island approach is meant to be distributed to multiple computers or processors at intervals identical computer.

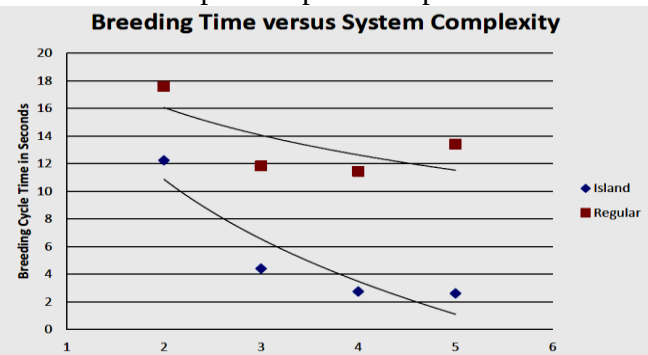

Fig 11: Seconds per Generation as system complexity increases

VIII. CONCLUSION

In this paper a comparative study has been performed between genetic approach and island approach based pole placement. Results show that the performance of island approach parallel genetic algorithm is much better in comparison to the conventional approaches. So island is considered as a good approach to find optimum location of poles for linear state space system.

XI. ACKNOWLEDGMENT

Authors would like to thanks to Dr. Sanjeev Jain, Director, MITS, Gwalior, MP, for promoting this work.

REFERENCES

[1] Y. J. Cao and Q. H. Wu, "Teaching Genetic Algorithm Using MATLAB", Int. J. Elect. Eng. Educ., Vol. 36, pp. 139–153. Manchester U.P., 1999. Printed in Great Britain.
[2] Alpay, Daniel, and I. Gohberg. "State Space Method: Generalizations and Applications." n.p.: Birkhäuser Verlag, 2006.
[3] P. J. Fleming and R. C. Purshouse, "Genetic Algorithm in Control Systems Engineering", CONTROL SYSTEM, ROBOTS AND AUTOMATION- Vol. XVII- *Genetic Algorithms in Control System Engineering.*
[4] Nise, N.S., "Control Systems Engineering," Hoboken, NJ: Wiley, 2004.
[5] K. Ogata, Modern Control Engineering, PHI Learning Private Ltd, Rimjhim House, NJ, 1997, 5th Ed., pp. 70–72 and pp. 739–950.
[6] Kautsky, J. and N.K. Nichols, "Robust Pole Assignment in Linear State Feedback," *Int. J. Control*, 41 (1985), pp. 1129-1155.
[7] Chiu H. Choi, "Step Response Improvement by Pole Placement with Observer," Proceedings of the 40th Southeastern Symposium on System Theory, pp.7-12, Mar. 2008.
[8] Schalkoff, Robert J., "Intelligent Systems: Principles, Paradigms, and Pragmatics." Sudbury, MA: Jones and Bartlett, 2011.
[9] Goldberg, David E. "Genetic Algorithms In Search, Optimization, And Machine Learning," n.p.: Reading, Mass.: Addison-Wesley Pub. Co., 1989.
[10] Guo, Pengfei, Xuezhi Wang, and Yingshi Han. "The Enhanced Genetic Algorithms for the Optimization Design." *2010 3rd International Conference on Biomedical Engineering and Informatics (BMEI 2010)* (2010): pp.2990-2994.
[11] Srinivas, M. and L.M. Patnaik, "Adaptive Probabilities of Crossover And Mutation In Genetic Algorithms." Ieee Transactions On Systems Man And Cybernetics 24.4 (n.d.): 656-667.Science Citation Index.
[12] Varsek, A., T. Urbancic, and B. Filipic, "Genetic algorithms in controller design and tuning," *IEEE Trans. on System, Man, and Cybernetics*, vol. 23, no. 5, 1993, pp.1330-1339.
[13] C. M. Fonseca and P. J. Fleming, "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization", Proc. ICGA 5, pp. 416-423, 1993.
[14] D Kobler, et al. "Parallel Island-Based Genetic Algorithm for Radio Network Design."Journal of Parallel and Distributed Computing 47.1 (n.d.): 86-90.

